ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA MAGISTRALE in INGEGNERIA GESTIONALE

ANOMALY DETECTION IN CONTESTI INDUSTRIALI: IMPLEMENTAZIONE DI MODELLI DI MACHINE LEARNING PER LA CLASSIFICAZIONE DI ERRORI NEI NASTRI TRASPORTATORI

TESI DI LAUREA
in
LABORATORY OF ARTIFICIAL INTELLIGENCE APPLICATIONS M

RELATORE:
Prof. Michele Lombardi

CANDIDATO:
Alberto Trenta

Anno Accademico 2023-2024
Sessione II

INDICE

INTRODUZIONE

La manutenzione predittiva rappresenta uno degli elementi cardine nelle strategie industriali moderne, rivestendo un ruolo cruciale nell'assicurare l'affidabilità delle apparecchiature e nel ridurre i costi associati ai guasti imprevisti. Nell'era dell'Industria 4.0 e della transizione verso l'Industria 5.0, l'integrazione di tecnologie avanzate come l'Internet of Things (IoT), l'Intelligenza Artificiale e il Machine Learning sta ridefinendo i processi produttivi, promuovendo efficienza, sostenibilità e collaborazione uomo-macchina. Tra le infrastrutture industriali più critiche, i nastri trasportatori giocano un ruolo fondamentale nel flusso produttivo di numerosi settori, come quello manifatturiero, minerario e logistico. Questi sistemi, pur essendo essenziali per la continuità operativa, sono suscettibili a diverse tipologie di anomalie, quali l'allentamento della catena o la rottura di giunzioni che possono causare interruzioni costose della produzione e compromettere la sicurezza dell'impianto. La capacità di rilevare ed identificare precocemente tali anomalie è pertanto cruciale per implementare interventi manutentivi proattivi, migliorare l'efficienza operativa e ridurre i rischi associati.

Con l'aumento esponenziale dell'uso di sensori industriali e della quantità di dati raccolti in tempo reale, le tecniche di machine learning si sono affermate come strumenti potenti per la rilevazione automatica delle anomalie. Algoritmi avanzati come Random Forest e XGBoost si distinguono per la loro capacità di gestire grandi volumi di dati, modellare pattern complessi e fornire previsioni accurate anche in ambienti dinamici e complessi. Tuttavia, la sfida risiede nell'ottimizzazione di tali modelli per applicazioni industriali specifiche, tenendo conto delle particolarità dei dati e dei requisiti operativi del settore.

Nel contesto di questa tesi magistrale, viene affrontato il problema della rilevazione delle anomalie nei nastri trasportatori industriali attraverso l'implementazione e l'ottimizzazione di modelli di machine learning.

Il lavoro è stato svolto in collaborazione con il CNR - STIIMA (Consiglio Nazionale delle Ricerche - Istituto di Sistemi e Tecnologie Industriali per il Manifatturiero Avanzato) e un'azienda del settore manifatturiero bolognese, offrendo un'interessante opportunità di applicare tecniche avanzate in un contesto industriale reale.

In particolare, il lavoro di tesi si concentra sulla progettazione, lo sviluppo e l'ottimizzazione di modelli predittivi basati su Random Forest e XGBoost per il rilevamento efficace delle anomalie. Per raggiungere questo scopo, è stato adottato un approccio sperimentale che comprende diverse fasi chiave:

1. **Analisi del Contesto e Raccolta Dati**: Una dettagliata comprensione del funzionamento del nastro trasportatore, dei suoi componenti critici e delle tipologie di guasti frequenti ha permesso di comprendere la struttura dei dati ed identificare la strategia più adatta.
2. **Preprocessing dei Dati**: Data l'elevata frequenza di campionamento e la complessità dei segnali acquisiti, è stata posta particolare attenzione al preprocessing dei dati. Tecniche di riduzione della dimensionalità, come il sottocampionamento e il binning basato su metriche statistiche (media, varianza e skewness), sono state implementate per rendere i dati gestibili senza perdere informazioni critiche. In particolare, l'utilizzo della funzione di autocorrelazione ha permesso di determinare una bin size di 440 campioni, il ché si traduce in una riduzione di circa 3 ordini di grandezza (146 volte) della dimensione dei dati.
3. **Implementazione dei Modelli di Machine Learning**: Sono stati sviluppati modelli di Random Forest e Graduebt Boosted Trees utilizzando librerie Python avanzate, come XGBoost. L'ottimizzazione degli iperparametri è stata effettuata tramite Grid Search, che consente di esplorare sistematicamente lo spazio delle possibili configurazioni e di identificare quelle che massimizzano le prestazioni dei modelli.

4. **Valutazione delle Prestazioni**: I modelli sono stati valutati utilizzando metriche chiave come accuratezza, F1-score, precisione, recall e l'area sotto la curva ROC (ROC-AUC). È stata eseguita una validazione incrociata per garantire la robustezza dei risultati e verificare la capacità dei modelli di generalizzare su dati non visti.
5. **Analisi dei Risultati e Discussione**: L'analisi dei risultati ha dimostrato come sia nel caso di classificazione binaria che di multiclassificazione i modelli raggiungano risultati importanti. In particolare, su dati di test mai visti, Random Forest riesce ad ottenere accuratezze di 99,87% nel caso binario e 99,2% nel caso multiclasse. XGB è comunque molto prestante con il 99,53% nel primo caso e il 98,4% di precisione nel classificare più tipologie di errore, a fronte di una minor richiesta di tempo di addestramento.

Nello specifico, la struttura della tesi è organizzata come segue:

- **Capitolo 1 - Contesto e Motivazione**: Introduce il problema, contestualizza l'importanza della manutenzione predittiva e dell'anomaly detection nell'industria moderna, e presenta il caso studio specifico.
- **Capitolo 2 - Background**: Fornisce una panoramica delle tecniche di machine learning utilizzate, degli strumenti per la data analytics e delle metodologie di valutazione delle prestazioni dei modelli.
- **Capitolo 3 – Preprocessing dei dati**: Descrive in dettaglio il processo di raccolta dati, il preprocessing effettuato e le tecniche utilizzate per preparare i dati per l'analisi.
- **Capitolo 4 - Progettazione e Ottimizzazione del Modello**: Illustra l'implementazione dei modelli di Random Forest e Gradient Boosted Trees, l'ottimizzazione degli iperparametri e le strategie adottate per migliorare le prestazioni.

- **Capitolo 5 - Risultati Empirici**: Presenta i risultati ottenuti dai modelli sui dati di test, analizza le prestazioni e discute i risultati in relazione agli obiettivi prefissati.
- **Capitolo 6 - Conclusioni e Sviluppi Futuri**: Riassume i contributi della tesi, evidenzia l'impatto del lavoro sul progetto complessivo e propone direzioni per future ricerche e implementazioni.

Questo lavoro contribuisce alla letteratura esistente dimostrando l'efficacia delle tecniche di ensemble learning nella rilevazione delle anomalie in contesti industriali reali. Inoltre, fornisce un framework applicabile ad altri sistemi industriali che necessitano di soluzioni avanzate per la manutenzione predittiva. Le implicazioni pratiche includono una potenziale riduzione significativa dei tempi di inattività non pianificati, l'ottimizzazione dei processi di manutenzione e il miglioramento della sicurezza operativa, contribuendo così alla competitività e sostenibilità delle aziende nell'era dell'Industria 4.0 e oltre.

1. CONTESTO E MOTIVAZIONE

1.1 L’Anomaly Detection nell’Industria

L'anomaly detection è una tecnica di analisi dei dati impiegata per identificare eventi o comportamenti anomali che si discostano da uno standard predefinito, noto come "comportamento nominale". Nell'industria moderna, questa tecnologia riveste un ruolo cruciale, specialmente in ambito industriale, dove viene utilizzata per monitorare in tempo reale il funzionamento delle macchine e delle operazioni. L'anomaly detection permette di rilevare tempestivamente malfunzionamenti o deviazioni operative, come guasti meccanici o anomalie nei processi, prima che possano trasformarsi in problemi gravi o emergenze. Questo approccio proattivo permette alle aziende di evitare interruzioni costose e ridurre i tempi di inattività.

Con l'avvento dell'Industry 4.0, l'uso dell'anomaly detection si è esteso grazie all'integrazione di sistemi cyber-fisici (CPS), Internet of Things (IoT) e tecnologie di machine learning, che consentono un'automazione intelligente e decisioni autonome basate sui dati. Le fabbriche intelligenti possono quindi monitorare continuamente lo stato delle macchine e delle linee di produzione, raccogliendo dati dai sensori per identificare in tempo reale qualsiasi deviazione dal comportamento normale. Questo non solo ottimizza l'efficienza operativa, ma riduce anche i costi di manutenzione e migliora la qualità dei prodotti.

Con il passaggio all'Industry 5.0, l'interazione tra uomo e macchina diventa ancora più centrale. L'obiettivo non è solo migliorare l'efficienza, ma anche favorire una collaborazione sinergica tra le capacità cognitive dell'essere umano e la precisione delle macchine intelligenti. In questo contesto, l'anomaly detection non serve soltanto a migliorare i processi industriali, ma svolge anche un ruolo chiave nella sicurezza, garantendo che le macchine funzionino correttamente e segnalando eventuali anomalie che potrebbero compromettere la sicurezza degli operatori. Questa capacità di prevenire guasti e ottimizzare le operazioni contribuisce a creare ambienti di lavoro più sicuri e sostenibili,

rendendo l'anomaly detection uno strumento fondamentale per affrontare le sfide tecnologiche ed economiche del futuro industriale.

1.2 Vantaggi dell'Anomaly Detection

L'implementazione di sistemi di anomaly detection offre numerosi vantaggi strategici per le aziende che operano all'interno dei paradigmi dell'Industry 4.0 e 5.0, in particolare:

1. **Riduzione dei Tempi di Inattività**: In un ambiente produttivo, ogni minuto di fermo macchina può comportare costi significativi, sia in termini di produzione persa che di ritardi nelle consegne. Grazie all'anomaly detection, le aziende possono prevedere guasti futuri e intervenire preventivamente, evitando così fermate impreviste e ottimizzando l'utilizzo delle risorse. Questo approccio predittivo consente una manutenzione pianificata, che risulta essere meno costosa e meno distruttiva rispetto alla riparazione di guasti improvvisi.
2. **Riduzione dei Costi di Riparazione e di Manodopera**: Oltre a evitare la perdita di produzione, l'anomaly detection aiuta a ridurre i costi di riparazione, poiché la manutenzione preventiva è tipicamente meno onerosa rispetto alla risoluzione di guasti emergenziali. Inoltre, la capacità di intervenire prima che si verifichino guasti riduce i costi di manodopera aggiuntivi, poiché il personale non sarà obbligato a gestire situazioni di emergenza che possono interrompere le normali attività lavorative.
3. **Sostenibilità**: L'anomaly detection contribuisce significativamente agli obiettivi di sostenibilità, pilastro dell'Industry 5.0. Monitorando e ottimizzando continuamente i processi produttivi, le aziende possono gestire le risorse in modo più efficiente, cosa che si traduce in una riduzione di sprechi e quindi di impatto ambientale. Le tecnologie di rilevamento delle anomalie possono essere integrate con i sistemi di gestione energetica per monitorare e ridurre il consumo di energia, contribuendo a una produzione

più sostenibile. Questo allineamento con i principi di sostenibilità è particolarmente importante in un'era in cui la responsabilità ambientale è diventata un fattore critico per il successo aziendale.

4. **Miglioramento della Qualità del Prodotto**: Un altro vantaggio significativo è legato alla qualità del prodotto finale. Un macchinario che funziona in modo anomalo può produrre articoli difettosi, portando a scarti e a un uso inefficiente delle risorse. Questo è particolarmente critico in settori come quello alimentare, farmaceutico e chimico, dove la qualità del prodotto ha implicazioni dirette sulla salute dei consumatori. La capacità di rilevare e correggere le anomalie tempestivamente consente alle aziende di mantenere elevati standard qualitativi, minimizzando gli sprechi e garantendo la conformità agli standard di settore.
5. **Riduzione delle Penalità Contrattuali**: Le penalità contrattuali sono un altro aspetto critico che può essere mitigato attraverso l'uso dell'anomaly detection. Ritardi nelle consegne o il mancato rispetto degli standard possono comportare penalità significative e danneggiare la reputazione dell'azienda. L'utilizzo di un sistema di rilevazione efficiente può permettere alle aziende di rispettare le scadenze e mantenere alti standard di qualità, evitando conseguenze negative sia in termini economici che reputazionali.

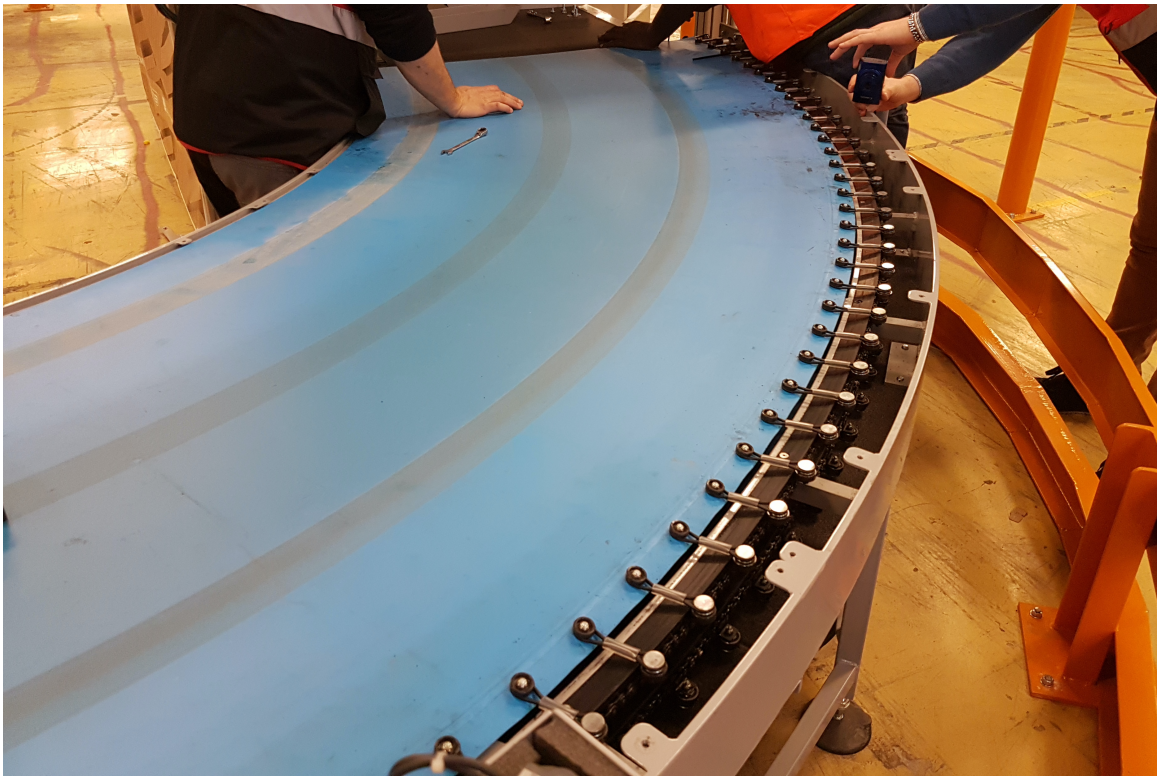
1.3 Introduzione al Caso Studio

Durante il mio tirocinio presso il CNR - STIIMA (Consiglio Nazionale delle Ricerche - Istituto di Sistemi e Tecnologie Industriali per il Manifatturiero Avanzato), ho avuto l'opportunità di partecipare a un progetto innovativo che, sebbene fosse ancora nelle sue fasi embrionali al mio arrivo, mirava a rivoluzionare il monitoraggio e la manutenzione dei nastri trasportatori utilizzati da un'azienda del bolognese. Il progetto, in collaborazione con questa azienda, è nato dalla necessità di migliorare l'efficienza operativa e la continuità produttiva, elementi cruciali per il successo nel settore manifatturiero.

I nastri trasportatori oggetto dello studio sono componenti fondamentali per il processo produttivo dell'azienda e includono diversi elementi, come cappi, cravatte e una catena di movimentazione. Tradizionalmente, il rilevamento dei guasti su questi sistemi si è sempre basato su tecniche reattive, come fermo macchina e ispezioni manuali. Tuttavia, tali approcci, oltre a essere costosi, comportano significativi tempi di inattività e spesso non riescono a individuare guasti minori che, se trascurati, possono evolversi in problemi molto più gravi. L'importanza di sviluppare un sistema più efficiente e proattivo ha portato alla decisione di esplorare tecniche di anomaly detection supportate dal machine learning. Sebbene il progetto miri a sviluppare, in futuro, un'integrazione più avanzata con un digital twin – una replica digitale del nastro trasportatore che potrebbe consentire un monitoraggio ancora più dettagliato in tempo reale – il mio contributo si è concentrato sul miglioramento degli strumenti di data analytics esistenti. In particolare, il mio lavoro ha riguardato la selezione e l'ottimizzazione delle tecniche di analisi dei dati, al fine di migliorare l'efficacia dei sistemi di monitoraggio attualmente in uso.
Questa fase del progetto è stata fondamentale per creare una base solida su cui costruire future implementazioni più avanzate. Il miglioramento dei sistemi di AD esistenti rappresenta un passo cruciale verso l'adozione di tecnologie più sofisticate e la transizione verso un sistema di manutenzione predittiva. Tale approccio consente di rilevare tempestivamente le anomalie, prevenendo guasti e fermate non pianificate, e migliorando così l'efficienza operativa complessiva.

Specifiche del Caso Studio:

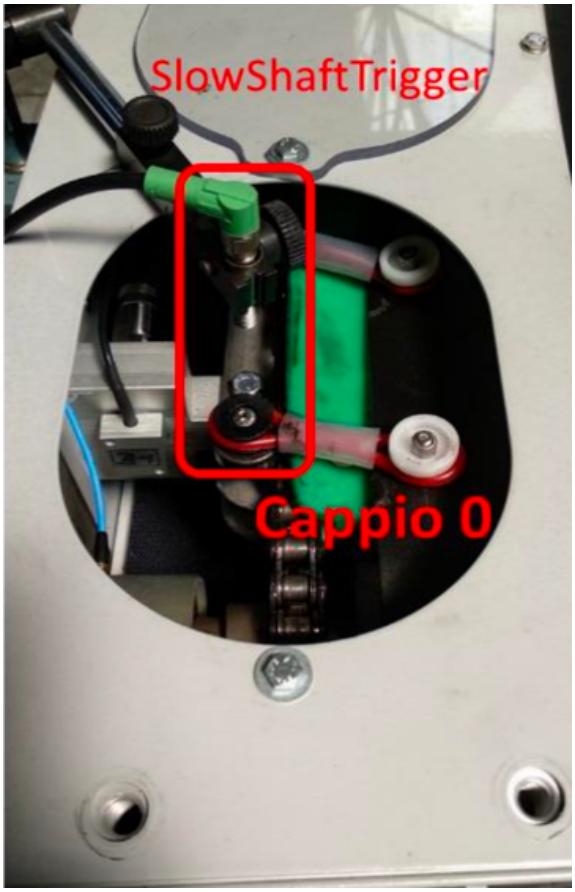
Il nastro trasportatore

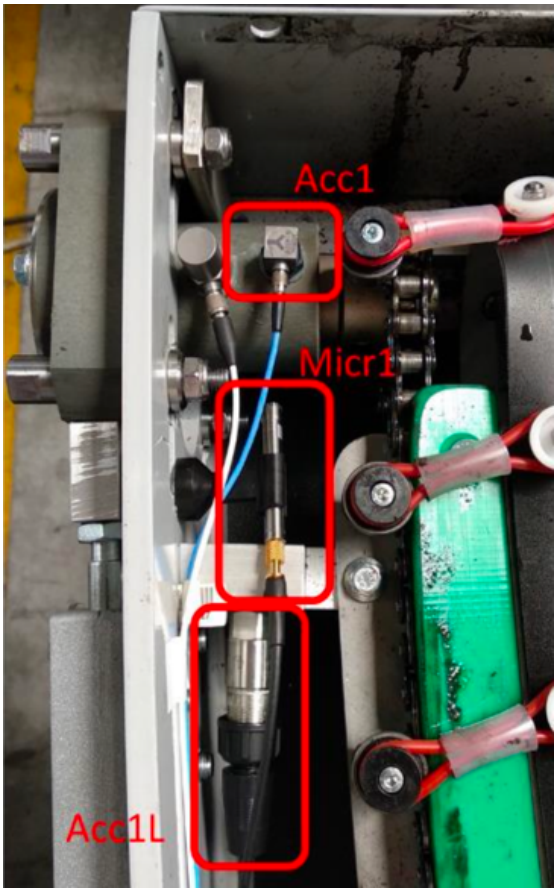


Il nastro trasportatore studiato rappresenta un elemento cruciale del sistema produttivo, essenziale per mantenere la continuità e l'efficienza operativa dell'azienda. Il sistema è costituito da 90 coppie di "bottoni", indicizzate da 0 a 89, che servono a trasmettere il movimento dalla catena al nastro. Ogni coppia di bottoni include un bottone collegato alla catena e uno al tappeto, uniti da un elastico rosso, noto come "cappio", che viene mantenuto in tensione da una cravatta, un elemento plastico semitrasparente. Questi componenti sono fondamentali per assicurare che il movimento del nastro avvenga senza interruzioni, riducendo al minimo il rischio di guasti improvvisi che potrebbero compromettere l'intera linea produttiva.

Il sistema è progettato per operare in ambienti industriali che richiedono un funzionamento continuo, ed è quindi esposto a una serie di sfide operative.

La complessità della struttura e la necessità di mantenere la tensione adeguata sui cappi e sulle cravatte sono fattori critici che possono influenzare la longevità e l'efficacia del nastro trasportatore.

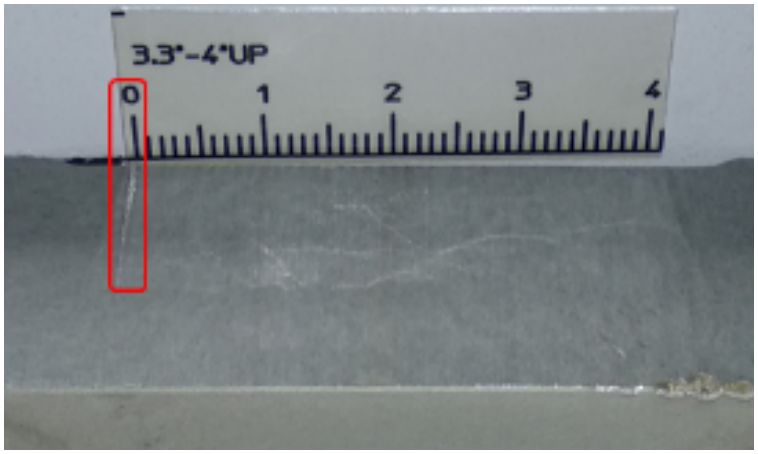




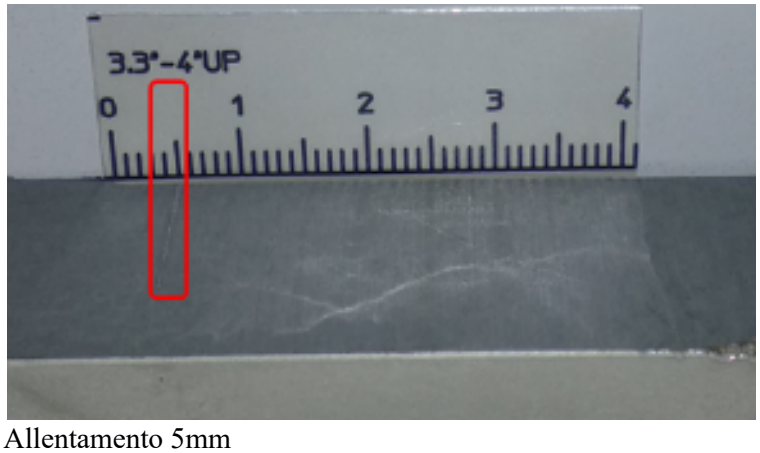
Tipologie di problemi frequenti

Durante il normale funzionamento, il nastro trasportatore può essere soggetto a due problemi principali, che devono essere monitorati attentamente per prevenire danni più gravi:

1. **Allentamento della catena**: L'allentamento della catena è una problematica ricorrente che, pur non causando un immediato fermo macchina, può compromettere significativamente le prestazioni del sistema. Un allentamento eccessivo può portare a una trasmissione inefficace del movimento, accelerando l'usura dei bottoni e del nastro stesso. Questo fenomeno, se non rilevato in tempo, può ridurre l'efficienza operativa e aumentare i costi di manutenzione a lungo termine.

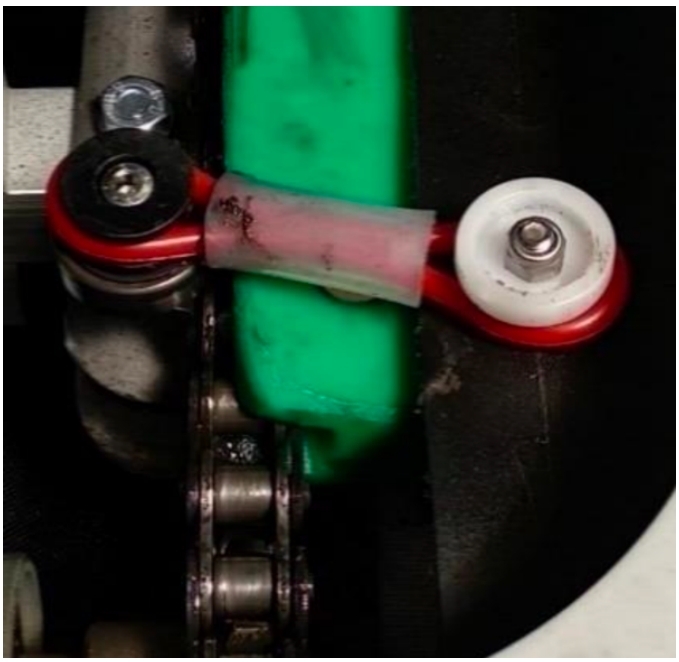


Situazione nominale



Allentamento 5mm

2. **Rottura di cappi e cravatte**: Questi componenti sono vitali per mantenere la tensione del nastro e assicurare un funzionamento regolare. La rottura o l'allentamento dei cappi o delle cravatte può causare un deterioramento delle prestazioni del nastro trasportatore, aumentando l'usura e il rischio di guasti catastrofici. La mancata rilevazione tempestiva di questi problemi può portare a interruzioni del servizio e a costose riparazioni.


Cappio (rosso) e Cravatta (trasparente)

Installazione e Disposizione dei Sensori

Per monitorare efficacemente lo stato del nastro trasportatore e prevenire i problemi sopra menzionati, sono stati installati diversi sensori in punti strategici lungo il sistema. Questi sensori sono stati disposti in cinque zone chiave, ognuna delle quali è stata selezionata per massimizzare la capacità di rilevamento delle anomalie:

- **Zona 1**: Situata all'inizio del percorso del nastro, questa zona include un accelerometro piezoelettrico triassiale da 500g, orientato lungo gli assi X (direzione del moto), Y (perpendicolare alla superficie di fissaggio), e Z

(trasversale al moto). È presente anche un accelerometro piezoelettrico monoassiale low-cost da 50g, orientato nella direzione del moto (asse X), oltre a un microfono piezoelettrico che rileva eventuali anomalie acustiche indicative di problemi meccanici.

- **Zona 2**: Questo segmento del nastro è monitorato da un accelerometro piezoelettrico monoassiale low-cost da 50g, anch'esso orientato lungo l'asse X (direzione del moto). Questa disposizione consente di rilevare variazioni nelle vibrazioni lungo la direzione principale del movimento, facilitando la diagnosi di problemi legati alla trasmissione del moto.
- **Zona 3**: Qui è installato un accelerometro piezoelettrico triassiale da 50g, configurato per rilevare le vibrazioni lungo gli assi X, Y, e Z. Oltre a questo, la zona include un accelerometro piezoelettrico monoassiale low-cost da 50g, orientato lungo l'asse X, e un accelerometro MEMS monoassiale da 500g, che fornisce una misura dettagliata delle vibrazioni lungo la stessa direzione. Questa combinazione di sensori permette di ottenere una visione completa delle forze meccaniche in gioco.
- **Zona 4**: Qui è posizionato un altro accelerometro piezoelettrico monoassiale low-cost da 50g, orientato lungo l'asse X. Questo sensore contribuisce al monitoraggio continuo delle condizioni del nastro trasportatore, rilevando potenziali disallineamenti o vibrazioni anomale che potrebbero indicare un problema imminente.
- **Zona 5**: Questa zona, situata verso la fine del percorso del nastro, è dotata di un accelerometro piezoelettrico triassiale da 500g, un accelerometro piezoelettrico monoassiale low-cost da 50g, e un accelerometro MEMS triassiale da 16g. La presenza di questi sensori consente di monitorare non solo le vibrazioni, ma anche eventuali variazioni nella forza applicata al nastro durante il suo movimento. Inoltre, in questa zona è installata una fotocellula per il rilevamento del

passaggio del primo cappio, che funge da trigger per la sincronizzazione dei dati.

- **Zona M**: In questa zona sono installate tre sonde di corrente, ciascuna collegata a una delle tre fasi del motore che aziona il nastro trasportatore. Queste sonde forniscono informazioni essenziali sul consumo di energia e sulle condizioni operative del motore, che sono indicatori indiretti della salute del nastro e dei suoi componenti meccanici.

Struttura dei Dati Raccolti

Le misurazioni sono state effettuate in condizioni controllate, variando intenzionalmente alcuni parametri operativi del nastro trasportatore per simulare guasti e monitorare la risposta del sistema. Le condizioni testate includono l'allentamento progressivo della catena (1mm, 2mm e 5mm) e la rimozione incrementale di diversi cappi e cravatte. Ogni prova è stata eseguita in modalità nominale e con variazioni specifiche delle condizioni operative.

I dati raccolti, campionati a una frequenza elevata di 51.200 Hz, garantiscono un'altissima risoluzione temporale. Ogni file di acquisizione copre un minuto e comprende segnali provenienti da 24 canali distinti, ognuno dei quali registra in totale 3.072.000 campioni. I segnali misurati includono accelerazioni, pressioni e tensioni elettriche, elementi fondamentali per la diagnosi dello stato del sistema. Alcuni di questi, come *BeckhoffSync*, usato per la sincronizzazione dei sistemi di acquisizione, saranno esclusi dall'analisi finale poiché non contribuiscono direttamente alla diagnosi delle condizioni operative del nastro. Tuttavia, segnali come *SlowShaftTrigger*, una misura binaria che registra il passaggio del primo cappio, risultano fondamentali per sincronizzare i dati di vibrazione con eventi specifici del ciclo operativo del nastro.

Questo permette di correlare le anomalie rilevate a segmenti particolari del nastro trasportatore, facilitando così l'individuazione delle cause profonde dei guasti osservati. Tale struttura dei dati è progettata per supportare l'analisi

dettagliata delle condizioni del nastro trasportatore e lo sviluppo di modelli di machine learning, in grado di rilevare tempestivamente le anomalie, migliorando l'efficienza operativa e riducendo i costi di manutenzione nel lungo termine.

Segnale	Unità di misura	Lunghezza
Micr1	Pascal	3072000
Acc1X	g	3072000
Acc1Y	g	3072000
Acc1Z	g	3072000
Acc5X	g	3072000
Acc5Y	g	3072000
Acc5Z	g	3072000
Acc1L	g	3072000
Acc3X	g	3072000
Acc3Y	g	3072000
Acc3Z	g	3072000
Acc3L	g	3072000
Acc5L	g	3072000
Acc2L	g	3072000
Acc4L	g	3072000
Mems3	Volt	3072000
Curr1	Volt	3072000
Curr2	Volt	3072000
Curr3	Volt	3072000
Mems5X	Volt	3072000
Mems5Y	Volt	3072000
Mems5Z	Volt	3072000
BeckhoffSync	Binario	3072000
SlowShaftTrigger	Binario	3072000

Lo stato dell’arte e la sfida

Al momento del mio ingresso nel progetto, il team aveva già raggiunto risultati significativi, che costituivano la base del mio lavoro. Una delle prime attività era stata una ricerca approfondita sulle prestazioni di diversi modelli di machine learning per la classificazione dei guasti. Dalle analisi comparative, Random Forest e gli alberi decisionali erano emersi come le tecniche più promettenti, in grado di gestire dati complessi e distinguere con successo tra condizioni operative nominali e vari guasti, come l'allentamento della catena, problemi ai cappi e danni alle cravatte. Un modello preliminare era già stato sviluppato e testato su un campione di dati acquisiti in ambiente reale, con risultati incoraggianti in termini di accuratezza. Tuttavia, sono emersi alcuni problemi critici. In particolare, i lunghi tempi di addestramento rappresentavano una sfida significativa. Per risolvere questo problema, si era esplorata l'opzione di sfruttare un supercalcolatore, che avrebbe consentito di elaborare un volume maggiore di dati in tempi ridotti. Tuttavia, questa soluzione comportava delle difficoltà, come l'accesso limitato al supercalcolatore, la gestione dei permessi di utilizzo e limitazioni in scalabilità del progetto stesso.

Come alternativa, si era ipotizzato di modificare la struttura dei dati, riducendo la durata di ogni file da 60 a 30 secondi. Inoltre, per affrontare ulteriormente la questione dei tempi di elaborazione, il team aveva sperimentato il sottocampionamento dei dati utilizzando il software DIADEM, passando da una frequenza di campionamento di 51,2 kHz a 10 kHz. Questa strategia mirava a ridurre la dimensione dei file, consentendo una gestione più efficiente dei dati, senza compromettere significativamente la qualità delle misurazioni.

Nonostante questi tentativi di ottimizzazione, il modello presentava ancora difficoltà nella discriminazione accurata delle classi di errore, in particolare nella distinzione tra guasti ai cappi e condizioni nominali. Gli errori di classificazione erano in parte attribuibili all'impostazione di soglie decisionali settate di default a 0.5, che portavano a classificazioni errate. Per migliorare

ulteriormente il modello, era necessario affinare queste soglie e bilanciare meglio il dataset, includendo una maggiore varietà di dati relativi ai guasti.
Il mio lavoro si è quindi concentrato sul trovare la strada migliore per incrementare le prestazioni del modello, ottimizzando i tempi di addestramento e perfezionando la capacità del sistema di distinguere accuratamente tra le diverse tipologie di guasti. In particolare, ho lavorato sulla parte di preprocessing, per ridurre le tempistiche di addestramento e verificare l'impatto di tali cambiamenti sulle prestazioni del modello.
Al termine del mio progetto, i miei risultati verranno integrati con quelli ottenuti dal team per sviluppare il modello finale, che sarà anche oggetto di un paper scientifico in collaborazione con il CNR.

2. BACKGROUND

Questo progetto ha richiesto uno studio approfondito e l'utilizzo delle principali tecnologie esistenti nel campo dell'analisi dei dati. In questo capitolo verranno analizzate e contestualizzate tali tecnologie, al fine di fornire le basi teoriche necessarie per comprendere appieno il lavoro svolto.

2.1 Modelli utilizzati

Random Forest

Il metodo Random Forest rappresenta un avanzato approccio di ensemble learning, una tecnica che aggrega le previsioni di più modelli di machine learning per produrre un risultato migliore rispetto a quello ottenuto da un singolo modello. Nello specifico, Random Forest costruisce un insieme di alberi decisionali indipendenti durante la fase di addestramento, combinando poi i loro risultati per ottenere previsioni più accurate e stabili.

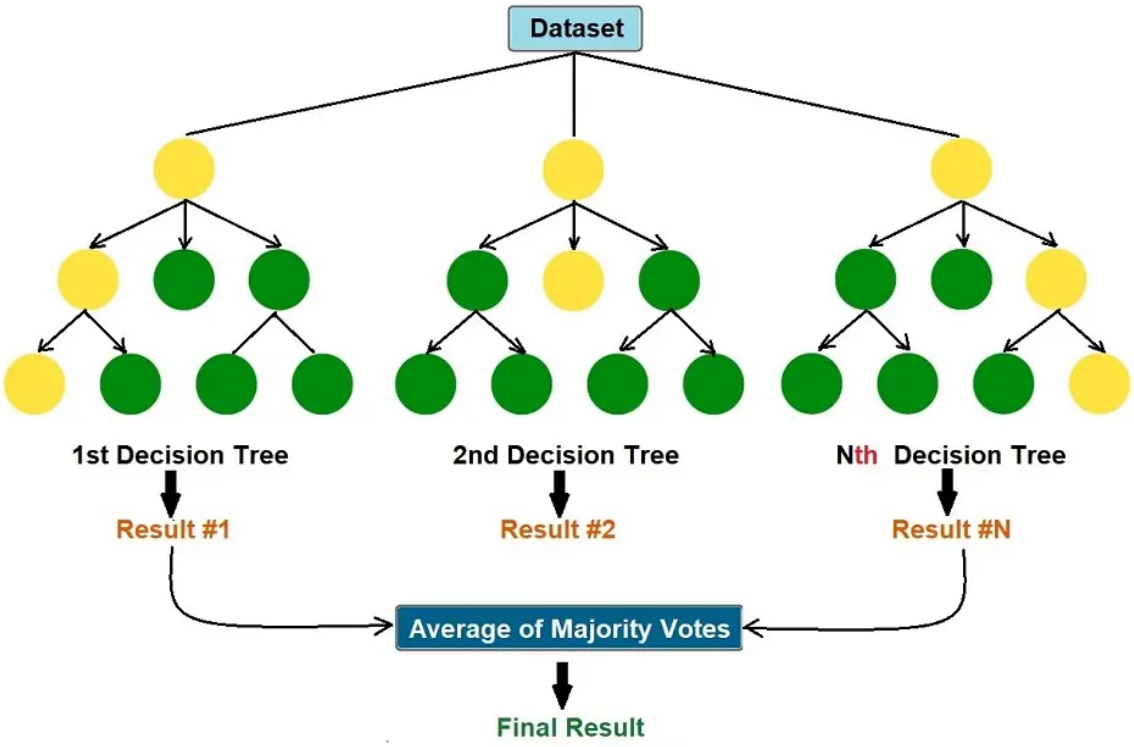
Il processo inizia con la generazione di numerosi alberi decisionali, ciascuno addestrato su un campione di dati ottenuto attraverso il metodo di bootstrap. Questo metodo prevede che ognuno di questi sia addestrato su un sottoinsieme leggermente diverso dei dati, aumentando così la diversità tra gli alberi. Durante la costruzione di ogni albero, si applica anche la selezione casuale degli attributi di split, il che consente di esplorare diverse combinazioni di attributi e di ridurre la correlazione tra gli alberi. Una volta addestrati, i risultati prodotti da tutti gli alberi vengono aggregati per ottenere la previsione finale del modello di Random Forest. Nel contesto della classificazione, l'aggregazione avviene tramite un sistema di votazione a maggioranza: ogni albero esprime un voto per una classe e la classe che riceve il maggior numero di voti diventa la previsione del modello. In ambito di regressione, la previsione finale è calcolata come la media delle previsioni fornite da tutti gli alberi.

Tra i parametri più rilevanti che influenzano le prestazioni e il comportamento

di una Random Forest vi sono il numero di alberi nella foresta (n_estimators), la profondità massima degli alberi (max_depth) e il numero minimo di campioni richiesti per dividere un nodo (min_samples_split). Il parametro n_estimators determina quanti alberi decisionali verranno costruiti nel modello: un numero maggiore di alberi può migliorare la precisione del modello riducendo l'overfitting, ma aumenta il tempo di addestramento e il consumo di risorse computazionali. La *max_depth* limita la profondità degli alberi, controllando così la complessità del modello e aiutando a prevenire l'overfitting. Se il parametro viene impostato su un valore troppo alto, gli alberi possono diventare troppo complessi e specifici per il dataset di addestramento, mentre valori troppo bassi possono portare a un underfitting, riducendo la capacità del modello di catturare relazioni complesse. Il parametro *min_samples_split* indica il numero minimo di campioni necessari per considerare la divisione di un nodo: impostando questo valore troppo basso si rischia di creare molti nodi poco significativi, mentre un valore troppo alto può ridurre la capacità dell'albero di adattarsi ai dati di addestramento. Altri parametri importanti includono il numero minimo di campioni per nodo foglia (min_samples_leaf), che impedisce la creazione di nodi con pochi campioni e migliora la generalizzazione, e il numero di caratteristiche da considerare per la suddivisione (max_features), che influisce sulla diversità degli alberi. Infine, il parametro bootstrap determina se utilizzare campionamenti con o senza ripetizione per la creazione degli alberi: il campionamento con ripetizione (default) aggiunge variabilità al modello, aumentando la robustezza della predizione.

La valutazione delle prestazioni del modello Random Forest può essere eseguita

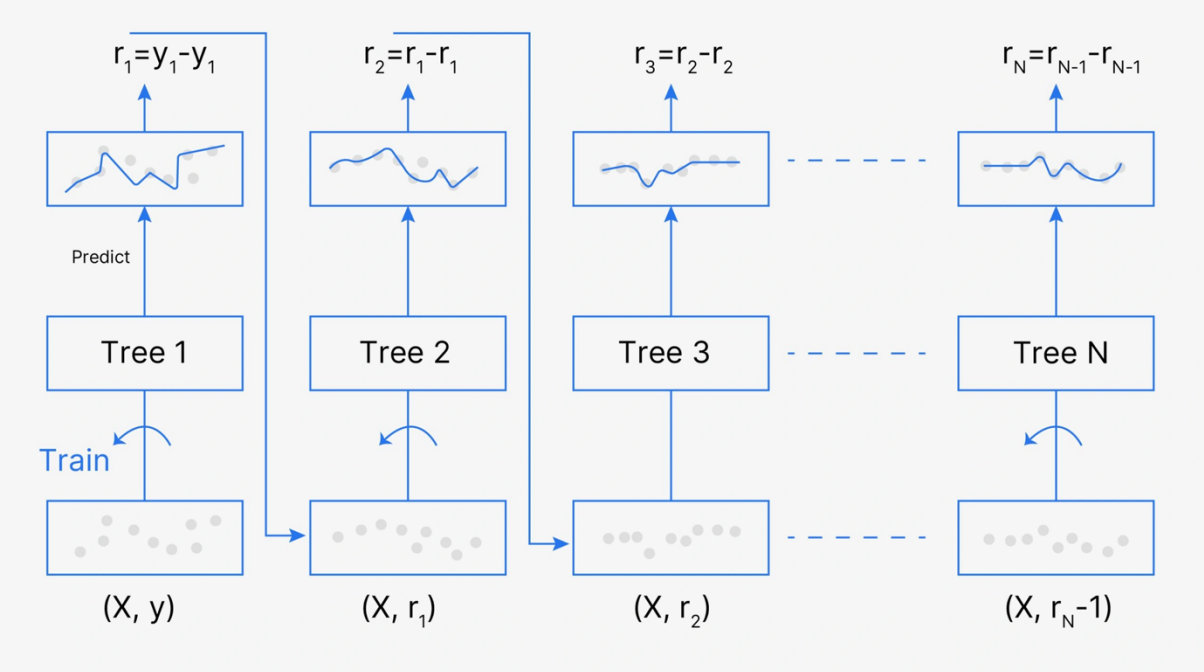
utilizzando diverse metriche, tra cui la matrice di confusione, l'accuracy, il precision score, il recall e l'F1-score.



Gradient Boosted Trees

Il metodo dei Gradient Boosted Trees (GBT) è una potente tecnica di machine learning basata su un approccio sequenziale per la costruzione di alberi decisionali. A differenza di algoritmi come Random Forest, che addestra alberi in parallelo, il boosting costruisce ogni albero in sequenza con l'obiettivo di correggere gli errori commessi dagli alberi precedenti. Il principio fondamentale del GBT è minimizzare una funzione di perdita, un parametro che quantifica quanto le previsioni del modello si discostano dai valori reali. Ogni nuovo albero viene costruito per migliorare la funzione di perdita residua, correggendo gli errori degli alberi precedenti. Il processo inizia con un singolo albero decisionale e, a ogni iterazione, viene costruito un nuovo albero addestrato per focalizzarsi sugli errori compiuti dal modello fino a quel punto. Questo è

possibile grazie al calcolo del "gradiente" della funzione di perdita: il gradiente indica la direzione e l'entità delle modifiche da apportare ai parametri del modello per ridurre la funzione di perdita. Il GBT applica il principio della discesa del gradiente, che aggiorna iterativamente il modello in modo che le previsioni migliorino progressivamente. A ogni iterazione, i pesi dei campioni mal classificati vengono aumentati, consentendo al modello di concentrare maggiore attenzione su questi esempi problematici. Questo continuo aggiornamento del modello permette di correggere gli errori residui e migliorare le previsioni globali. Il processo continua fino a quando viene raggiunto un numero prefissato di alberi o finché la funzione di perdita non cessa di migliorare. La funzione di perdita è l'elemento centrale del GBT: per i problemi di classificazione, la funzione di perdita più comunemente utilizzata è la log-loss (devianza), mentre per i problemi di regressione si usa solitamente la perdita quadratica media (Mean Squared Error, MSE). In ogni caso, l'obiettivo finale del GBT è ridurre progressivamente la funzione di perdita ad ogni passaggio, migliorando le previsioni del modello fino a ottenere risultati ottimali.



Nel corso di questa tesi mi riferirò ai risultati dei modelli di GBT con la nomenclatura XGB in modo coerente a quanto fatto nel tirocinio, stando ad indicare che i modelli hanno utilizzato la libreria xgboost.

XGBoost (Extreme Gradient Boosting)

XGBoost è una libreria open-source progettata per ottimizzare l'implementazione dei Gradient Boosted Trees, rendendola estremamente efficiente e scalabile. XGBoost si distingue per una serie di migliorie rispetto ai metodi tradizionali di boosting, tra cui la gestione ottimizzata della memoria e l'implementazione parallela. Questa libreria è ampiamente utilizzata nelle competizioni di machine learning e in applicazioni reali che richiedono alte performance, come la classificazione e la regressione su dataset di grandi dimensioni. Uno dei principali vantaggi di XGBoost rispetto al GBT tradizionale è l'integrazione di tecniche di regolarizzazione L1 (lasso) e L2 (ridge), che aiutano a ridurre l'overfitting e migliorare la generalizzazione del modello. Grazie a queste tecniche, XGBoost gestisce meglio i dataset rumorosi e ad alta dimensionalità, fornendo previsioni più stabili e accurate. Inoltre, XGBoost ottimizza l'efficienza computazionale utilizzando il calcolo parallelo, il che permette di sfruttare al massimo le risorse hardware disponibili e di ridurre notevolmente i tempi di addestramento, soprattutto su dataset di grandi dimensioni. Un'altra caratteristica distintiva è la gestione nativa dei dati mancanti: XGBoost considera i dati mancanti come una possibile categoria durante la suddivisione degli alberi, migliorando così la robustezza del modello in presenza di dataset incompleti. XGBoost introduce anche una tecnica avanzata di pruning, basata sulla riduzione della loss (Gain), che elimina automaticamente i rami non necessari durante la costruzione degli alberi, mantenendo il modello compatto ed efficiente. Questo consente di mantenere modelli complessi in grado di adattarsi a situazioni reali senza sovraccaricare il sistema con calcoli superflui. I principali parametri che influenzano il

comportamento di XGBoost includono il numero di alberi (n_estimators), la profondità massima degli alberi (max_depth), la learning rate, e i parametri di regolarizzazione (lambda per L2 e alpha per L1). L'ottimizzazione di questi parametri è essenziale per trovare il giusto equilibrio tra bias e varianza, garantendo che il modello si adatti bene ai dati di addestramento senza diventare troppo complesso. Le performance di XGBoost sono solitamente valutate utilizzando metriche come l'accuracy, il precision score, il recall, e l'F1-score. Tuttavia, per applicazioni in cui è importante valutare la capacità del modello di discriminare tra classi, la curva ROC e l'AUC (Area Under the Curve) sono metriche particolarmente efficaci. Un AUC vicino a 1 rappresenta un'ottima capacità discriminativa, indicando che il modello è altamente performante nel separare le classi target.

In Letteratura

Nella letteratura scientifica, sia Random Forest (RF) che XGBoost (XGB) sono frequentemente citati come metodi molto efficaci per la rilevazione delle anomalie in contesti industriali complessi, grazie alle loro capacità di gestione di dataset eterogenei e complessi. Breiman (2001) ha introdotto Random Forest, un metodo di ensemble learning basato su alberi decisionali, noto per la sua robustezza contro l'overfitting e per la sua capacità di gestire dataset con elevato rumore. Questo modello si distingue per la sua capacità di generalizzare bene anche in presenza di dati rumorosi o non lineari. È ampiamente utilizzato nelle applicazioni di manutenzione predittiva per rilevare guasti o anomalie nei processi produttivi. Chen e Guestrin (2016) hanno sviluppato XGBoost, che si basa su tecniche di boosting. XGBoost offre eccellenti capacità di gestione dei dati ad alta dimensionalità e la sua efficienza computazionale lo rende una scelta preferita per scenari industriali che richiedono una rapida identificazione delle anomalie. Grazie alla sua capacità di utilizzare la regolarizzazione L1 e L2, XGBoost riduce significativamente il rischio di overfitting, migliorando

così la precisione delle previsioni. Diversi studi, come quello di Lu et al. (2024), hanno confermato l'efficacia di XGBoost nel rilevamento in tempo reale delle anomalie, specialmente in scenari industriali dove la velocità e la precisione sono essenziali per la manutenzione predittiva.
Uno studio comparativo di Naghibi et al. (2020) ha messo in evidenza che, sebbene entrambi i modelli offrano elevate prestazioni, XGBoost tende a superare Random Forest in termini di velocità di addestramento e capacità di gestire dataset con elevate variazioni, rendendolo una scelta ideale in contesti dove i dati possono essere altamente variabili

2.2 Strumenti per la Data Analytics

Python

Python è un linguaggio di programmazione ad alto livello che ha assunto un ruolo centrale nella data analytics grazie alla sua semplicità, versatilità e alla vasta gamma di librerie disponibili. La sua leggibilità e la facilità di implementazione di soluzioni complesse lo rendono ideale per l'analisi dei dati.

Le principali librerie utilizzate in questo ambito sono:

- **Pandas**: Fornisce strutture dati avanzate come *Series* e *DataFrame* per la manipolazione efficiente di dati strutturati. Facilita operazioni essenziali come la pulizia, l'aggregazione e la trasformazione dei dati.
- **NumPy**: Costituisce il fondamento del calcolo scientifico in Python, offrendo supporto per array multidimensionali e funzioni matematiche ad alte prestazioni. È fondamentale per l'elaborazione numerica e spesso utilizzato in combinazione con Pandas.
- **Scikit-learn**: È la libreria di riferimento per l'implementazione di algoritmi di machine learning. Include strumenti per l'apprendimento supervisionato e non supervisionato, oltre a metodi per la selezione degli

iperparametri, la validazione incrociata e la valutazione delle prestazioni dei modelli.

- **Matplotlib**: Principale libreria per la visualizzazione dei dati, consente di creare grafici e diagrammi di alta qualità, essenziali per interpretare visivamente i risultati delle analisi e dei modelli di machine learning. Insieme a **Seaborn**, permette rappresentazioni grafiche avanzate.

Queste librerie costituiscono una pipeline potente per l'analisi dei dati: Pandas e NumPy gestiscono e trasformano i dati, Scikit-learn offre strumenti per costruire e valutare modelli predittivi, mentre Matplotlib visualizza i risultati. Questa sinergia rende Python una delle soluzioni più complete e versatili per la data analytics e il machine learning.

Jupyter Notebook

Jupyter Notebook è un ambiente di sviluppo integrato (IDE) essenziale nella scienza dei dati e nella data analytics, che consente di combinare codice eseguibile, testo descrittivo, visualizzazioni grafiche e formule matematiche in un unico documento interattivo. Originariamente parte del progetto IPython, Jupyter si è affermato grazie alla sua flessibilità e alla capacità di facilitare l'esplorazione e la visualizzazione dei dati. La natura interattiva di Jupyter Notebook permette l'esecuzione incrementale del codice con visualizzazione immediata dei risultati, supportando un'iterazione rapida fondamentale per l'analisi di dataset complessi e lo sviluppo di modelli di machine learning. L'integrazione di visualizzazioni grafiche direttamente nel notebook facilita la comprensione dei dati e delle loro tendenze. Jupyter Notebook supporta numerosi linguaggi di programmazione, tra cui i famosissimi Python, R e Julia. In particolare, Python è predominante grazie alla vasta gamma di librerie specifiche per la data analytics, come Pandas per la manipolazione dei dati, NumPy per il calcolo numerico, Matplotlib e Seaborn per la visualizzazione, e Scikit-learn per l'implementazione di algoritmi di machine learning. La capacità

di documentare il processo analitico attraverso testo descrittivo e spiegazioni dettagliate rende l'analisi trasparente e riproducibile, un aspetto cruciale nella ricerca scientifica, soprattutto quando si lavora in team. La compatibilità con sistemi di controllo di versione come Git e l'integrazione con piattaforme cloud come Google Colab e JupyterHub estendono le potenzialità di Jupyter Notebook, consentendo l'accesso remoto e la collaborazione in tempo reale.

2.3 Valutazione delle Prestazioni del Modello

Nel contesto della classificazione degli errori, è fondamentale valutare accuratamente le prestazioni dei modelli di machine learning per garantire che essi siano non solo accurati, ma anche affidabili e robusti. Python, attraverso le sue librerie come Scikit-learn, offre un ampio spettro di metriche e strumenti per questa valutazione.

• **Accuracy**

Questa metrica misura la proporzione di predizioni corrette rispetto al totale delle predizioni effettuate.
È calcolata come il rapporto tra la somma dei veri positivi (TP) e dei veri negativi (TN) rispetto al totale di predizioni:

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

L'accuracy è utile in molteplici scenari, ma può risultare fuorviante in presenza di classi molto sbilanciate, dove una classe potrebbe dominare sull'altra.

• **Precision**

La precisione valuta la capacità del modello di evitare falsi positivi, ovvero quanto tra le predizioni positive sia realmente corretto:

$$Precision = \frac{TP}{TP + FP}$$

La precision è cruciale in contesti dove è importante minimizzare i falsi positivi, come nella rilevazione di frodi o nella diagnosi medica.

• **Recall (o Sensitivity)**

Il recall misura la capacità del modello di identificare correttamente tutti i veri positivi:

$$Recall = \frac{TP}{TP + FN}$$

Un alto recall è essenziale in situazioni in cui è importante catturare ogni possibile caso positivo, come nella rilevazione di anomalie o in sistemi di sicurezza.

• **F1 Score**

L'F1 score è la media armonica di precision e recall, fornendo un bilanciamento tra queste due metriche, particolarmente utile in contesti con classi sbilanciate:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

L'F1 score è elevato solo se sia precision che recall sono alte, rendendolo una metrica robusta per valutare modelli in scenari complessi.

Per ottenere una comprensione ancora più approfondita delle prestazioni del modello, si utilizzano strumenti grafici come la Confusion Matrix e la ROC Curve. La Confusion Matrix è una rappresentazione tabellare che mostra il numero di predizioni corrette e incorrette per ciascuna classe, includendo i conteggi di veri positivi, veri negativi, falsi positivi e falsi negativi. Questo strumento permette di avere una panoramica completa delle prestazioni del modello, evidenziando specifici pattern di errore che possono essere critici per migliorare il modello.

Parallelamente, la ROC Curve (Receiver Operating Characteristic) offre una rappresentazione visiva del trade-off tra il tasso di veri positivi (recall) e il tasso di falsi positivi a diverse soglie di classificazione. La curva ROC è fondamentale per capire come il modello bilancia queste due metriche in base alla soglia scelta. L'area sotto la curva (AUC) quantifica la performance complessiva del modello su tutte le soglie possibili, con un AUC vicino a 1 che indica un'eccellente capacità discriminativa.

Questi strumenti grafici, derivati dalle metriche di precision e recall, forniscono all'analista un output visivo che facilita l'interpretazione e la comparazione delle performance dei modelli. La Confusion Matrix permette di visualizzare in modo dettagliato dove il modello commette errori, mentre la ROC Curve e l'AUC consentono di valutare la capacità del modello di distinguere tra le classi positive e negative, aiutando a scegliere la soglia di decisione più appropriata per l'applicazione specifica.

3. PREPROCESSING DEI DATI

3.1 Importanza del Preprocessing nell'Analisi Dati

Il preprocessing dei dati è una fase cruciale in qualsiasi progetto di analisi dati, specialmente nel contesto del machine learning (ML) e della classificazione. Il successo di un modello di machine learning dipende in gran parte dalla qualità dei dati forniti durante la fase di addestramento, e il preprocessing è essenziale per garantire che i dati siano adeguatamente preparati per l'analisi.

Questo include una serie di tecniche e procedure volte a trasformare i dati grezzi in un formato più adatto per i modelli che li utilizzano: tali tecniche possono includere la pulizia dei dati, la gestione dei valori mancanti, la normalizzazione o standardizzazione delle variabili, la riduzione della dimensionalità e la selezione delle features. La letteratura scientifica dimostra che il preprocessing può avere un impatto significativo sulle prestazioni. Ad esempio, Alam e Yao (2019) hanno evidenziato come diverse tecniche possano influenzare l'accuratezza degli algoritmi di classificazione, dimostrando che un'attenta selezione è indispensabile per migliorare le prestazioni del modello. Uno dei principali vantaggi del suo utilizzo è la capacità di ridurre la complessità dei dati, il che aiuta a migliorare l'efficienza computazionale e a prevenire l'overfitting. Un dataset ben preprocessato riduce il rumore e le ridondanze, rendendo i pattern più evidenti e i modelli più capaci di generalizzare sui dati non visti. Tuttavia, il preprocessing non è privo di sfide: un uso eccessivo di tecniche di riduzione della dimensionalità o di selezione delle caratteristiche può portare alla perdita di informazioni rilevanti, compromettendo la capacità del modello di fare previsioni accurate. Inoltre, alcune tecniche richiedono scelte soggettive da parte dell'analista, come la decisione di come trattare i valori mancanti, che possono introdurre bias nel modello.

Nella fase di classificazione, l'autocorrelazione, la normalizzazione, e il binning sono particolarmente rilevanti. Queste tecniche non solo aiutano a strutturare i

dati in modo più efficace ma, come sottolineato dallo studio di Lu et al. (2024), possono anche migliorare notevolmente le prestazioni dei modelli in scenari complessi, dove i dati possono presentare elevati livelli di rumore e variabilità.

3.2 La Struttura dei Dati

Il dataset utilizzato come base per l'analisi in questo progetto è stato ottenuto da file in formato TDMS (Technical Data Management Streaming), un formato comunemente impiegato in contesti industriali per l'acquisizione di dati grezzi ad alta frequenza. Questi file TDMS contenevano dati provenienti da sensori installati su un nastro trasportatore, registrati a una frequenza di campionamento iniziale di 51.2 kHz. Tuttavia, per ragioni di efficienza computazionale e considerando la necessità di ridurre la complessità del dataset, è stato deciso di eseguire un sottocampionamento dei dati a 10 kHz utilizzando il software DIADEM. Questo processo di riduzione della dimensionalità è stato supportato da ricerche precedenti che hanno dimostrato come la riduzione della frequenza di campionamento da 51.2 kHz a 10 kHz non comporti una perdita significativa di informazioni rilevanti per l'analisi delle anomalie nel contesto considerato.
Il preprocessing iniziale eseguito tramite DIADEM ha incluso la pulizia dei dati, che ha coinvolto la rimozione degli outlier e la gestione dei valori nulli (zeri), garantendo così che i dati utilizzati per le analisi successive fossero privi di errori grossolani e pronti per il processo di binning e modellazione. Questa fase preliminare di pulizia è essenziale per assicurare che i modelli di machine learning possano essere addestrati su dati di alta qualità, riducendo il rischio di distorsioni dovute a valori anomali o errati.
Il dataset finale ottenuto da questo processo rappresenta un sottoinsieme ben curato dei dati originali, ottimizzato per bilanciare la complessità computazionale con la necessità di mantenere un elevato livello di dettaglio nelle analisi. Questo dataset è stato successivamente utilizzato per identificare

periodicità nei segnali, eseguire il binning e addestrare modelli di machine learning per la rilevazione delle anomalie.

3.3 Ricerca della dimensione dei bin

Identificazione delle periodicità e funzione di Autocorrelazione

L'identificazione delle periodicità nei segnali è un passaggio critico nel preprocessing dei dati, specialmente in contesti industriali dove la qualità e la precisione dei dati sono essenziali per l'efficacia dei modelli di machine learning. Una tecnica fondamentale utilizzata per questo scopo è l'autocorrelazione, che consente di individuare la somiglianza di un segnale con sé stesso a diversi intervalli temporali (lag), rivelando eventuali periodicità intrinseche nel segnale.

L'autocorrelazione, infatti, è una funzione matematica che misura la correlazione tra valori di una serie temporale e i loro stessi valori traslati nel tempo.

Questa funzione è definita come:

$$R(\tau) = \frac{1}{N - \tau} \sum_{t=1}^{N-\tau} x(t) * x(t + \tau)$$

Dove:

- **R(τ)** è il valore dell'autocorrelazione al lag τ,
- **x(t)** è il valore della serie temporale al tempo t,
- **τ** è il lag,
- **N** è la lunghezza totale della serie temporale.

La funzione di autocorrelazione fornisce un picco significativo quando il segnale è più simile a sé stesso, indicando la presenza di periodicità. Nel caso di dati provenienti da sensori industriali, identificare queste periodicità è cruciale

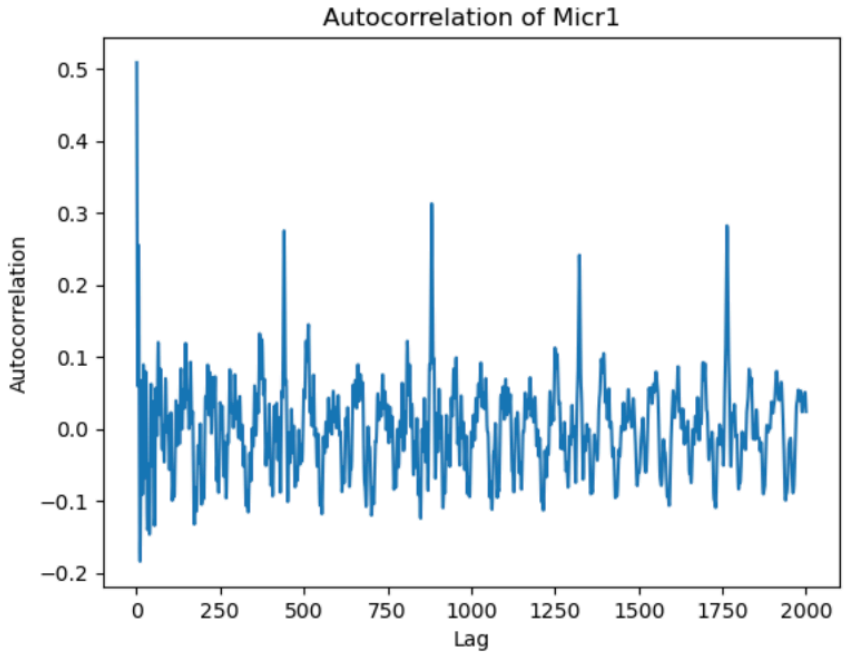
per stabilire il bin size ottimale, ossia l'intervallo temporale su cui aggregare i dati per le analisi successive.

Implementazione della Funzione di Autocorrelazione

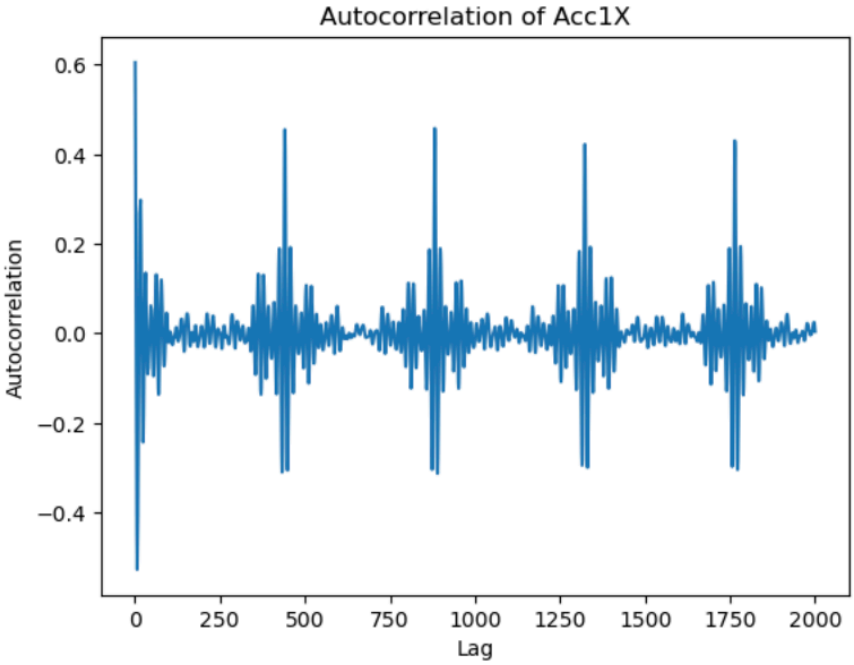
L'autocorrelazione della serie temporale viene calcolata tramite l'utilizzo della funzione integrata pandas.Series.autocorr all'interno della libreria Pandas, ed il codice utilizzato per stampare a video le periodicità è il seguente:

```
def plot_autocorrelation(file_path, column_name, max_lag):
    try:
        # Load the TDMS file
        tdms_file = TdmsFile.read(file_path)

        # Convert the TDMS file to a DataFrame
        df = tdms_file.as_dataframe()

        # Clean and rename the columns
        df = createPandasDF(df)

        df.head()

        # Extract the specified column
        series = df[column_name]

        # Calculate and plot the autocorrelation for the specified column
        autocorr_values = [series.autocorr(lag=i) for i in range(1, max_lag + 1)]  # Exclude lag 0
        plt.plot(range(1, max_lag + 1), autocorr_values)
        plt.title(f'Autocorrelation of {column_name}')
        plt.xlabel('Lag')
        plt.ylabel('Autocorrelation')
        plt.show()

        # Find the lag corresponding to the maximum peak starting from lag 200
        lag_max_peak = np.argmax(autocorr_values[200:600]) + 200
        max_peak_value = autocorr_values[lag_max_peak - 1]
        print(f"The maximum peak between 200 and 600 occurs at lag {lag_max_peak} with value {max_peak_value}")

        lag_max_peak = np.argmax(autocorr_values[600:1200]) + 600
        max_peak_value = autocorr_values[lag_max_peak - 1]
        print(f"The maximum peak between 600 and 1200 occurs at lag {lag_max_peak} with value {max_peak_value}")

        lag_max_peak = np.argmax(autocorr_values[1200:1500]) + 1200
        max_peak_value = autocorr_values[lag_max_peak - 1]
        print(f"The maximum peak between 1200 and 1500 occurs at lag {lag_max_peak} with value {max_peak_value}")

        lag_max_peak = np.argmax(autocorr_values[1500:1900]) + 1500
        max_peak_value = autocorr_values[lag_max_peak - 1]
        print(f"The maximum peak between 1500 and 1900 occurs at lag {lag_max_peak} with value {max_peak_value}")

    except Exception as e:
        print(f"An error occurred: {e}")


file_path = 'data_Prova40_0033_nominale_sampled.tdms'
plot_autocorrelation(file_path, column_name='Micr1', max_lag=2000)
```

Questo codice permette di evidenziare le periodicità presenti nei dati grezzi, ottenuti da file TDMS, esplicitando il lag ottimale che rappresenta la periodicità dominante.




```
The maximum peak between 200 and 600 occurs at lag 440 with value 0.24406696029174715
The maximum peak between 600 and 1200 occurs at lag 881 with value 0.27894003534970213
The maximum peak between 1200 and 1500 occurs at lag 1322 with value 0.22264505448718003
The maximum peak between 1500 and 1900 occurs at lag 1763 with value 0.2533792149668315
```




```
The maximum peak between 200 and 600 occurs at lag 440 with value 0.4035168953150327
The maximum peak between 600 and 1200 occurs at lag 881 with value 0.4055791765715413
The maximum peak between 1200 and 1500 occurs at lag 1322 with value 0.38490019353293753
The maximum peak between 1500 and 1900 occurs at lag 1763 with value 0.37089486388989745
```

La figura mostra il risultato dell’analisi dell'autocorrelazione, il quale rivela che la maggior parte dei segnali mostrava un picco significativo a multipli di 440 campioni. Questo valore è stato quindi scelto come bin size, e rappresenta un buon compromesso tra la conservazione dell'informazione e l'evitare una compressione eccessiva dei dati.

3.4 Costruzione del DataFrame di addestramento

Una volta stabilita la dimensione ottimale, è stata implementata la funzione di binning per costruire un DataFrame aggregato, utilizzando le metriche di media, varianza e skewness per ciascun intervallo.
Le metriche utilizzate sono fondamentali per rappresentare efficacemente le caratteristiche principali dei dati, consentendo al modello di machine learning di apprendere dalle proprietà statistiche essenziali del segnale.

- **Media (np.mean)**: La media è ampiamente utilizzata nella letteratura per la sua capacità di stabilizzare le variazioni dei dati, rendendo i modelli più resistenti al rumore e alle anomalie temporanee. È cruciale perché rappresenta la tendenza centrale di ciascun bin, fornendo un valore medio che sintetizza le fluttuazioni del segnale.
- **Varianza (np.var)**: La varianza misura la dispersione dei dati all'interno di ogni bin, catturando la variabilità intrinseca del segnale. Questa metrica è essenziale per identificare periodi di instabilità o cambiamenti nel comportamento del sistema. Studi dimostrano che la varianza è un indicatore chiave per rilevare anomalie e transizioni di stato nei segnali industriali, rendendola una scelta robusta per l'analisi dei dati nel contesto della manutenzione predittiva e della diagnostica industriale.
- **Skewness (scipy.stats.skew):** La skewness, o asimmetria, indica la distribuzione dei dati all'interno del bin, permettendo di identificare eventuali deviazioni dalla normalità. Una distribuzione asimmetrica può

segnalare la presenza di fenomeni anomali o di eventi rari, che possono essere critici per la classificazione e il rilevamento delle anomalie. La skewness è frequentemente utilizzata per caratterizzare la forma delle distribuzioni dei dati e per migliorare le performance dei modelli predittivi, specialmente in presenza di dati con distribuzioni non simmetriche.

Queste metriche sono ampiamente supportate dalla letteratura scientifica come strumenti efficaci per sintetizzare le informazioni contenute in segnali complessi, riducendo al contempo la dimensionalità del dataset senza perdere significativi dettagli informativi. Ad esempio, diversi studi hanno dimostrato che la combinazione di media, varianza e skewness fornisce una rappresentazione compatta ma informativa dei dati, migliorando la capacità dei modelli di machine learning di generalizzare su nuovi dati e di rilevare con precisione le anomalie .

Il codice per implementare il binning è il seguente:

```python
def toBinned(df):

    bin_size = 440

    # Inizializza un dizionario vuoto per memorizzare i risultati del binning per ogni colonna
    binned_data = {}

    # Itera su tutte le colonne da binnare
    for col in df.columns:
        # Esegui il binning raggruppando per bin_size
        binned_col = df[col].groupby(df.index // bin_size)

        # Calcola le statistiche per ogni bin
        binned_stats = binned_col.agg(['mean', 'skew', 'var'])

        # Rinomina le colonne risultanti in base alla colonna e alla metrica utilizzata
        binned_stats.columns = [f'{col}_{stat}_binned' for stat in ['mean', 'skew', 'var']]

        # Aggiungi i risultati al dizionario
        binned_data[col] = binned_stats

    # Crea un dataframe completo con tutte le colonne binnate
    binned_df = pd.concat(binned_data.values(), axis=1)

    return binned_df
```

Sebbene il processo di binning riduca notevolmente la dimensione del dataset, la riduzione non avviene in maniera diretta e proporzionale. Poiché si utilizzano tre metriche (media, varianza e skewness) per ogni sensore, si aggiungono nuove colonne al DataFrame. Nello specifico, per ogni sensore vengono generate tre colonne che rappresentano le tre metriche, il che porta il numero totale di colonne a 66 (22 segnali originali * 3 metriche), a cui si aggiunge una colonna per segnalare la presenza di errori. Pertanto, anche se il numero di righe del dataset viene ridotto con il binning, il numero di colonne aumenta per includere tutte le nuove metriche.

```
In [23]: concatenated_DF
```

Out[23]:

	Micr1_mean_binned	Micr1_skew_binned	Micr1_var_binned	Acc1X_mean_binned
0	-0.004816	0.051653	4.143655	0.056115
1	0.057922	0.221297	3.347931	0.054229
2	0.061762	-0.011147	3.545638	0.054216
3	0.070155	-0.083714	4.332263	0.056790
4	0.092257	-0.081737	3.935596	0.057864
...	...	...	...	...
49580	-0.120196	-0.013659	3.546483	0.045707
49581	-0.204270	-0.078290	3.663386	0.047415
49582	0.097258	-0.005709	3.475474	0.046167
49583	0.205695	0.134289	3.781502	0.045732
49584	-0.168962	-0.255723	4.431289	0.050094

49585 rows × 67 columns

4. PROGETTAZIONE E OTTIMIZZAZIONE DEL MODELLO

4.1 Implementazione dei modelli di Machine Learning

L'implementazione dei modelli di machine learning per la rilevazione delle anomalie nei nastri trasportatori industriali è stata eseguita utilizzando le librerie Python *sklearn* per il modello Random Forest e *xgboost* per il modello Gradient Boosting. Entrambi i modelli sono stati addestrati su un set di dati specifico, suddiviso accuratamente in training set e test set per garantire la robustezza delle prestazioni e la validità del modello. In particolare, i dati utilizzati per l'addestramento dei modelli sono stati raccolti da diversi file di dati grezzi, precedentemente preprocessati come spiegato nel capitolo precedente (sottocampionamento con DIADEM e Binning). Durante questa fase, è stato fondamentale selezionare accuratamente i file da utilizzare per l'addestramento, lasciandone alcuni da parte come "dati mai visti" per verificare la capacità dei modelli di generalizzare e risolvere il problema nel caso reale.
Per poter identificare correttamente le anomalie, nei dati è stata aggiunta una colonna Error, che indica con 1 la presenza di un errore e con 0 la normalità. Questo flag viene utilizzato dagli algoritmi di classificazione come etichetta target (y) durante l'addestramento. In pratica, i modelli imparano a distinguere tra stati normali e anomali basandosi sulle caratteristiche fornite, prevedendo y_pred che indica la probabilità o la classe di appartenenza per ogni campione nel dataset di test. I dati sono stati suddivisi in training set (70%) e test set (30%), inoltre è stata implementata una validazione incrociata a 5 fold per garantire che questi si adattassero bene ai dati di training e fossero in grado di generalizzare efficacemente sui dati non visti in precedenza.

In seguito il codice python per separare train e test set:

```python
from sklearn.model_selection import train_test_split

# Separate the target variable (Error) from the other features of the dataframe.
X = concatenated_DF.drop('Error', axis=1)
y = concatenated_DF['Error']

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state =1984)
```

```python
X_train
```

	Micr1_mean_binned	Micr1_skew_binned	Micr1_var_binned	Acc1X_mean_binned	Acc1X_skew_binned	Acc1X_var_binned	Acc1Y_mean_binned	Acc1Y_sk
25793	0.071637	0.094035	4.374311	0.055705	0.098888	0.669847	0.034495	
14357	-0.044106	-0.376459	4.384407	0.051386	0.148517	1.235245	0.046440	
15620	0.137475	-0.048821	3.349801	0.047647	-0.309106	0.753470	0.043908	
10988	0.257709	-0.125600	4.156038	0.056455	0.012344	1.005996	-0.003458	
48478	0.035106	-0.041245	3.440878	0.053779	-0.157061	0.789361	-0.001518	
...	...	...	...	...	...	...	...	
37960	0.179735	-0.070622	3.545582	0.042666	-0.318118	0.897428	-0.016969	
10009	-0.059764	-0.353036	4.028986	0.051028	-0.195224	0.784857	-0.026147	
1779	-0.130296	0.034122	3.435643	0.055864	-0.308474	1.028630	0.024231	
38655	0.082377	-0.054999	3.492467	0.043022	-0.312043	0.706234	-0.012278	
30319	0.080472	-0.076026	3.687134	0.052688	-0.452770	0.830450	-0.022123	

34709 rows × 66 columns

```python
y_train
```

```
25793    1
14357    1
15620    1
10988    0
48478    1
        ..
37960    1
10009    0
1779     0
38655    1
30319    1
Name: Error, Length: 34709, dtype: int64
```

Uno dei vantaggi di Python è proprio quello di poter utilizzare modelli già solidi tramite le librerie messe a disposizione, ma la libertà progettuale in questo caso sta nel tuning degli iperparametri, ovvero le variabili di ingresso che devono essere impostate prima dell'addestramento del modello e che influiscono significativamente sulle prestazioni finali.

Nel caso di **Random Forest**, gli iperparametri principali includono:

- **`n_estimators`:** il numero di alberi nella foresta.
- **`max_depth`:** la profondità massima degli alberi.
- **`min_samples_split`**: il numero minimo di campioni richiesti per suddividere un nodo.
- **`min_samples_leaf`**: il numero minimo di campioni richiesti per essere una foglia.

Per **XGB**, gli iperparametri principali includono:

- **`n_estimators`**: il numero di alberi di boosting da costruire.
- **`max_depth`:** la profondità massima degli alberi.
- **`learning_rate`:** la velocità con cui il modello adatta i pesi durante l'addestramento.
- **`min_child_weight`:** il peso minimo che deve avere un nodo figlio affinché venga suddiviso.
- **`colsample_bytree`:** la frazione di caratteristiche da campionare per ciascun albero.

Un'attenta regolazione di questi influisce non solo sulle prestazioni del modello, ma anche sui tempi di addestramento. Un parametro critico, come la profondità massima degli alberi (`max_depth`), ha un forte impatto sull'efficacia del modello: un valore troppo elevato può portare il modello a sovra-adattarsi ai dati di training, cogliendo dettagli troppo specifici (come il rumore), con il rischio di overfitting. Al contrario, una profondità troppo bassa può far sì che il modello non catturi correttamente i pattern presenti nei dati, con il rischio opposto di underfitting.
L'ottimizzazione degli iperparametri rappresenta quindi una fase cruciale nel

processo di modellazione, essenziale per migliorare le prestazioni e adattare i modelli al meglio alle caratteristiche specifiche del dataset. In questo progetto è stata condotta utilizzando la tecnica di Grid Search, che esplora sistematicamente lo spazio degli iperparametri per identificare la configurazione che massimizza la performance del modello.

4.2 Grid Search e la sua implementazione

La Grid Search è stata eseguita utilizzando la funzione `GridSearchCV` di `sklearn`, che permette di testare tutte le possibili combinazioni di iperparametri specificati, selezionando quelli che forniscono le migliori prestazioni in termini di accuratezza, f1_score o un'altra metrica specificata dal parametro `refit`, su un set di validazione incrociata.
In particolare ho chiamato la funzione facendo due prove con diverse combinazioni di iperparametri, come rappresentato in tabella:

	Hyperparameter	**Prova 1**	**Prova 2**
Random Forest	n_estimators	[30, 50, 100, 200]	[30, 50, 100]
	max_depth	[2, 3, 5, 10, 20, 40]	[2, 3, 5, 10]
	min_samples_split	[2, 4]	[1, 2, 3]
	min_samples_leaf	[1, 2]	[1, 2, 3]

	Hyperparameter	**Prova 1**	**Prova 2**
XGB	n_estimators	[30, 80, 130, 170]	[30, 60, 80, 90]
	learning_rate	[0.01, 0.1, 0.2]	[0.01, 0.05, 0.1]
	max_depth	[4, 6, 5, 10, 20]	[1, 2, 3]
	min_child_weight	[1, 5]	[1, 2, 3]
	colsample_bytree	[0.8, 1.0]	[0.8, 1.0]

Per completezza, la ricerca è stata fatta sia in termine di *accuracy* che di *f1_score*, sia per il modello di classificazione binaria, che per il multiclassificatore.
Il modello in uscita dalla gridsearch è stato poi salvato in locale tramite la funzione dump di joblib, in modo tale da poterlo caricare ed utilizzare facilmente in fase di test su dati sconosciuti.

Un esempio di codice per il caso binario è quello riportato qua sotto:

```
#f1_score
param_grid_rf = {
    'n_estimators': [30, 50, 100, 200],
    'max_depth': [2, 3, 5, 10, 20, 40],
    'min_samples_split': [2, 4],
    'min_samples_leaf': [1, 2],
}

start_time = time.time()

grid_search = GridSearchCV(estimator=model, param_grid=param_grid_rf, cv=5, n_jobs=-1,
                           scoring=scorers, refit='f1_score', verbose=2)
grid_search.fit(X_train, y_train)

end_time = time.time()
elapsed_time = end_time - start_time
print(f"The model took {elapsed_time/60: .3f} minutes to train.")

print("Migliori parametri:", grid_search.best_params_)
print("Migliore score:", grid_search.best_score_)

best_model = grid_search.best_estimator_
dump(best_model, 'miglior_modello_rf1F1.joblib')

y_pred = best_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
Fitting 5 folds for each of 96 candidates, totalling 480 fits
The model took  21.299 minutes to train.
Migliori parametri: {'max_depth': 40, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
Migliore score: 0.8513185371707153
              precision    recall  f1-score   support

           0       0.90      0.68      0.77      4044
           1       0.89      0.97      0.93     10832

    accuracy                           0.89     14876
   macro avg       0.89      0.83      0.85     14876
weighted avg       0.89      0.89      0.89     14876
```

Già qui si nota come per testare 480 diverse combinazioni di iperparametri ci siano voluti poco più di 21 minuti, contro le diverse ore del singolo modello sviluppato in precedenza.

Il procedimento è stato lo stesso sia per i classificatori binari che per i multiclassificatori.

I dataframe utilizzati sono sostanzialmente identici, con una piccola modifica dell'etichetta di errore, che nel caso binario segnala semplicemente la presenza di errore o di comportamento nominale, mentre nel caso multiclasse specifica di quale tipo di errore si tratta.

```
start_time = time.time()

grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid_xgb2, cv=5, n_jobs=-1,
                           scoring=scorers, refit='f1_score', verbose=2)
grid_search.fit(X_train, y_train)


end_time = time.time()
elapsed_time = end_time - start_time
print(f"The model took {elapsed_time/60: .3f} minutes to train.")

print("Migliori parametri:", grid_search.best_params_)
print("Migliore score:", grid_search.best_score_)

best_model = grid_search.best_estimator_

# Salvataggio del modello ottimizzato su disco
dump(best_model, 'miglior_multiclass_xgb2f1.joblib')

y_pred = best_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
[CV] END colsample_bytree=0.8, learning_rate=0.05, max_depth=2, min_child_weight=2, n_estimators=30; total time=
1.3s
[CV] END colsample_bytree=0.8, learning_rate=0.05, max_depth=2, min_child_weight=2, n_estimators=60; total time=
2.2s

The model took  5.608 minutes to train.
Migliori parametri: {'colsample_bytree': 0.8, 'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3, 'n_estim
ators': 90}
Migliore score: 0.8601125752469734
              precision    recall  f1-score   support

           0       0.84      0.69      0.76      4044
           1       0.72      0.75      0.73      3043
           2       0.98      1.00      0.99      1644
           3       0.91      0.99      0.95      6145

    accuracy                           0.86     14876
   macro avg       0.86      0.86      0.86     14876
weighted avg       0.86      0.86      0.86     14876
```

I migliori risultati in termini di refit = ‘accuracy’ e refit = ‘f1_score’ sono riportati nelle tabelle in seguito

Classificazione Binaria

Modello	Tempi (min)	Accuracy	Numero candidati
RF 1	21.183	0.89029	96
RF 2	8.634	0.87438	108
XGB 1	28.283	0.90066	240
XGB 2	2.153	0.87620	216

Modello	Tempi (min)	F1 Score	Numero candidati
RF 1	21.299	0.85132	96
RF 2	8.149	0.81937	108
XGB 1	20.569	0.87046	240
XGB 2	2.266	0.83293	216

Multiclassificazione

Modello	Tempi (min)	Accuracy	Numero candidati
RF 1	29.554	0.87055	96
RF 2	8.230	0.85528	108
XGB 1	32.352	0.85572	240
XGB 2	5.624	0.86323	216

Modello	Tempi (min)	F1 Score	Numero candidati
RF 1	28.913	0.86804	96
RF 2	9.709	0.85098	108
XGB 1	30.272	0.84803	240
XGB 2	5.608	0.86011	216

I risultati emersi dalla Grid Search sono particolarmente significativi, in quanto mostrano che i modelli ottimizzati possono passare alla fase di test con performance solide e tempi di addestramento contenuti.
Nello specifico, Random Forest ha dimostrato prestazioni eccellenti sia nel caso binario che nella multiclassificazione, sebbene XGB abbia offerto una maggiore efficienza in termini di tempi di esecuzione.

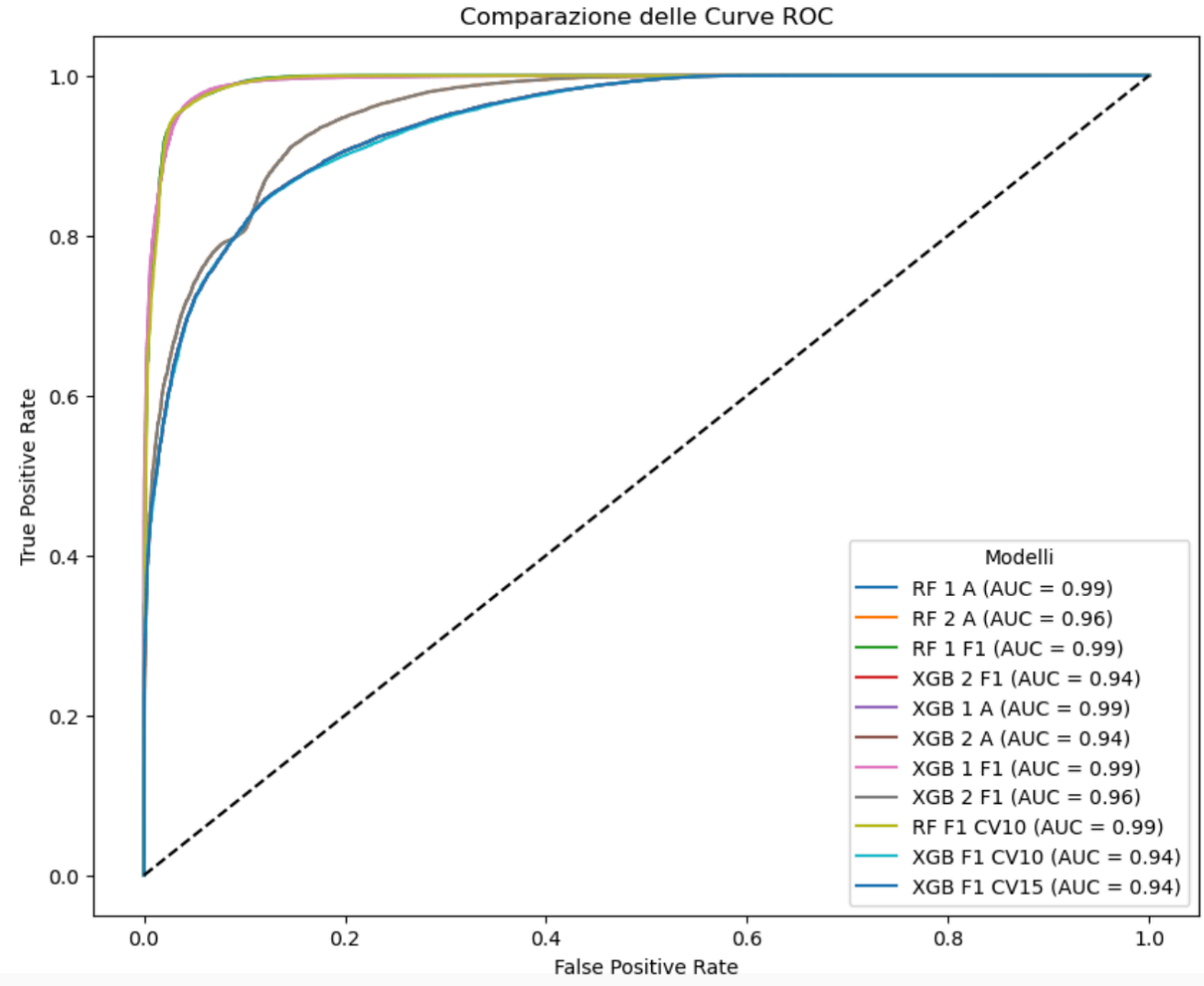
Analizzando i risultati in dettaglio:
nel caso binario, Random Forest (Prova 1) ha ottenuto un'accuracy di 0.8903 e un F1 score di 0.8513, con un tempo di 21.3 minuti per testare 96 candidati. Questo è un ottimo risultato, anche considerando che la seconda prova ha raggiunto prestazioni simili con tempi inferiori: Random Forest (Prova 2) ha registrato un'accuracy di 0.8744 e un F1 score pari a 0.8744, con un tempo di esecuzione di soli 8.8 minuti su 108 candidati.
D'altro canto, XGB ha dimostrato una maggiore rapidità nei tempi di esecuzione, specialmente nella seconda prova, dove ha testato 216 candidati in soli 2.2 minuti, ottenendo un'accuracy di 0.8762 e un F1 score di 0.8329. Questo fa di XGB un modello altamente efficiente, nonostante una leggera riduzione delle performance rispetto a Random Forest.
È interessante notare che nella prima prova XGB ha registrato un'accuracy di 0.9007 e un F1 score di 0.8705, ma a fronte di un tempo di esecuzione più lungo, pari a 28.3 minuti.
Per quanto riguarda la multiclassificazione, i risultati hanno mostrato un andamento simile. Random Forest (Prova 1) ha ottenuto un'accuracy di 0.8706 e un F1 score di 0.8680, con tempi di addestramento di 29.5 minuti su 96 candidati. La seconda prova ha ridotto significativamente i tempi di esecuzione (8.2 minuti) con un'accuracy di 0.8553 e un F1 score di 0.8510.
XGB, nel contesto multiclassificatore, ha mostrato tempi di esecuzione ancora più rapidi: nella seconda prova, ha completato l'addestramento in 5.6 minuti su 216 candidati, con un'accuracy di 0.8632 e un F1 score di 0.8601. Questi risultati indicano che XGB è particolarmente adatto a scenari dove la rapidità di esecuzione è cruciale, senza sacrificare troppo le prestazioni.

5. RISULTATI EMPIRICI

I migliori modelli in uscita dalla ricerca a griglia sono poi stati testati per capire se effettivamente, nel caso reale (ovvero in condizioni totalmente estranee a quelle di addestramento) i risultati fossero quelli attesi.

Comparando i modelli restituiti da GridSearch era apparso come la capacità di Random Forest fosse leggermente migliore di XGB, seppur entrambe eccellenti.

Nel caso binario, infatti, la ROC era la seguente:



A questo punto l'ultimo passo è stato quello di verificare empiricamente la bontà dei modelli per trarre le conclusioni finali.

5.1 Costruzione del Dataframe di Test

Il dataset di test utilizzato per la valutazione dei modelli è stato costruito con dati non utilizzati nella fase di addestramento per garantire la generalizzazione e la robustezza dei modelli. Questa separazione tra dati di allenamento e di test consente di valutare l'efficacia dei modelli su "dati mai visti" e garantisce che i modelli non siano overfittati ai dati di training.

Il dataset risultante contiene 8184 righe e 67 colonne, che rappresentano le medie, le varianze e la skewness delle misurazioni dei sensori. Seppur la dimensione del dataset di test è molto ridotta rispetto a quella di training, i campioni sono sufficienti per capire la capacità del modello di discriminare le varie classi.

La colonna target, chiamata Error, assume valori diversi in base alla tipologia di errore rilevato.

La divisione delle classi è la seguente:

- **Classificatore binario**:
  - 0 = Comportamento nominale
  - 1 = Errore (includendo rottura dei cappi, allentamento della catena, o rottura delle cravatte)

- **Multiclassificatore**:
  - 0 = Comportamento nominale
  - 1 = Rottura dei cappi
  - 2 = Allentamento della catena
  - 3 = Rottura delle cravatte

5.2 Risultati su dati mai visti

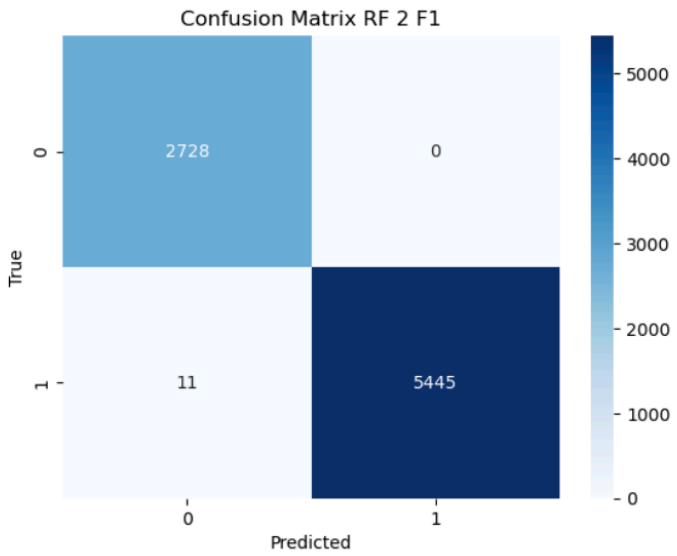
I risultati ottenuti dai migliori modelli sono riportati qui sotto:

Caso binario:

Classification Report for model 3:
RF 2 F1

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2728
1	1.00	1.00	1.00	5456
accuracy			1.00	8184
macro avg	1.00	1.00	1.00	8184
weighted avg	1.00	1.00	1.00	8184

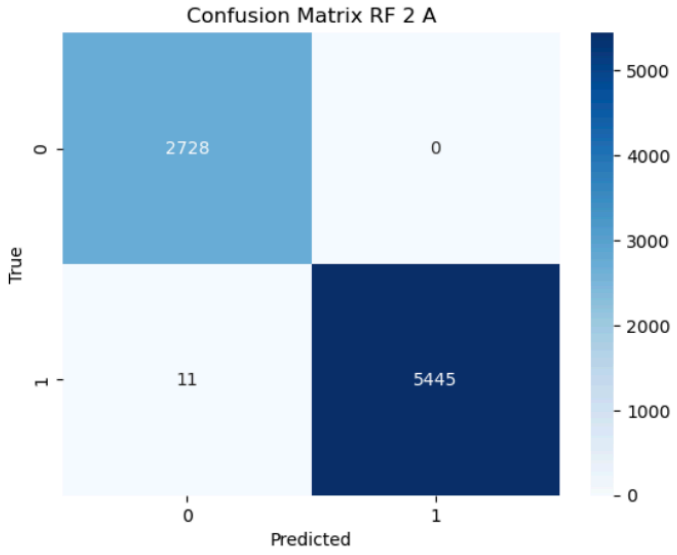
Confusion Matrix
[[2728 0]
[11 5445]]



Classification Report for model 4:
RF 2 A

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2728
1	1.00	1.00	1.00	5456
accuracy			1.00	8184
macro avg	1.00	1.00	1.00	8184
weighted avg	1.00	1.00	1.00	8184

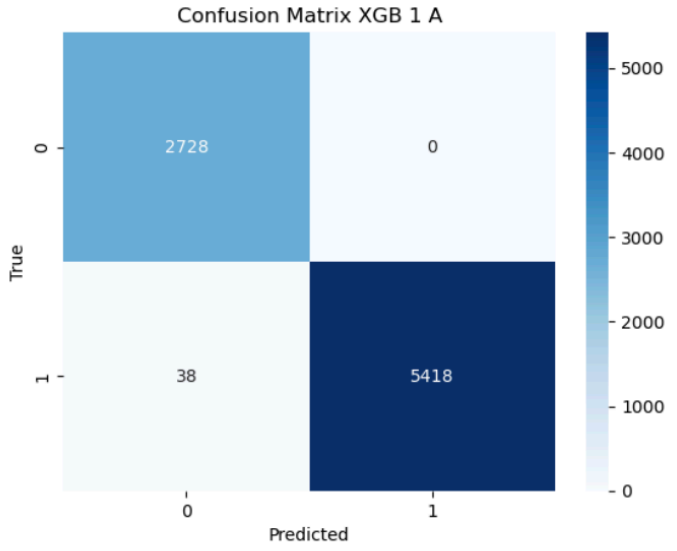
Confusion Matrix
[[2728 0]
[11 5445]]



Classification Report for model 5:
XGB 1 A

	precision	recall	f1-score	support
0	0.99	1.00	0.99	2728
1	1.00	0.99	1.00	5456
accuracy			1.00	8184
macro avg	0.99	1.00	0.99	8184
weighted avg	1.00	1.00	1.00	8184

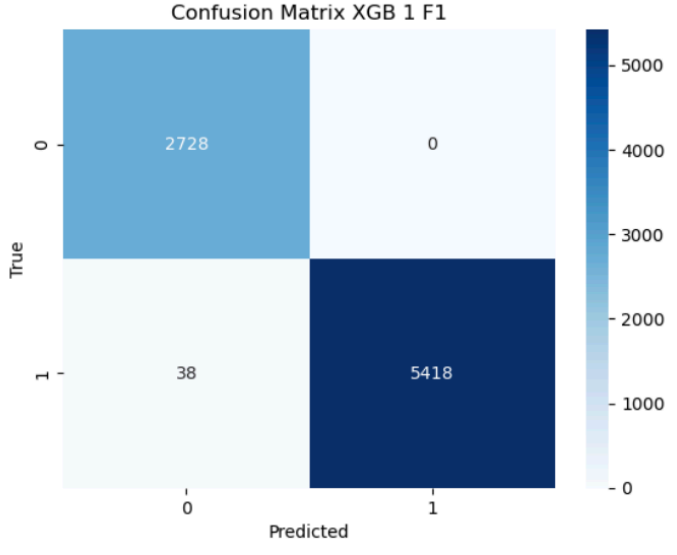
Confusion Matrix
[[2728 0]
[38 5418]]



Classification Report for model 6:
XGB 1 F1

	precision	recall	f1-score	support
0	0.99	1.00	0.99	2728
1	1.00	0.99	1.00	5456
accuracy			1.00	8184
macro avg	0.99	1.00	0.99	8184
weighted avg	1.00	1.00	1.00	8184

Confusion Matrix
[[2728 0]
[38 5418]]



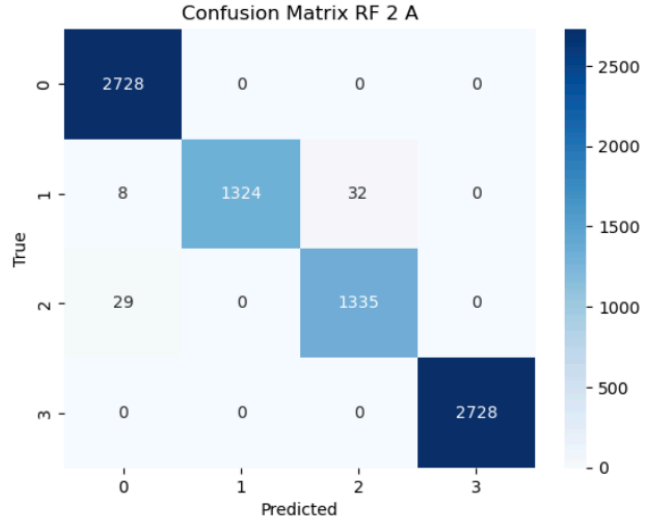
Multiclassificazione:

Classification Report for model 10, RF 2 A

	precision	recall	f1-score	support
0	0.99	1.00	0.99	2728
1	1.00	0.97	0.99	1364
2	0.98	0.98	0.98	1364
3	1.00	1.00	1.00	2728
accuracy			0.99	8184
macro avg	0.99	0.99	0.99	8184
weighted avg	0.99	0.99	0.99	8184

Confusion Matrix

```
[[2728    0    0    0]
 [   8 1324   32    0]
 [  29    0 1335    0]
 [   0    0    0 2728]]
```

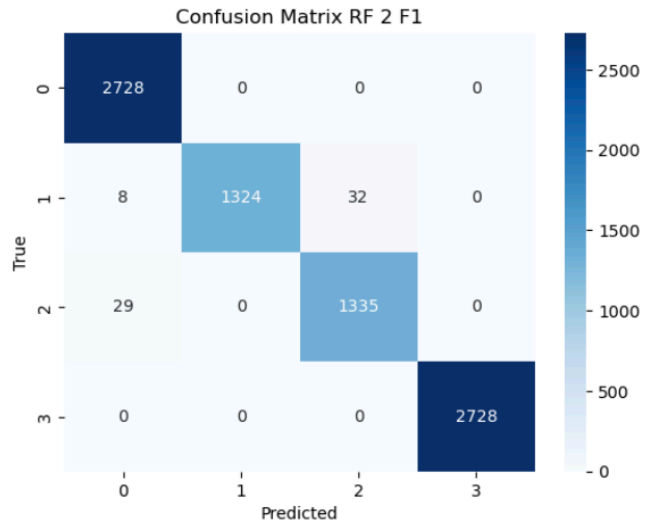


Classification Report for model 11, RF 2 F1

	precision	recall	f1-score	support
0	0.99	1.00	0.99	2728
1	1.00	0.97	0.99	1364
2	0.98	0.98	0.98	1364
3	1.00	1.00	1.00	2728
accuracy			0.99	8184
macro avg	0.99	0.99	0.99	8184
weighted avg	0.99	0.99	0.99	8184

Confusion Matrix

```
[[2728    0    0    0]
 [   8 1324   32    0]
 [  29    0 1335    0]
 [   0    0    0 2728]]
```

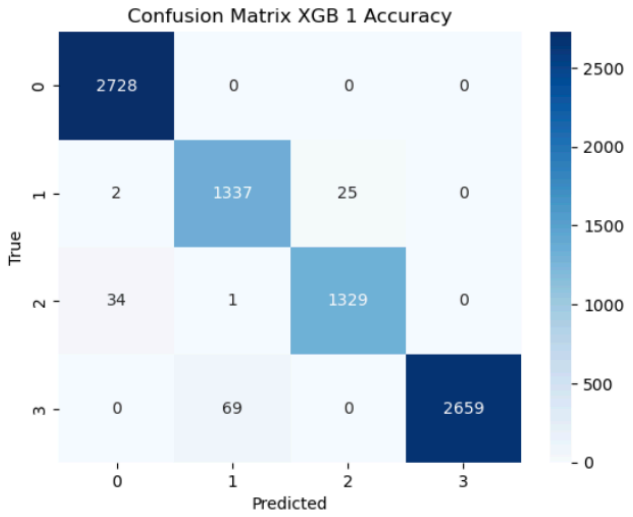


Classification Report for model 6, XGB 1 Accuracy

	precision	recall	f1-score	support
0	0.99	1.00	0.99	2728
1	0.95	0.98	0.96	1364
2	0.98	0.97	0.98	1364
3	1.00	0.97	0.99	2728
accuracy			0.98	8184
macro avg	0.98	0.98	0.98	8184
weighted avg	0.98	0.98	0.98	8184

Confusion Matrix

```
[[2728    0    0    0]
 [   2 1337   25    0]
 [  34    1 1329    0]
 [   0   69    0 2659]]
```

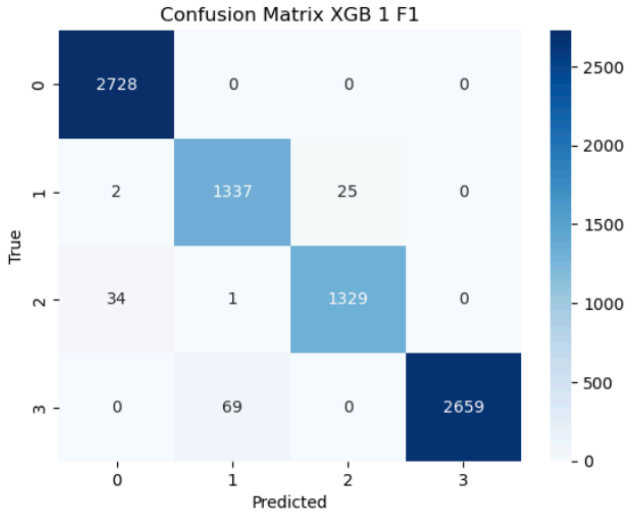


Classification Report for model 7, XGB 1 F1

	precision	recall	f1-score	support
0	0.99	1.00	0.99	2728
1	0.95	0.98	0.96	1364
2	0.98	0.97	0.98	1364
3	1.00	0.97	0.99	2728
accuracy			0.98	8184
macro avg	0.98	0.98	0.98	8184
weighted avg	0.98	0.98	0.98	8184

Confusion Matrix

```
[[2728    0    0    0]
 [   2 1337   25    0]
 [  34    1 1329    0]
 [   0   69    0 2659]]
```



Dove le configurazioni iperparametriche dei modelli sono

Classificatore binario	Multiclassificatore
Random Forest 2 (Accuracy): o Max Depth: 10 o Min Samples Leaf: 1 o Min Samples Split: 3 o N Estimators: 100	**Random Forest 2 (Accuracy)**: o Max Depth: 10 o Min Samples Leaf: 2 o Min Samples Split: 1 o N Estimators: 100
Random Forest 2 (F1 Score): o Max Depth: 10 o Min Samples Leaf: 1 o Min Samples Split: 3 o N Estimators: 100	**Random Forest 2 (F1 Score)**: o Max Depth: 10 o Min Samples Leaf: 2 o Min Samples Split: 1 o N Estimators: 100
XGB 2 (Accuracy): o Colsample bytree: 1.0 o Learning Rate: 0.1 o Max Depth: 3 o Min Child Weight: 2 o N Estimators: 90	**XGB 1 (F1 Score)**: o Colsample bytree: 0.8 o Learning Rate: 0.15 o Max Depth: 15 o Min Child Weight: 5 o N Estimators: 170
XGB 2 (F1 Score): o Colsample bytree: 1.0 o Learning Rate: 0.1 o Max Depth: 3 o Min Child Weight: 2 o N Estimators: 90	**XGB 1 (Accuracy)**: o Colsample bytree: 0.8 o Learning Rate: 0.15 o Max Depth: 15 o Min Child Weight: 5 o N Estimators: 170

5.3 Analisi dei risultati

I risultati ottenuti dalla classificazione binaria e dalla multiclassificazione dimostrano che sia Random Forest (RF) sia XGB offrono eccellenti prestazioni nel rilevamento delle anomalie nei nastri trasportatori. Nel caso della classificazione binaria, Random Forest ha superato XGB in termini di accuratezza e F1 score, con soli 11 falsi negativi rispetto ai 38 di XGB. Questo suggerisce che entrambi i modelli funzionano molto bene, ma RF riesce a bilanciare meglio precisione e recall nella classificazione degli errori rispetto a XGB. La maggiore capacità di generalizzazione di RF potrebbe essere attribuita alla sua natura di modello di ensemble, che sfrutta una varietà di alberi decisionali per adattarsi meglio alle caratteristiche specifiche del dataset di test. Le configurazioni iperparametriche di RF, in particolare la profondità massima degli alberi (max_depth = 10) e il numero di alberi (n_estimators = 100), hanno permesso al modello di cogliere una buona quantità di informazioni senza soffrire di overfitting. Il fatto che questi parametri siano stati mantenuti costanti sia nella prova per l'accuracy sia in quella per l'F1 score, conferma la robustezza di RF in questo contesto. Al contrario, XGB, pur essendo efficiente, ha avuto difficoltà a bilanciare precisione e recall con la stessa efficacia, probabilmente a causa della complessità del boosting, che richiede una più fine regolazione dei parametri, come il learning rate e il max_depth. Questo è dimostrato dal fatto che, nonostante il modello XGB utilizzasse una configurazione di colsample_bytree a 1.0 e una profondità degli alberi ridotta a 3, il numero di falsi negativi è stato più elevato.

Per quanto riguarda la multiclassificazione, RF ha ancora una volta dimostrato una leggera superiorità in termini di accuratezza e F1 score. La capacità di RF di gestire i vari livelli di complessità del dataset multiclasse è emersa chiaramente, con un lieve miglioramento nei punteggi di precisione e F1 score per ciascuna classe rispetto a XGB. Tuttavia, si nota una leggera imprecisione di

RF nel classificare i cappi, con 40 errori su un supporto di 1364 campioni, e l'allentamento della catena, con 29 errori (su 1364 campioni). Ciò si traduce in una capacità del modello di classificare correttamente il 99,2% degli esempi (8115/8184), contro il 98,4% di XGB. Anche in questo caso, la configurazione iperparametrica di RF, con una profondità degli alberi ottimizzata e un numero adeguato di alberi, ha fornito prestazioni stabili e prevedibili.
Le configurazioni di XGB, che includevano una combinazione di max_depth = 3 e learning_rate = 0.1, hanno portato a un'efficienza superiore in termini di tempo di esecuzione, ma a scapito di una lieve riduzione nella precisione e recall. XGB ha mostrato difficoltà nella gestione della complessità del problema multiclassificatore rispetto a RF, soprattutto nella classificazione delle classi con meno esempi, come i cappi e l'allentamento della catena.
In conclusione, si può affermare che entrambi i modelli raggiungono prestazioni eccellenti, dimostrando la solidità dell'approccio utilizzato in questo progetto. RF si distingue per la sua capacità di bilanciare meglio precisione e recall, rendendolo una scelta ottimale per problemi complessi con un dataset multiclasse. XGB, pur essendo estremamente efficiente in termini di tempo, richiede un tuning iperparametrico più accurato per eguagliare la precisione di RF nei casi più complessi.

5.4 Interpolazione ed Estrapolazione

Come già discusso, l'acquisizione e la qualità dei dati sono fondamentali per la costruzione di un modello solido e affidabile. La veridicità dei dati raccolti e la loro rappresentatività del caso reale garantiscono che il modello possa generalizzare correttamente. Elementi cruciali come il posizionamento dei sensori, la loro quantità e qualità, influiscono direttamente sulla precisione e accuratezza delle informazioni ottenute. Tuttavia, anche il numero di rilevazioni rappresenta un fattore di costo significativo, poiché comporta il fermo macchina e l'impiego di personale. Per questo motivo, è essenziale adottare strategie che

ottimizzino non solo la qualità dei dati, ma anche la loro quantità. Un modello efficace deve essere in grado di interpolare ed estrapolare i dati in modo accurato, generalizzando bene anche con una disponibilità ridotta di esempi durante l'addestramento. In termini semplici, interpolazione si riferisce alla capacità del modello di predire correttamente valori all'interno di un intervallo di dati osservati, mentre estrapolazione riguarda la predizione di valori al di fuori di tale intervallo. Lo sviluppo di un modello capace di svolgere entrambe le funzioni con precisione permette di ridurre significativamente il numero di acquisizioni necessarie, con un conseguente risparmio in termini di spazio di archiviazione, tempo e risorse economiche. Nel caso specifico di questo progetto, i dati campionati a 51,2 kHz e organizzati in file da un minuto richiedevano circa 589,8 MB per file, per un totale di diverse decine di GB per l'intero set di acquisizioni.

Durante l'ultima fase del progetto, uno degli obiettivi principali è stato verificare la capacità del modello di interpolare ed estrapolare correttamente i dati, in modo da ridurre il numero di acquisizioni necessarie. Sebbene la metodologia di addestramento sia rimasta invariata, sono stati introdotti alcuni accorgimenti nella selezione dei file di training e di verifica.

Per quanto riguarda l'interpolazione, sono state selezionate prove con allentamento della catena a 1 mm e 5 mm, verificando successivamente se il modello fosse in grado di riconoscere correttamente dati con allentamento a 2 mm, mai visti in fase di training.

```
# cerco di interpolare i 2mm
file_paths_catena_interp = [
    "/Catena/data_Prova41_0003_cat1_sampled.tdms",       # 1mm
    "/Catena/data_Prova41_0009_catena_1mm_sampled.tdms", # 1mm
    "/Catena/data_Prova43_0002_cat5_sampled.tdms",       # 5mm
]
```

```
# cerco di estrapolare i 5mm
file_paths_catena_estrap = [
    "/Catena/data_Prova41_0003_cat1_sampled.tdms",       # 1mm
    "/Catena/data_Prova41_0009_catena_1mm_sampled.tdms", # 1mm
    "/Catena/data_Prova42_0004_catena_2mm_sampled.tdms", # 2mm
]
```

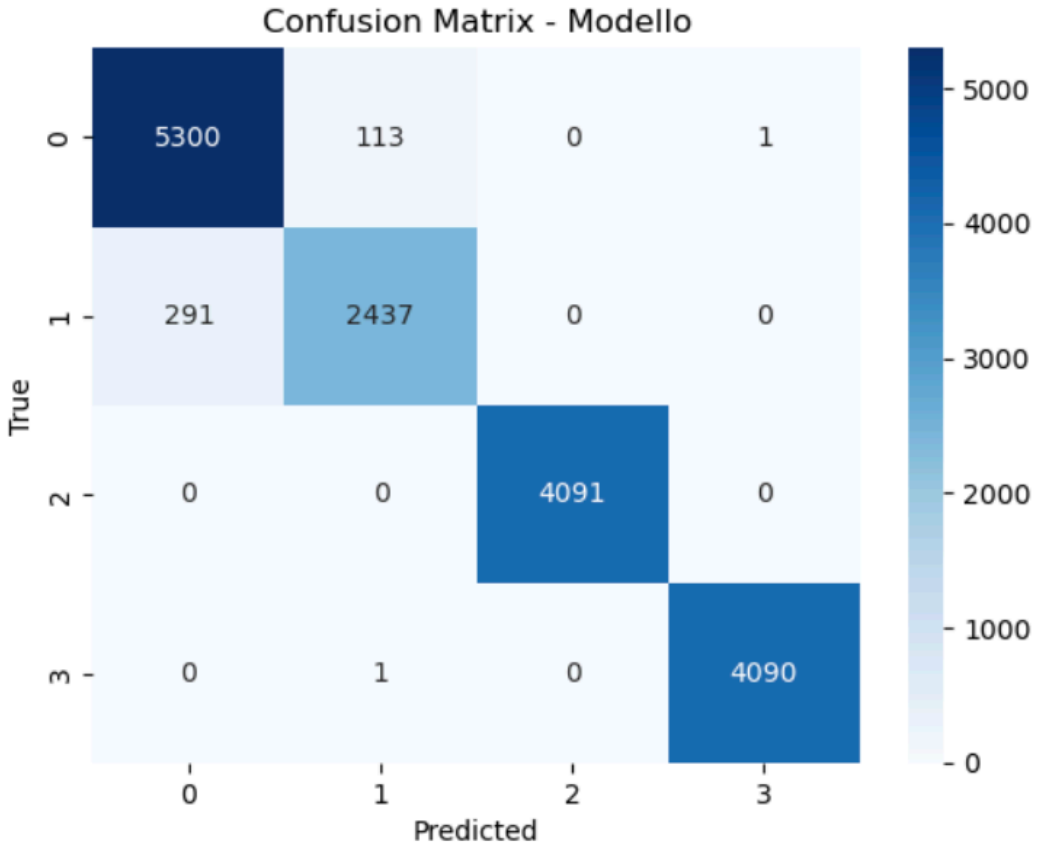
I risultati hanno dimostrato che il modello è stato in grado di interpolare con successo, riconoscendo correttamente gli errori in condizioni non presenti durante l'addestramento.

Questo evidenzia la capacità del modello di predire accuratamente all'interno degli intervalli di dati già osservati. L'estrapolazione, invece, ha richiesto una verifica più complessa, poiché il modello doveva predire dati al di fuori dell'intervallo di osservazione. Anche in questo caso, i risultati si sono dimostrati molto positivi, confermando che il modello è in grado di gestire condizioni operative che si discostano da quelle viste durante l'addestramento, almeno per quanto riguarda l'allentamento della catena.

Per quanto riguarda i cappi e le cravatte, il problema dell'interpolazione non è definibile con precisione, poiché i dati relativi a questi elementi rientrano sempre in un contesto di estrapolazione. Anche in queste circostanze, il modello ha dimostrato buone capacità di estrapolazione, sebbene ulteriori studi potrebbero essere necessari per migliorare ulteriormente la sua capacità di generalizzazione.

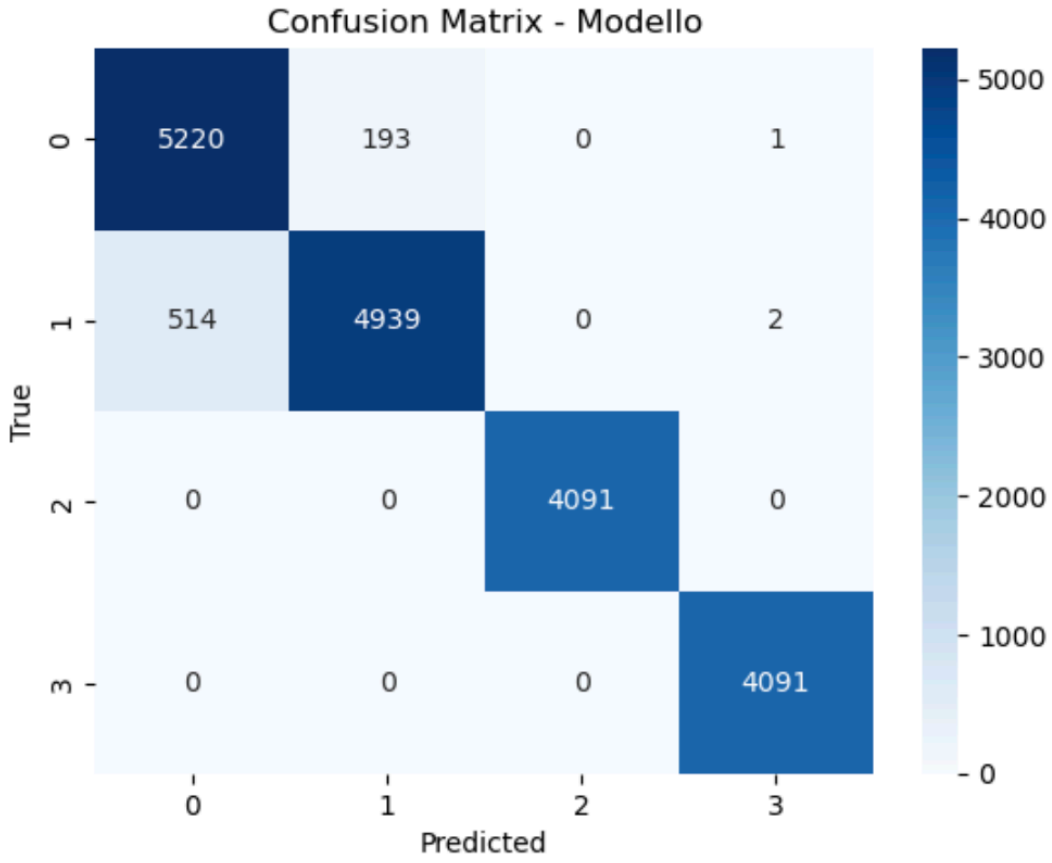
Estrapolazione 5mm

```
Classification Report
              precision    recall  f1-score   support

           0       0.95      0.98      0.96      5414
           1       0.96      0.89      0.92      2728
           2       1.00      1.00      1.00      4091
           3       1.00      1.00      1.00      4091

    accuracy                           0.98     16324
   macro avg       0.98      0.97      0.97     16324
weighted avg       0.98      0.98      0.97     16324

Confusion Matrix
[[5300  113    0    1]
 [ 291 2437    0    0]
 [   0    0 4091    0]
 [   0    1    0 4090]]
```



Interpolazione 2mm

```
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.96      0.94      5414
           1       0.96      0.91      0.93      5455
           2       1.00      1.00      1.00      4091
           3       1.00      1.00      1.00      4091

    accuracy                           0.96     19051
   macro avg       0.97      0.97      0.97     19051
weighted avg       0.96      0.96      0.96     19051

Confusion Matrix
[[5220  193    0    1]
 [ 514 4939    0    2]
 [   0    0 4091    0]
 [   0    0    0 4091]]
```



I modelli addestrati hanno delle ottime prestazioni, ma con una leggera perdita di precisione per quanto riguarda la classificazione dei cappi, che rimane comunque del 90,5%.

6. CONCLUSIONI E SVILUPPI FUTURI

Questa esperienza ha segnato un notevole passo in avanti nello sviluppo del sistema di manutenzione predittiva per nastri trasportatori industriali studiato da CNR-STIIMA. Uno dei contributi principali è stato l'ottimizzazione del processo di preprocessing dei dati tramite tecniche di binning, che ha ridotto la complessità computazionale pur mantenendo le informazioni chiave. Questo ha reso possibile l'analisi efficiente dei dati senza compromettere la precisione dei risultati, permettendo ai modelli di apprendere in modo efficace dai segnali dei sensori. Nello specifico, la funzione di autocorrelazione ha rilevato delle periodicità nei dati, che hanno consentito la regolazione della bin size a 440 campioni. Per ogni finestra sono state poi estrapolate le metriche di media, varianza e asimmetria. Questi dati compressi sono poi stati forniti ai modelli Random Forest e XGB, selezionati per la loro capacità di gestire dataset complessi e per la loro robustezza in contesti industriali. La scelta è sia supportata dalla letteratura che da ricerche precedenti del team.
L'ottimizzazione degli iperparametri di RF e XGB, effettuata tramite GridSearchCV, ha consentito di esplorare varie configurazioni tramite validazione incrociata a 5 fold, migliorando le prestazioni complessive dei modelli. Durante la fase di Grid Search, i modelli sono stati confrontati in termini di tempi di addestramento e prestazioni, sia per la classificazione binaria che per la multiclassificazione.

Sono riportati qui sotto i principali risultati ottenuti in fase di training e test, per i casi di classificazione binaria e di multiclassificazione.

Caso binario:

I risultati numerici della ricerca iperparametrica hanno mostrato che:

- Il modello Random Forest (RF 1), con un tempo di addestramento di circa 21 minuti, ha ottenuto un'accuracy di 0.8903 e un F1-score di 0.8513.

- RF 2, una versione più ottimizzata e rapida, ha ridotto i tempi di esecuzione a 8.6 minuti, mantenendo un'accuracy di 0.8744 e un F1-score di 0.8744, dimostrando che la riduzione del tempo di esecuzione ha avuto un impatto limitato sulle prestazioni complessive.
- XGB 1, con tempi di addestramento di 28 minuti, ha raggiunto un'accuracy di 0.9007 e un F1-score di 0.8705, ottenendo risultati leggermente migliori rispetto a RF in termini di precisione.
- XGB 2, invece, si è addestrato in soli 2.15 minuti, con un'accuracy di 0.8762 e un F1-score di 0.8329, dimostrando che un modello più rapido può comunque offrire prestazioni competitive, a scapito di una piccola riduzione della precisione.

Multiclassificazione:

- RF 1, con un tempo di addestramento di circa 29 minuti, ha ottenuto un'accuracy di 0.8706 e un F1-score di 0.8680.
- RF 2, più ottimizzato nei tempi (circa 8 minuti), ha registrato un'accuracy di 0.8553 e un F1-score di 0.8510. Questo dimostra che una configurazione meno complessa può comunque garantire ottime prestazioni con un notevole risparmio di tempo.
- XGB 1, nonostante i tempi di addestramento più lunghi (32 minuti), ha ottenuto un'accuracy di 0.8557 e un F1-score di 0.8480, mentre XGB 2 si è addestrato in 5.6 minuti, con un'accuracy di 0.8632 e un F1-score di 0.8601, dimostrando la sua efficienza computazionale.

Si può notare come un aumento di tempi di addestramento non ha necessariamente un impatto significativo sulle prestazioni, e che nello specifico, gli iperparametri che allungano maggiormente i tempi sono il numero di alberi decisionali generati e la loro profondità.

Test:

Sui dati mai visti, sia i modelli Random Forest che XGB hanno mostrato ottime prestazioni nel rilevamento delle anomalie nei nastri trasportatori.

Nel caso della classificazione binaria, Random Forest ha superato XGB in termini di F1 score e accuratezza, con soli 11 falsi negativi contro i 38 di XGB, su un totale di 8184 campioni, ottenendo rispettivamente accuratezze del 99.87% e 99,53%. Questo suggerisce che RF riesce meglio a bilanciare precision e recall, riducendo il rischio di falsi negativi, critici in un contesto industriale.

Per la multiclassificazione, RF ha superato nuovamente XGB, riuscendo a distinguere in maniera più accurata tra le diverse tipologie di errori con il 99,2% contro il 98,4%. I punteggi ottenuti mostrano che RF ha una maggiore stabilità nella gestione di dati complessi e con più classi, riuscendo a ridurre gli errori di classificazione rispetto a XGB, che necessiterebbe di un tuning più accurato degli iperparametri per migliorare le sue prestazioni in questo contesto.

Impatto del lavoro sul progetto

I risultati ottenuti hanno fornito una metodologia solida e ben ottimizzata per l'addestramento di modelli di machine learning in questo specifico scenario industriale, contribuendo significativamente all'efficacia e all'affidabilità della manutenzione predittiva. Il sistema sviluppato non solo migliora l'efficienza operativa, prevenendo guasti e riducendo i tempi di inattività, ma offre anche un approccio versatile grazie alla combinazione dei modelli binari e multiclassificatori. La capacità di rilevare rapidamente errori e di classificare in modo accurato la natura dei guasti consente una manutenzione più precisa e mirata.

Per il CNR, questo progetto rappresenta una base concreta e robusta per future implementazioni, con un modello che potrà facilmente integrarsi nei loro processi esistenti. L'applicazione di questa metodologia nel contesto reale

permetterà di migliorare la gestione delle risorse industriali e potrà fungere da punto di partenza per sviluppare ulteriori soluzioni, come l'integrazione di un digital twin in tempo reale. La solidità dei risultati conferma che i modelli addestrati possono essere una risorsa chiave per il monitoraggio continuo e predittivo, portando un valore aggiunto significativo alle attività del CNR.

Direzioni future

Nonostante i risultati eccellenti, il progetto presenta ancora alcune limitazioni che dovranno essere affrontate per portare il sistema verso un'applicazione real-time e un Digital Twin. I risultati ottenuti dipendono dalla macchina utilizzata, un MacBook Air M1 (2020) con 8 GB di RAM, che ha imposto alcuni limiti di natura computazionale, sebbene abbia dimostrato che non è necessario l'uso di un supercalcolatore. Le prestazioni potrebbero essere migliorate con l'uso di hardware più potente, soprattutto in termini di RAM e capacità GPU, poiché la Grid Search e il training su modelli complessi come XGB richiedono molte risorse.

Inoltre, il sistema dovrà evolversi verso una maggiore efficienza nella raccolta e gestione dei dati. Un'implementazione futura dovrebbe includere l'uso di sliding windows per consentire la classificazione in tempo reale, migliorando la gestione del flusso di dati continui. Infine, per garantire una reale predizione dei guasti, sarà necessario perfezionare ulteriormente il sistema di classificazione multiclassificatore e valutare l'uso di modelli che possano gestire con accuratezza situazioni che non rientrano nei casi di errore già addestrati, introducendo magari una classe "altro" per guasti inaspettati.

In conclusione, il progetto ha ottimizzato con successo i modelli di classificazione delle anomalie, raggiungendo l'obiettivo del tirocinio, ma ulteriori sviluppi saranno necessari per portare il sistema a un'applicazione real-time efficiente e robusta, integrabile in un contesto di Digital Twin e manutenzione predittiva avanzata.

Bibliografia

Ahmad, R., & Kamaruddin, S. (2012). An overview of time-based and condition-based maintenance in industrial application. *Computers & Industrial Engineering*, 63(1), 135–149. https://doi.org/10.1016/j.cie.2012.02.002

Alam, M. N., & Yao, X. (2019). An ensemble of diverse classifiers for anomaly detection. *Knowledge-Based Systems*, 182, 104842. https://doi.org/10.1016/j.knosys.2019.07.003

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Braiek, H. B., & Khomh, F. (2020). On testing machine learning programs. *Journal of Systems and Software*, 164, 110542. https://doi.org/10.1016/j.jss.2020.110542

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

Carletti, M., Masiero, C., Beghi, A., & Susto, G. A. (2019). Explainable machine learning in Industry 4.0: Evaluating feature importance in anomaly detection to enable root cause analysis. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)* (pp. 3891–3901). IEEE. https://doi.org/10.1109/SMC.2019.8913901

Carvalho, T. P., Soares, F. A. A. M. N., Vita, R., Francisco, R. P., Basto, J. P., & Alcalá, S. G. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, 137, 106024. https://doi.org/10.1016/j.cie.2019.106024

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM. https://doi.org/10.1145/2939672.2939785

Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2–3), 103–130. https://doi.org/10.1023/A:1007413511361

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Friedman, J. H., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2), 337–407. https://doi.org/10.1214/aos/1016218223

Garcia, S., Luengo, J., & Herrera, F. (2015). *Data Preprocessing in Data Mining*. Springer. https://doi.org/10.1007/978-3-319-10247-4

Guo, J., & Shen, Y. (2022). Online anomaly detection of industrial IoT based on hybrid machine learning architecture. *Complexity*, 2022, Article ID 8568917. https://doi.org/10.1155/2022/8568917

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2ª ed.). Springer. https://doi.org/10.1007/978-0-387-84858-7

Jardine, A. K. S., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7), 1483–1510. https://doi.org/10.1016/j.ymssp.2005.09.012

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1), 81–106. https://doi.org/10.1007/BF00116251

Qi, Q., & Tao, F. (2018). Digital twin and big data towards smart manufacturing and industry 4.0: 360-degree comparison. *IEEE Access*, 6, 3585–3593. https://doi.org/10.1109/ACCESS.2018.2793265

Regattieri, A., & Manzini, R. (2007). *Manutenzione dei sistemi di produzione*. McGraw-Hill Education.

Rokach, L., & Maimon, O. (2005). Top-down induction of decision trees classifiers: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4), 476–487. https://doi.org/10.1109/TSMCC.2004.843247

Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3ª ed.). Prentice Hall.

Schapire, R. E., & Freund, Y. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 119–139. https://doi.org/10.1006/jcss.1997.1504

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

Tancredi, G. P. C., Vignali, G., & Bottani, E. (2022). Integration of digital twin, machine-learning and Industry 4.0 tools for anomaly detection: An application to a food plant. *Sensors*, 22(11), 4143. https://doi.org/10.3390/s22114143

Wuest, T., Weimer, D., Irgens, C., & Thoben, K. D. (2016). Machine learning in manufacturing: Advantages, challenges, and applications. *Production & Manufacturing Research*, 4(1), 23–45. https://doi.org/10.1080/21693277.2016.1192517