Relazione esame

Studio e implementazione di algoritmi di machine learning per il rilevamento di anomalie su serie temporali real-time

Federico Brescia s4511157

Edge Computing

UNIVERSITÀ DEGLI STUDI
DI GENOVA

Indice

1 Introduzione

L'edge computing è un modello di calcolo distribuito nel quale l'elaborazione dei dati avviene il più vicino possibile a dove i dati vengono generati o al loro utilizzatore, migliorando i tempi di risposta, risparmiando sulla larghezza di banda, la sicurezza e sfruttando la flessibilità del cloud ibrido senza applicare alti carichi al cloud.

Questo modello si contrappone al cloud computing che richiede di trasferire i dati da e verso dei data center remoti, con possibili problemi di latenza, costo e sicurezza. L'edge computing sfrutta i dispositivi edge, come sensori, videocamere, PC industriali e gateway, per eseguire applicazioni di livello enterprise in prossimità delle fonte di dati, come le reti IoT, i dispositivi mobili e le smart city.

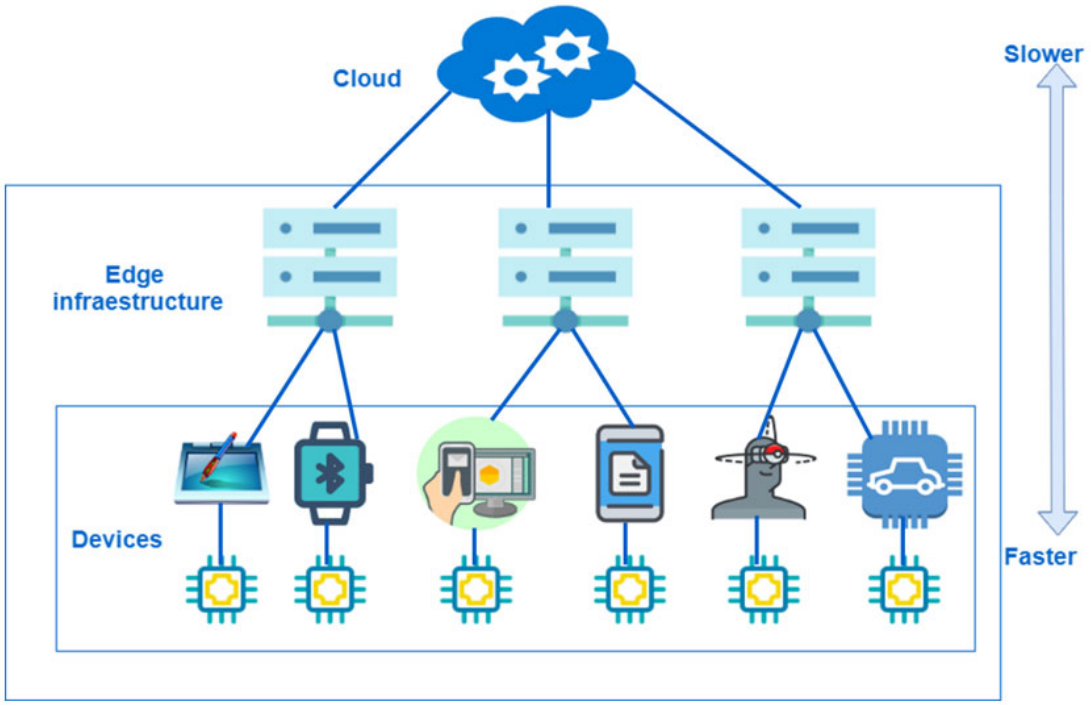

Figura 1: Schema del compito dell'edge computing

L'edge computing offre numerosi vantaggi, tra cui:

- Una maggiore velocità di elaborazione e analisi dei dati, grazie alla riduzione della distanza tra i dati e le applicazioni.

- Una minore dipendenza dalla connettività di rete e dalla larghezza di banda, riducendo il rischio di interruzioni o ritardi nel flusso dei dati.
- Una maggiore sicurezza e privacy dei dati sensibili, che possono essere elaborati localmente senza doverli inviare al cloud.
- Una maggiore scalabilità e flessibilità delle soluzioni IT, che possono adattarsi alle esigenze specifiche di ogni scenario applicativo.

L'edge computing è una tecnologia in rapida evoluzione, che si basa sullo sviluppo dell'intelligenza artificiale e dell'IoT.

L'edge computing ha molte applicazioni possibili in diversi settori, come la sanità, l'industria 4.0, il retail e i trasporti. Tuttavia, l'edge computing presenta anche delle sfide e dei requisiti necessari per la sua implementazione, come la gestione dell'infrastruttura IT distribuita, l'integrazione tra i dispositivi edge e il cloud e la garanzia della qualità e dell'affidabilità dei dati.

In questo contesto, il rilevamento di anomalie su serie temporali è una sfida importante, poiché può aiutare a prevenire guasti, ottimizzare le operazioni e migliorare la qualità dei servizi.

Questa relazione si propone di confrontare e implementare, con la ditta MBDA, diversi algoritmi di machine learning per il rilevamento di anomalie su serie temporali all'interno di un datalogger.

Il data logger è un dispositivo che regista i dati ambientali (temperatura, umidità, pressione, ecc.) in un microcontrollore stm32 a basso consumo. Gli algoritmi per il rilevamento delle anomalie dovranno essere implementati sullo stesso micro come task nel sistema operativo real-time FreeRtos.

L'obbiettivo è valutare le prestazioni e i vantaggi di questi algoritmi in termini di accuratezza, efficienza e scalabilità in un dispositivo prototipo.

2 Rilevamento di anomalie real-time

2.1 Definizione di serie temporale

Una serie temporale è una sequenza di dati discreti presi a intervalli di tempo successivi uguali. Le serie temporali sono utilizzate in molti campi della scienza applicata e dell'ingegneria che coinvolgono misurazioni temporali, come la statistica, l'elaborazione del segnale, il riconoscimento dei pattern, la finanza, le previsioni meteorologiche, l'astronomia e l'ingegneria delle comunicazioni. L'analisi delle serie temporali comprende metodi per analizzare i dati delle serie temporali per estrarre statistiche significative e altre caratteristiche dei dati. La differenza tra un semplice compito di regressione e un'analisi delle serie temporali è che, in quest'ultimo caso, il modello deve non solo apprendere la correlazione tra le caratteristiche ma anche quella con il tempo.

I ricercatori hanno definito due categorie principali di serie temporali:

- Serie temporale univariabile: Una serie temporale univariabile $X = x_{tt\in T}$ è definita come un insieme ordinato di osservazioni a valori reali, x_t, dove ogni osservazione è registrata in un momento specifico $t \in T \subseteq Z+$.
- Serie temporale multivariabile: Una serie temporale multivariabile $X = x_{tt\in T}$ è definita come un insieme ordinato di vettori k-dimensionali, ognuno dei quali è registrato in un momento specifico $t \in T \subseteq Z+$ e consiste di k osservazioni a valori reali, $x_t = (x_{1t}, \ldots, x_{kt})$.

Un metodo di rilevamento univariabile considera solo una singola variabile dipendente dal tempo, mentre un metodo di rilevamento multivariabile è in grado di lavorare contemporaneamente con più di una variabile. Inoltre, il metodo di rilevamento può essere univariabile anche se i dati in ingresso sono una serie temporale multivariabile perché può essere eseguita un'analisi individuale su ogni variabile dipendente dal tempo senza considerare le dipendenze che possono esistere tra le variabili. Al contrario, una tecnica multivariabile non può essere utilizzata se i dati in ingresso sono una serie temporale univariabile.

2.2 Definizione di anomalie

Le anomalie, spesso indicate come outlier, eventi rari o devianti, sono punti dati o modelli nei dati che non si conformano a una nozione di comportamento normale.

La rilevazione delle anomalie è quindi il compito di trovare quei modelli nei dati che non aderiscono alle norme previste, date le osservazioni precedenti. La capacità di riconoscere o rilevare comportamenti anomali può fornire informazioni estremamente utili in diversi settori. Segnalare casi insoliti o attuare una risposta pianificata quando si verificano può risparmiare tempo, costi e clienti alle aziende. Pertanto, la rilevazione delle anomalie ha trovato diverse applicazioni in una varietà di settori, tra cui l'analisi IT, l'analisi delle intrusioni di rete, la diagnostica medica, la protezione dalle frodi finanziarie, il controllo della qualità della produzione, l'analisi del marketing e dei social media e altro ancora.

Esistono diverse tecniche per il rilevamento di anomalie, che si basano su diversi presupposti e modelli.

Alcune delle tecniche più comuni sono:

- Scomposizione delle serie temporali: questa tecnica consiste nel dividere una serie temporale in tre componenti: tendenza, stagionalità e residuo. La tendenza rappresenta la direzione generale della serie nel tempo, la stagionalità rappresenta i pattern ciclici o periodici e il residuo rappresenta la parte casuale o irregolare. Le anomalie vengono rilevate confrontando il residuo con dei limiti superiori e inferiori calcolati in base a una soglia di sensibilità.
- Modelli statistici: questa tecnica consiste nell'adattare un modello statistico ai dati delle serie temporali e per poi calcolare la probabilità che un nuovo valore sia generato dal modello. Se la probabilità è molto bassa, il valore viene considerato anomalo. Alcuni esempi di modelli statistici sono i modelli ARIMA (Auto-Regressive Integrated Moving Average) o i modelli di media mobile.
- Machine learning: questa tecnica consiste nell'usare algoritmi di apprendimento automatico per apprendere le caratteristiche normali delle serie temporali e poi rilevare le deviazioni da esse. Alcuni esempi di algoritmi sono le reti neurali, gli alberi decisionali, le macchine a vettori di supporto o gli algoritmi basati sulla distanza.

Queste sono solo alcune delle possibili tecniche per il rilevamento di anomalie. Ogni tecnica ha dei vantaggi e svantaggi a seconda del tipo della complessità dei dati, degli obiettivi e dei vincoli dell'analisi. Per scegliere la tecnica più adatta, è importante capire bene il contesto e il dominio applicativo del problema.

In letteratura le anomalie nelle serie temporali possono essere ampiamente classificate in tre categorie: anomalie puntuali, anomalie contestuali e anomalie collettive.

- L'anomalia più semplice e comune nell'area delle serie temporali è l'anomalia puntiforme. Le anomalie puntiformi sono punti che deviano dalla maggioranza degli altri campioni e spesso rappresentano un'irregolarità o una deviazione che accade casualmente e può non avere una particolare interpretazione. Un approccio ingenuo per una data serie temporale univariabile può essere quello di considerare come anomalia ogni punto x_t con distanza dal suo valore atteso maggiore di una soglia predefinita: $|x_t - \widehat{x}_t| > \tau$ dove x_t è il valore osservato e $\widehat{x}_t$ è il suo valore atteso.

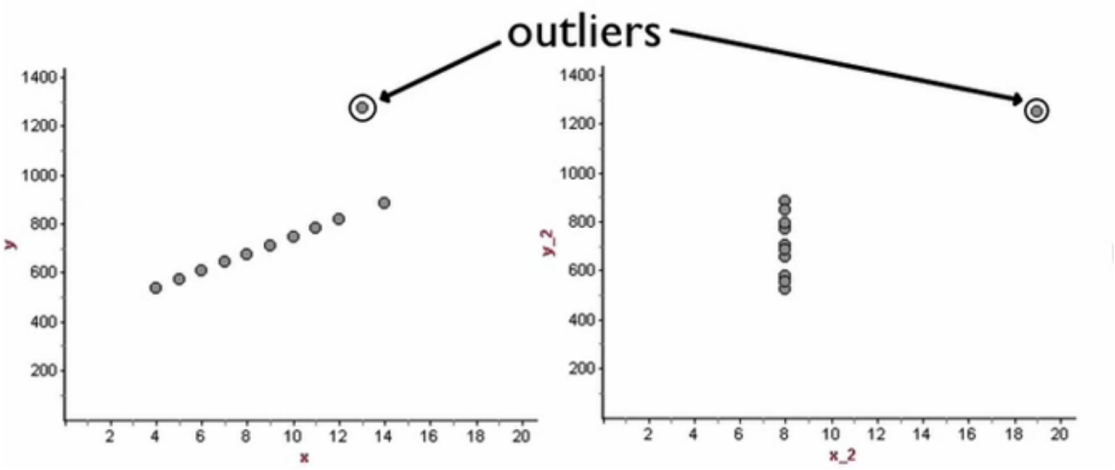

Figura 2: Esempio di anomalia puntiforme

- Un' anomalia contestuale è un'istanza di dati che potrebbe essere considerata anomala solo in un contesto specifico. Queste anomalie sono le più difficili da riconoscere perché dobbiamo addestrare il modello in modo che sia in grado di catturare anche il contesto di ogni timestamp.
- Le raccolte anomale di singoli punti dati sono note come anomalie collettive o di gruppo, in cui ogni punto appare individualmente normale mentre osservato in un gruppo mostra caratteristiche insolite. Inoltre, queste sequenze possono essere sottosequenze periodiche di anomalie che si ripetono nel tempo.

Negli ultimi anni, è risultato necessario avere strumenti per analizzare tali dati, apprendere i modelli e rilevare autonomamente le anomalie. Per questo motivo, i metodi di rilevamento delle anomalie Deep (DAD) hanno dimostrato di rilevare tutti e tre i tipi di anomalie con grande successo. I modelli di deep learning possono rilevare anomalie sia in dati univariabili che multivariabili, che l'anomalia

colpisca un singolo sensore o più variabili dipendenti dal tempo. In altre parole, nel caso di serie temporali multivariabili, possiamo trovare anomalie in una o più caratteristiche.

In questo contesto si dovranno studiare e implementare algoritmi di machine learning e deep learning che possano analizzare i dati provenienti da diverse fonti e fornire soluzioni ottimali per i problemi di interesse. Gli algoritmi di machine learning dovranno essere in grado di apprendere da dati non strutturati o parzialmente strutturati, di adattarsi a scenari dinamici e incerti, e di integrare conoscenze pregresse o esperte. Inoltre, si dovranno valutare le prestazioni degli algoritmi di machine learning in termini di accuratezza, robustezza, efficienza e interpretabilità, e confrontarle.

2.3 Rilevamento di anomalie con machine learning

La rilevazione delle anomalie è un compito binario in cui un campione può essere classificato normale o anomalo. Le anomalie sono rare e nella maggior parte dei casi è difficile ottenere le loro etichette perché dobbiamo rompere manualmente la macchina e quindi registrare i campioni.

Nel caso di modelli di machine learning ci sono vari approcci di rilevazione delle anomalie. Gli approcci di rilevazione delle anomalie possono essere categorizzati in base al tipo di dati necessari per addestrare il modello. Nella maggior parte dei casi d'uso, ci si aspetta che i campioni anomali rappresentino una percentuale molto piccola dell'intero dataset. Pertanto, anche quando sono disponibili dati etichettati, i campioni di dati normali sono più facilmente disponibili dei campioni anomali. Questa assunzione è fondamentale per la maggior parte delle applicazioni odierna.

In base ai dati e alle etichette disponibili, è possibile dividere gli algoritmi in tre categorie: apprendimento con supervisione, apprendimento non supervisionato e apprendimento semi-supervisionato.

2.3.1 Apprendimento supervisionato

Nell'apprendimento con supervisione, le macchine apprendono una funzione che mappa le caratteristiche di input in output basati su coppie di input-output di esempio. L'obiettivo degli algoritmi di rilevazione delle anomalie supervisionate è quello di incorporare la conoscenza specifica dell'applicazione nel processo di rilevazione delle anomalie. Con sufficienti esempi normali e anomali, il compito di

rilevazione delle anomalie può essere riformulato come un compito di classificazione in cui le macchine possono imparare a prevedere con precisione se un dato esempio è un'anomalia o meno. Detto questo, per molti casi d'uso di rilevazione delle anomalie la proporzione tra esempi normali e anomali è altamente sbilanciata; mentre potrebbero esserci più classi anomale, ciascuna di esse potrebbe essere piuttosto sottorappresentata. Questo approccio assume che si abbiano esempi etichettati per tutti i tipi di anomalie che potrebbero verificarsi e che si possano classificare correttamente. Nella pratica, questo non è solitamente il caso, poiché le anomalie possono assumere molte forme diverse, con nuove anomalie emergenti al momento del test. Pertanto, gli approcci che generalizzano bene e sono più efficaci nell'identificare anomalie precedentemente non viste sono preferibili.

2.3.2 Apprendimento non supervisionato

Con l'apprendimento non supervisionato, le macchine non possiedono coppie di input-output di esempio che consentono di apprendere una funzione che mappa le caratteristiche di input in output. Invece, apprendono trovando la struttura all'interno delle caratteristiche di input. Poiché, come già detto in precedenza, i dati anomali etichettati sono relativamente rari, gli approcci non supervisionati sono più popolari di quelli supervisionati nel campo della rilevazione delle anomalie. Detto questo, la natura delle anomalie che si spera di rilevare è spesso altamente specifica. Pertanto, molte delle anomalie trovate in modo completamente non supervisionato potrebbero corrispondere a rumore e potrebbero non essere interessanti per il compito in questione. Questo approccio ibrido è ben adatto ad applicazioni come la rilevazione delle intrusioni di rete, dove si possono avere più esempi della classe normale e alcuni esempi di classi di intrusioni, ma nuovi tipi di intrusioni possono sorgere nel tempo.

2.3.3 Apprendimento semi-supervisionato

Gli approcci di apprendimento semi-supervisionato rappresentano una sorta di via di mezzo, impiegando un insieme di metodi che sfruttano grandi quantità di dati non etichettati e piccole quantità di dati etichettati. Molti casi d'uso reali di rilevazione delle anomalie sono ben adatti all'apprendimento semi-supervisionato, poiché ci sono un gran numero di esempi normali disponibili da cui imparare, ma relativamente pochi esempi delle classi più insolite o anomale di interesse. Seguendo l'assunzione che la maggior parte dei punti dati all'interno di un dataset non etichettato sia normale, si può addestrare un modello robusto su un dataset

non etichettato e valutarne le prestazioni (e regolare i parametri del modello) utilizzando una piccola quantità di dati etichettati.

3 Stato dell'arte

Nella letteratura attuale, un approccio comune e ampiamente utilizzato per la rilevazione di anomalie consiste nel trovare una funzione di decisione che definisca il modello di normalità. In questo approccio, si definisce innanzitutto una certa funzione di decisione e poi si ottimizzano i parametri di questa funzione rispetto a un criterio obiettivo predefinito, esempi di algoritmi sono:

- Rilevamento di anomalie basato su clustering: questo metodo raggruppa i punti dati in base alla loro somiglianza e considera come anomalie quelli che appartengono a cluster piccoli o isolati. [1]
- Isolation Forests: questo metodo costruisce alberi di decisione casuali per isolare i punti dati e assegna uno score di anomalia in base al numero di split necessari per raggiungere il punto.
- Support Vector Machine: questo metodo cerca di separare i punti dati normali da quelli anomali con un iperpiano ottimale, utilizzando una funzione kernel per mappare i dati in uno spazio di dimensione superiore.
- Rilevamento di anomalie utilizzando la distribuzione gaussiana: questo metodo assume che i dati normali seguano una distribuzione gaussiana e calcola la probabilità che un punto appartenga a tale distribuzione. I punti con bassa probabilità sono considerati anomali.

Tuttavia, gli algoritmi basati su questo approccio esaminano i dati delle serie temporali su una finestra temporale sufficientemente lunga per ottenere una prestazione accettabile. Pertanto, le loro prestazioni dipendono significativamente dalla lunghezza di questa finestra temporale in modo che questo approccio richieda una selezione attenta della lunghezza della finestra temporale per fornire una prestazione soddisfacente. Per migliorare le prestazioni dei dati delle serie temporali, sono stati introdotti il kernel Fisher e i modelli generativi. Tuttavia, il principale svantaggio del modello kernel Fisher è che richiede l'inversione della matrice delle informazioni Fisher, che ha una complessità computazionale elevata.[2] D'altra parte, per ottenere una prestazione adeguata da un modello generativo come un modello nascosto di Markov (HMM), si dovrebbero selezionare attentamente i suoi parametri strutturali, ad esempio il numero di stati e

la topologia del modello. Inoltre, il tipo di algoritmo di addestramento ha anche effetti considerevoli sulle prestazioni dei modelli generativi, il che limita la loro utilizzo nelle applicazioni reali. Pertanto, sono stati introdotti le reti neurali, in particolare gli approcci basati su reti neurali ricorrenti (RNN), grazie alla loro struttura di memoria intrinseca che può memorizzare informazioni "temporali" o "di stato". [3]

Alcuni esempi di questi metodi sono:

- Rilevamento delle anomalie basato sulla decomposizione del segnale: questo metodo scompone la serie temporale in componenti diverse, come trend, stagionalità e residuo, e applica tecniche di rilevamento delle anomalie su ciascuna componente. Alcuni algoritmi per la decomposizione del segnale sono la scomposizione classica e STL (Seasonal and Trend decomposition using Loess).
- Modelli nello spazio degli stati: questi modelli descrivono l'evoluzione della serie temporale come una funzione dello stato interno del sistema e dell'input esterno. Alcuni algoritmi per modellare lo spazio degli stati sono il livellamento esponenziale, Holt Winters e ARIMA (AutoRegressive Integrated Moving Average).
- Deep learning: questi metodi utilizzano reti neurali profonde per apprendere una rappresentazione della serie temporale e rilevare le deviazioni da essa. Alcuni tipi di reti neurali usate per questo scopo sono i codificatori automatici basati su feedforward, che comprimono e ricostruiscono i dati con uno strato nascosto a bassa dimensionalità, e le reti neurali ricorrenti e LSTM (Long Short-Term Memory), che memorizzano informazioni sul passato con delle unità di memoria. [4] [5]
- Riduzione della dimensionalità: questi metodi riducono la complessità dei dati trasformandoli in uno spazio a dimensione inferiore, dove è più facile individuare le anomalie. Alcuni algoritmi per la riduzione della dimensionalità sono RPCA (Robust Principal Component Analysis), SOM (Self-Organizing Maps), discord (subsequence anomaly detection) e lineare a tratti (piecewise linear approximation).

Tuttavia, poiché l'architettura RNN di base non ha strutture di controllo (gate) per regolare la quantità di informazioni da memorizzare, viene introdotta un'architettura RNN più avanzata con diverse strutture di controllo, ovvero la rete LSTM (long short-term memory). Tuttavia, gli approcci basati su reti neurali non possono ottimizzare direttamente un criterio obiettivo per la rilevazione delle

anomalie a causa della mancanza di etichette dei dati in un framework non supervisionato. Pertanto, prima prevedono una sequenza dai suoi campioni passati e poi determinano se la sequenza è un'anomalia o meno in base all'errore di previsione, ovvero un'anomalia è un evento che non può essere previsto dai dati nominali passati. Di conseguenza, richiedono un modello probabilistico per l'errore di previsione e una soglia sul modello probabilistico per rilevare le anomalie, il che comporta problemi di ottimizzazione impegnativi e limita di conseguenza le loro prestazioni. Inoltre, sia gli approcci comuni che quelli basati su reti neurali possono elaborare solo sequenze di vettori a lunghezza fissa, il che limita significativamente il loro utilizzo nelle applicazioni reali. [6] [7]

4 Flusso di lavoro

Un sistema di rilevamento delle anomalie è utile per monitorare il funzionamento di una macchina e segnalare eventuali deviazioni dalla norma. Per costruire un tale sistema, dobbiamo seguire alcuni passaggi:

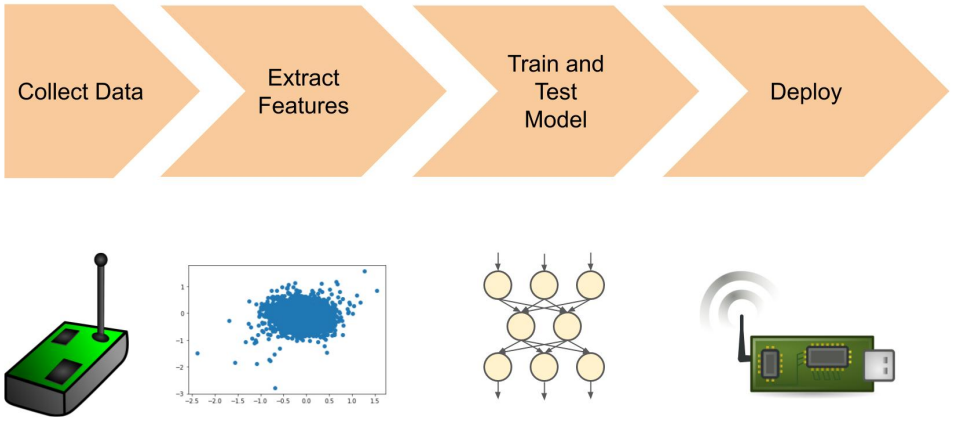

Figura 3: Flusso di lavoro per la creazione di un sistema di rilevamento anomalie

- Il primo passo è raccogliere i dati che rappresentano il comportamento normale della macchina. Questi dati possono essere misurazioni di vari sensori, parametri di controllo o altri indicatori di prestazione. Se possibile,

è bene raccogliere anche dati che simulano un'anomalia, per poter valutare la capacità del sistema di riconoscerla.

- Il secondo passo è estrarre le caratteristiche dai dati che possano essere utili per distinguere tra normalità e anomalia. Queste caratteristiche possono essere statistiche descrittive, trasformate spettrali, misure di complessità o altro. L'obiettivo è ridurre la dimensionalità dei dati e catturare le informazioni rilevanti.
- Il terzo passo è addestrare un modello di machine learning che possa classificare i dati in normali o anomali. Il modello deve essere testato con dati normali e anomali (se presenti) per verificare la sua accuratezza e robustezza.
- Il quarto passo è implementare il sistema finale che possa applicare il modello ai dati in tempo reale e generare degli allarmi in caso di anomalie.

5 Raccolta dei dati

Il dataset che abbiamo generato si basa sui dati raccolti da una scheda prototipo stm32l476g-disco, che è una piattaforma integrata per lo sviluppo di applicazioni su microcontrollori STM32L4. La scheda dispone di vari sensori, tra cui un accelerometro che abbiamo usato per rilevare le anomalie nelle vibrazioni. L'accelerometro favorisce misure sui assi X,Y,Z in accelerazione di gravità G, quindi con $1G = 9,8\ m/s^2$.

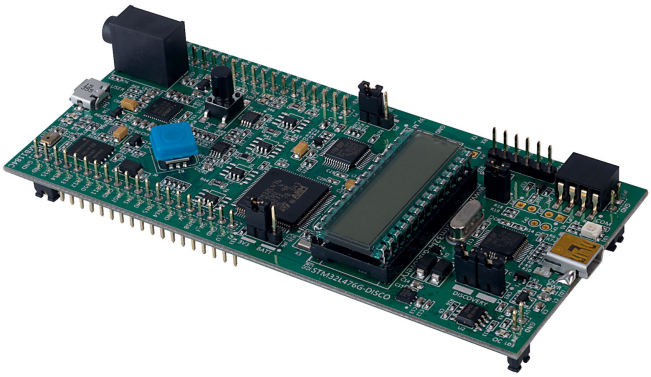

Figura 4: Scheda stm32l476g-disco utilizzata come prototipo

Abbiamo creato un dataset con misurazioni da 200 campioni, considerato come campioni normali quelli in cui la scheda era ferma o soggetta a vibrazioni minime, e li abbiamo salvati in vari file csv. Come campioni anomali, abbiamo registrato le vibrazioni brusche o risonanti che possono verificarsi in caso di malfunzionamenti o urti. I dispositivi su cui verrà integrato il datalogger possono avere molteplici malfunzionamenti. In questo caso, abbiamo ipotizzato un motore elettrico come dispositivo che potrebbe presentare l'usura della meccanica interna e quindi un incremento notevole delle vibrazioni. Il nostro obiettivo è di utilizzare il dataset per allenare un modello di apprendimento automatico in grado di classificare le vibrazioni e riconoscere le anomalie.

6 Estrazione caratteristiche dei dati

Nella maggior parte dei sistemi di apprendimento automatico, non possiamo o non vogliamo inviare direttamente i dati grezzi al nostro modello. Alcuni modelli, come le reti neurali, possono essere addestrati per estrarre le caratteristiche, ma spesso comporta un aumento della complessità computazionale. Se possiamo determinare quali caratteristiche ci consentiranno di rilevare le anomalie (o

prevedere i valori o classificare i casi) più facilmente, risparmieremo molti sforzi di formazione e potenza di elaborazione in futuro.

Una “caratteristica” può essere quasi qualsiasi cosa che viene estratta dai dati grezzi. Potrebbe essere il dato grezzo stesso, combinazioni di diversi dati del sensore, analisi statistica su diverse misurazioni (media, varianza, ecc.), O trasformazioni, come la trasformata di Fourier nel dominio delle frequenze. Una volta scelte una o più caratteristiche, possiamo quindi addestrare un modello utilizzando quelle caratteristiche. Da quel momento in poi, ogni volta che vogliamo utilizzare il modello per fare previsioni/classificazioni, dobbiamo estrarre le stesse caratteristiche dai nuovi dati.

Ci sono molti algoritmi che possono essere utilizzati per automatizzare il processo di estrazione delle caratteristiche e ridurre il numero di dimensioni che entrano in un modello di apprendimento automatico. Per le immagini, potrebbe trattarsi di qualcosa come la rilevazione dei bordi. Per i suoni, potrebbe trattarsi di una trasformata di Fourier veloce (FFT). Gli algoritmi, come il clustering k-means e l’analisi delle componenti principali (PCA), possono aiutare a determinare raggruppamenti e ridurre le dimensioni dei dati.

Un modo per esplorare i dati è calcolare alcune statistiche descrittive per ogni set di campioni, come la media, la varianza, la curtosi e skew. Queste statistiche ci possono dare un’idea della forma e della dispersione dei dati. Possiamo visualizzare queste statistiche in grafici a dispersione, dove ogni punto rappresenta un set di campioni.

Di seguito sono mostrati come esempi un grafico e scatter di un set di dati a cui di seguito sono state applicate numerose operazioni per individuare le caratteristiche.

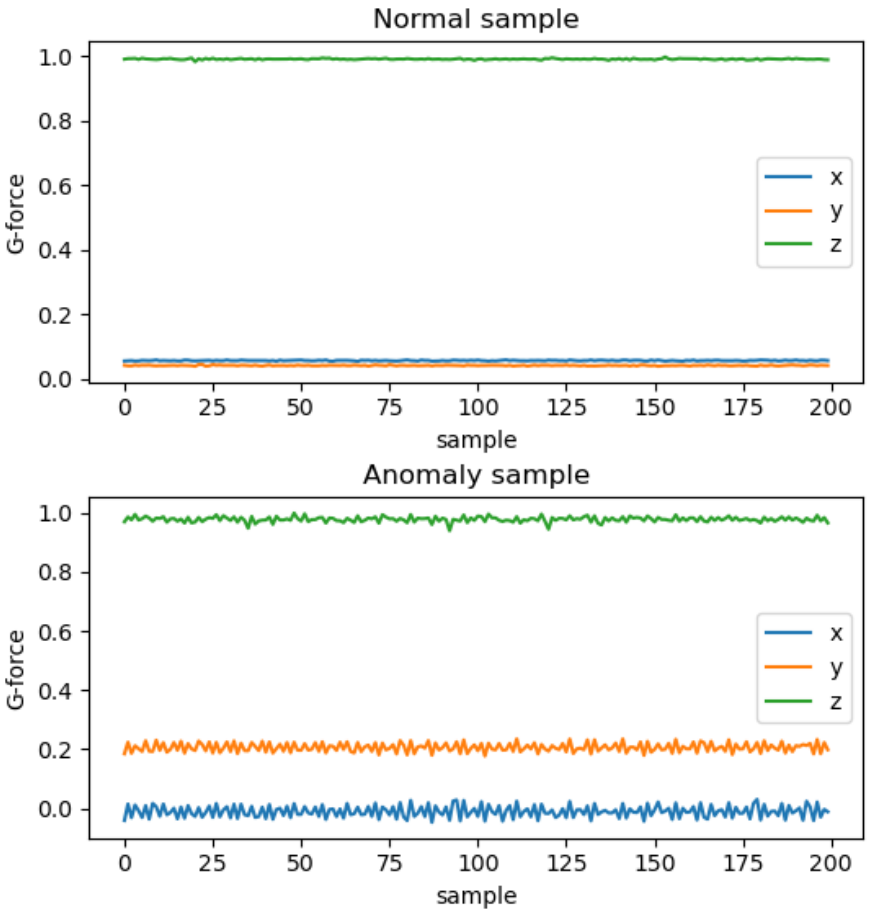

Figura 5: Grafico delle accelerazioni gravitazionali di un set di dati normali, in basso contenenti anomalie

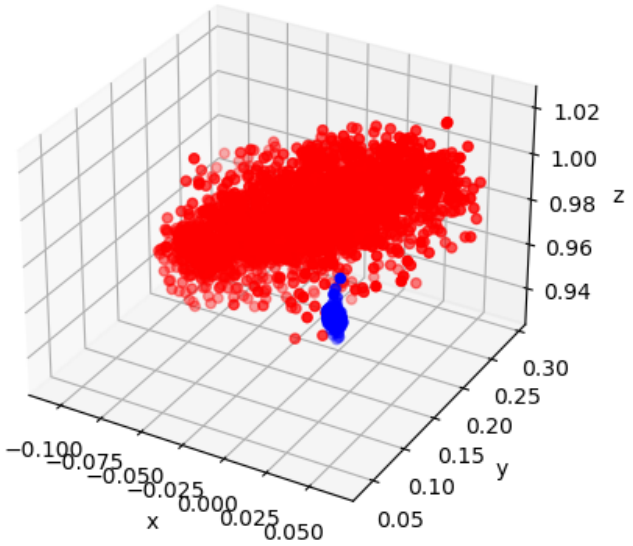

Figura 6: Scatter delle accelererazioni di un set di dati normali in blu e anomalie in rosso

I punti blu corrispondono ai dati normali, mentre i punti rossi corrispondono ai dati anomali.

Per analizzare i dati di vibrazione e trovare caratteristiche utili al nostro modello, dobbiamo eliminare la componente costante che potrebbe essere dovuta all'orientamento o al movimento dell'accelerometro e cercare operazioni in modo che ci permettano di visualizzare immediatamente un insieme contenente tutti i punti normali, ben distinto da un insieme contenente le anomalie. Per fare questo, calcoliamo la media di ogni set e la sottraiamo da ogni valore in quel set. In questo modo, ci concentriamo solo sulle fluttuazioni attorno alla media mostrate nelle immagini seguenti

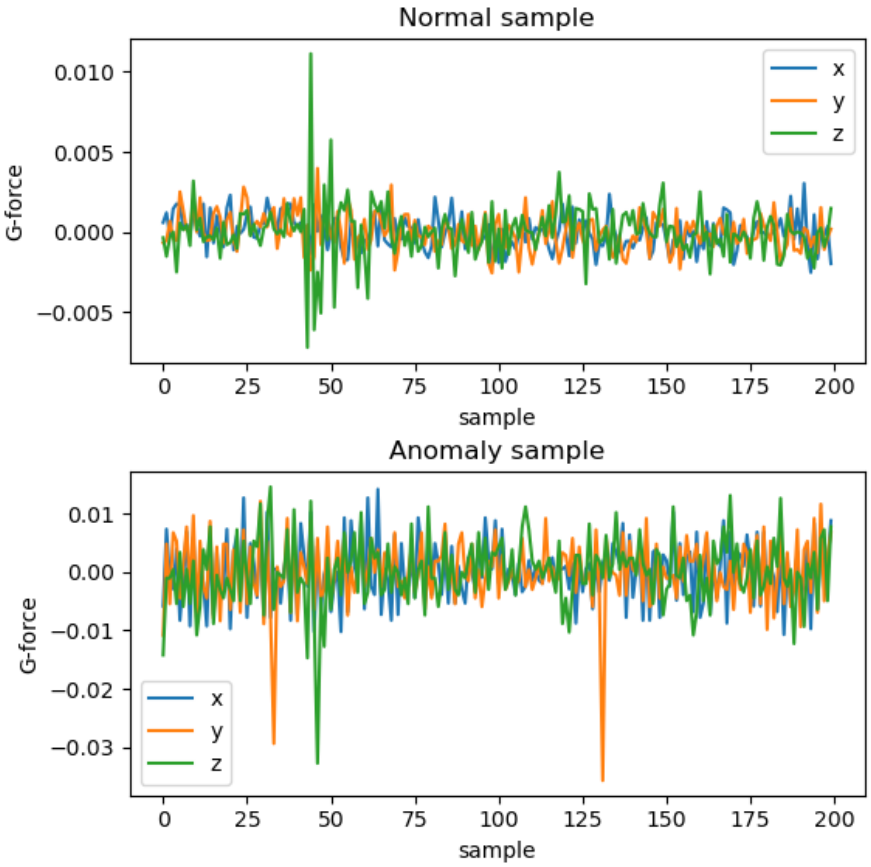

Figura 7: Grafico delle accelerazioni gravitazionali di un set di dati normali senza componenti in continua, in basso contenenti anomalie

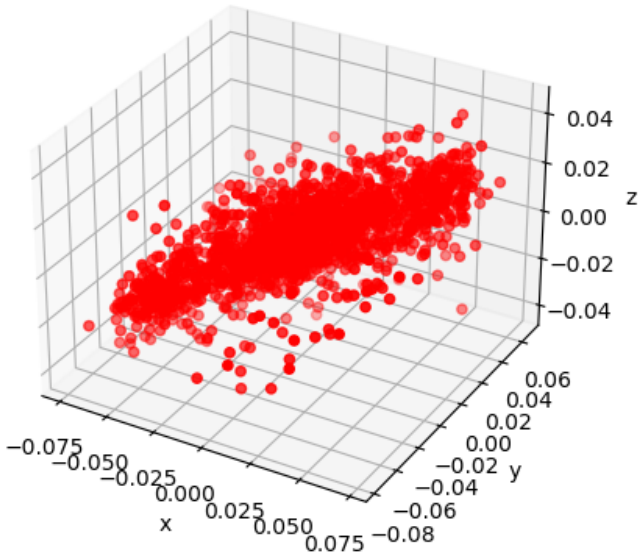

Figura 8: Scatter delle accelererazioni di un set di dati normali senza componenti in continua. normali in blu e anomalie in rosso

Dallo scatter si può notare che non è possibile distinguere facilmente un insieme contenenti tutti e soli i punti di dati normali, quindi è necessario ricercare delle statistiche descrittive adeguate.

6.1 Medie

Le medie, che dovrebbero essere vicine a zero dopo aver sottratto la media da ogni set, sono mostrate nel grafico sottostante per ogni asse dell'accelerometro.

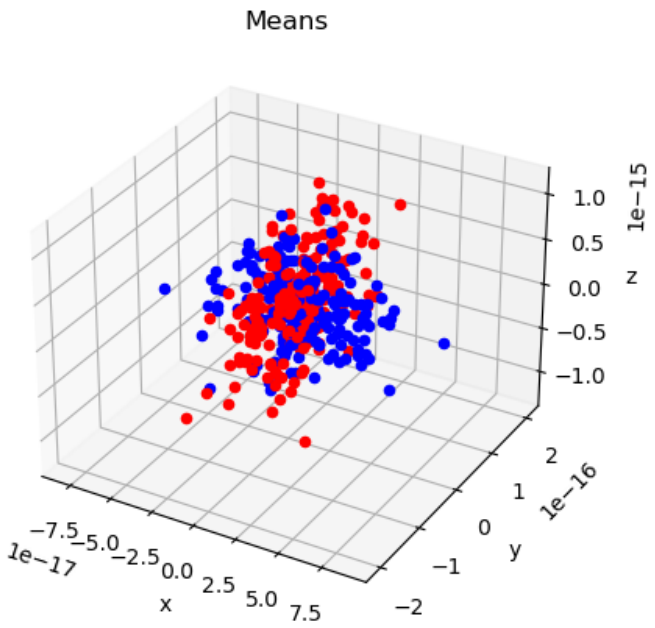

Figura 9: Scatter delle medie di un set di dati normali e contenenti anomalie

Utilizzando le medie non è ancora possibile distinguere facilmente un insieme contenenti tutti e soli i punti di dati normali.

6.2 Varianza

La varianza è un indice statistico che misura il grado di dispersione dei valori di una variabile quantitativa attorno alla media aritmetica. La varianza si calcola come la media dei quadrati degli scarti dalla media, ovvero la somma dei quadrati delle differenze tra ogni valore e la media, divisa per il numero totale di valori. La varianza è sempre positiva o nulla, e ha le stesse unità di misura della variabile al quadrato.

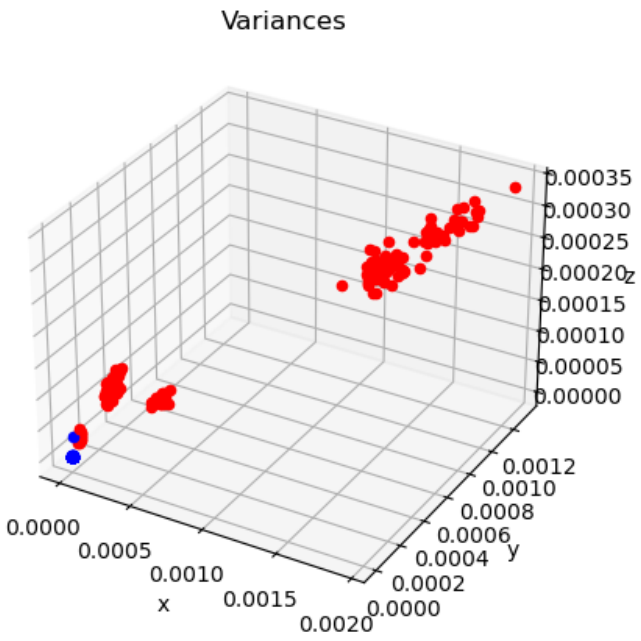

Figura 10: Scatter delle varianze di un set di dati normali e contenenti anomalie

Utilizzando le varianze si riesce a distinguere degli insiemi adeguati che separano dati normali e dati anomali ma non facilmente distinguibili tra loro.

6.3 Curtosis

La curtosi è una misura della forma di una distribuzione di probabilità che indica quanto si discosta dalla normalità distributiva. Una distribuzione normale ha una curtosi pari a zero, mentre una distribuzione che ha code più spesse o più appuntite di una normale ha una curtosi positiva. Al contrario, una distribuzione che ha code più sottili o più piatte di una normale ha una curtosi negativa. La curtosi si calcola come il rapporto tra il momento centrato di ordine quattro e il quadrato della varianza . Questa formula è anche nota come indice di Pearson. La curtosi è utile per descrivere il comportamento delle code di una distribuzione e il rischio di eventi estremi.

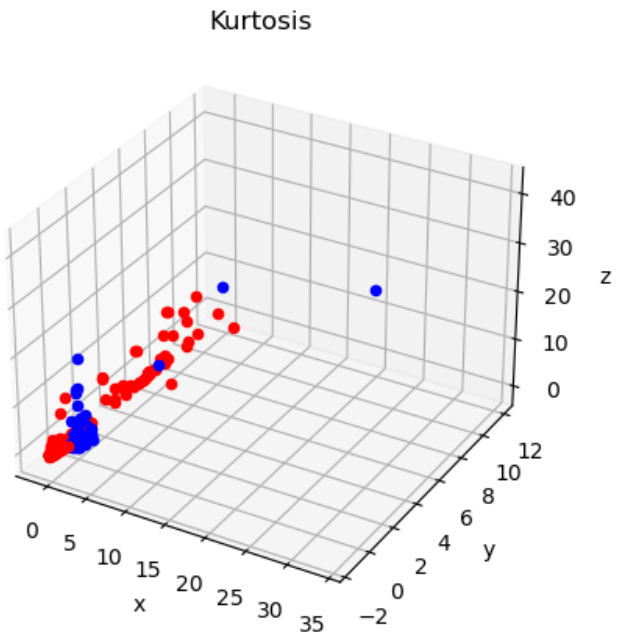

Figura 11: Scatter delle curtosi di un set di dati normali e contenenti anomalie

Utilizzando la curtosis non è possibile distinguere facilmente un insieme contenenti tutti e soli i punti di dati normali.

6.4 Skew

Lo skew caratterizza il grado di asimmetria di una distribuzione intorno alla sua media. L'asimmetria positiva indica una distribuzione con una coda asimmetrica che si estende verso i valori più positivi. L'asimmetria negativa indica una distribuzione con una coda asimmetrica che si estende verso i valori più negativi.

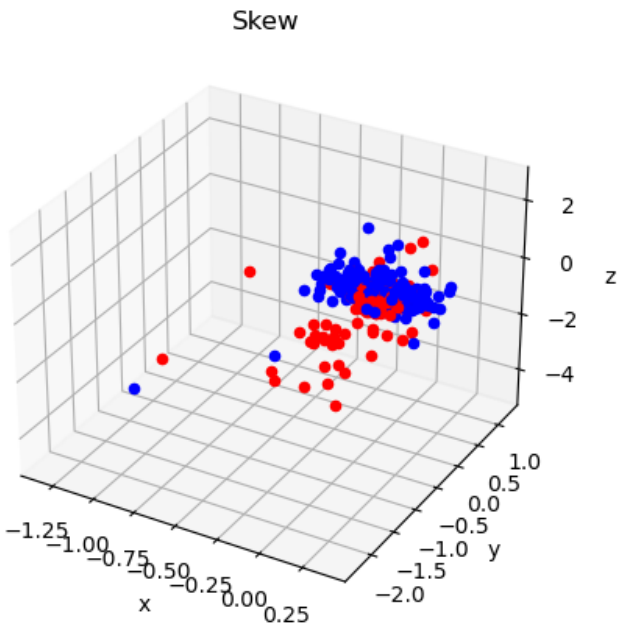

Figura 12: Scatter delle skew di un set di dati normali e contenenti anomalie

Utilizzando la skew non è possibile distinguere facilmente un insieme contenenti tutti e soli i punti di dati normali.

6.5 Deviazione media assoluta

La deviazione media assoluta o Median Absolute Deviation (MAD) è una misura robusta della variabilità di un campione univariato di dati quantitativi. Mentre la deviazione standard (e la varianza) sono ottime per descrivere la dispersione dei dati in una distribuzione normale, possono essere facilmente influenzati da valori anomali e dati non normalmente distribuiti. Di conseguenza, MAD offre un modo più robusto per misurare la dispersione dei dati non normali. Poiché spesso si trovano dati che non sono normalmente distribuiti, MAD può essere la migliore caratteristica per misurare la dispersione.

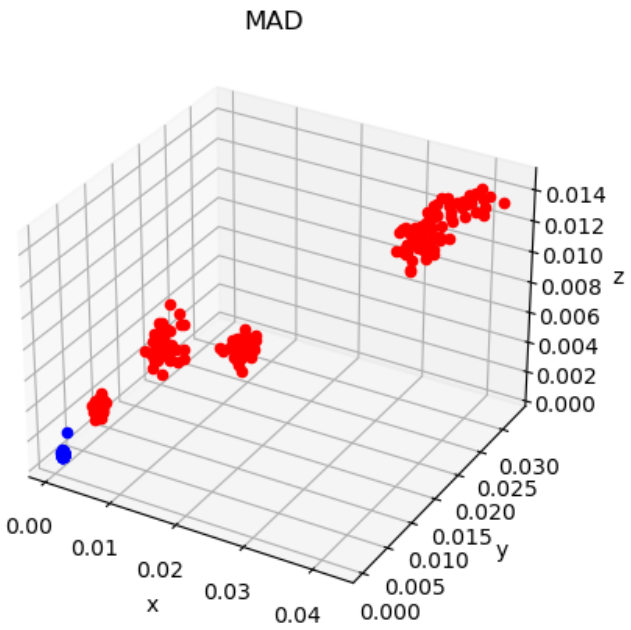

Figura 13: Scatter delle distanze medie assolute di un set di dati normali e contenenti anomalie

Utilizzando la deviazione media assoluta si ha una chiara distinzione tra insieme di dati normali e anomalie, quindi può essere una caratteristica efficace per individuare le anomalie in questo scenario cercando i punti che superano la soglia determinato dall'insieme dei dati normali.

6.6 Trasformata di Fourier

La trasformata di Fourier è uno strumento matematico che permette di analizzare i dati in termini di frequenza. Si basa sul principio che ogni funzione periodica può essere scomposta in una somma infinita di onde sinusoidali di diverse frequenze e ampiezze. Queste onde sinusoidali sono chiamate armoniche e costituiscono lo spettro della funzione.

La trasformata di Fourier consente di passare dal dominio del tempo, in cui la funzione è espressa come una variazione nel tempo, al dominio della frequenza, in cui la funzione è espressa come una combinazione lineare di armoniche. Questo passaggio è utile per studiare le proprietà dei dati, come la presenza di rumore, di picchi o di periodicità.

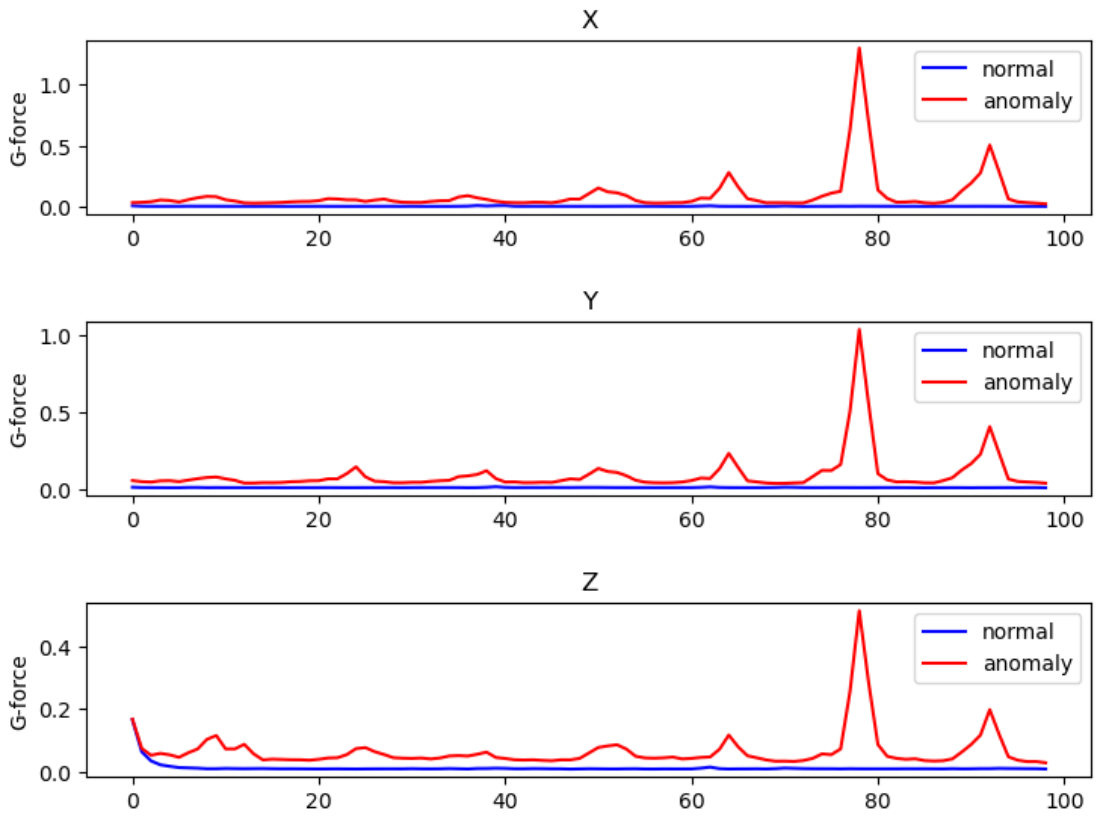

Figura 14: Grafico delle trasformate di fourier di un set di dati normali e contenenti anomalie

Anche attraverso la trasformata di fourier è possibile individuare efficacemente anomalie controllando se viene superata una certa soglia ad alcune componenti frequenziali.

In seguito è stato deciso di troncare il numero di campioni nelle misurazioni a 128 per garantire l'omogeneità nelle successive operazioni.

7 Scelta e addestramento modelli

Basandosi sulle analisi effettuate nelle sezioni precedenti, si è deciso di focalizzare l'attenzione su modelli di machine learning basati sulla deviazione media assoluta perché risulata una metrica molto efficace per la distinzione di misurazioni anomali e normali. I modelli sono tre: uno che verifica il superamento di una soglia con le misure delle distanze di Mahalanobis, una rete neurale che utilizza le distanze come input e una rete neurale più grande che utilizza direttamente i dati grezzi.

7.1 Distanza Mahalanobis

La distanza Mahalanobis è una misura di distanza tra un punto e una distribuzione, introdotta da P. C. Mahalanobis nel 1936. Essa è basata sulle correzioni tra variabili attraverso le quali differenti pattern possono essere identificati ed analizzati. Si tratta di un'utile maniera per determinare similarità di uno spazio campionario incognito rispetto ad uno noto.

La distanza Mahalanobis si differenzia dalla distanza euclidea in quanto tiene conto delle correlazioni all'interno dell'insieme dei dati. In altre parole, essa considera la forma della distribuzione dei dati e non solo la loro distanza da un punto. La distanza di Mahalanobis è quindi priva di unità di misura, invariante per cambiamenti di scala e dipendente dalla matrice di covarianza dei dati.

La formula della distanza di Mahalanobis è la seguente:

$$D_M(x) = \sqrt{(x-\mu)^T \mathbf{\Sigma}^{-1} (x-\mu)}$$

dove x è il vettore del punto da misurare, μ è il vettore della media della distribuzione e Σ è la matrice di covarianza della distribuzione.

La distanza Mahalanobis ha diverse applicazioni in statistica, come il riconoscimento di pattern, l'analisi delle componenti principali, la classificazione dei cluster, la rilevazione degli outlier e il controllo della qualità.

Per ogni campione (composto da diverse misurazioni ciascuno), calcoliamo la MAD sull'asse X, la MAD sull'asse Y e la MAD sull'asse Z. La distanza di Mahalanobis tiene conto delle componenti principali, il che ci consente di disegnare un'ellisse come confine dei dati invece di un cerchio. Quando calcoliamo la distanza di Mahalanobis, otteniamo un singolo numero, quindi è facile confrontarlo con una soglia. Qualsiasi cosa sopra quella soglia è un'anomalia e qualsiasi cosa uguale o inferiore ad essa è considerata normale.

Il modello creato deve leggere dal dataset e trovare le matrici di media e covarianza dei campioni “normali”. Una volta che abbiamo la matrice media e di covarianza, possiamo quindi scrivere una funzione che calcola la distanza di Mahalanobis tra la media e un nuovo punto.

Possiamo testare la sua efficacia calcolando la distanza di Mahalanobis con tutti i campioni normali e anomali e tracciamo gli istogrammi delle loro distanze (il blu rappresenta le distanze normali, il rosso le distanze anomale). Come si può vedere, c’è una chiara separazione tra i raggruppamenti rossi e blu.

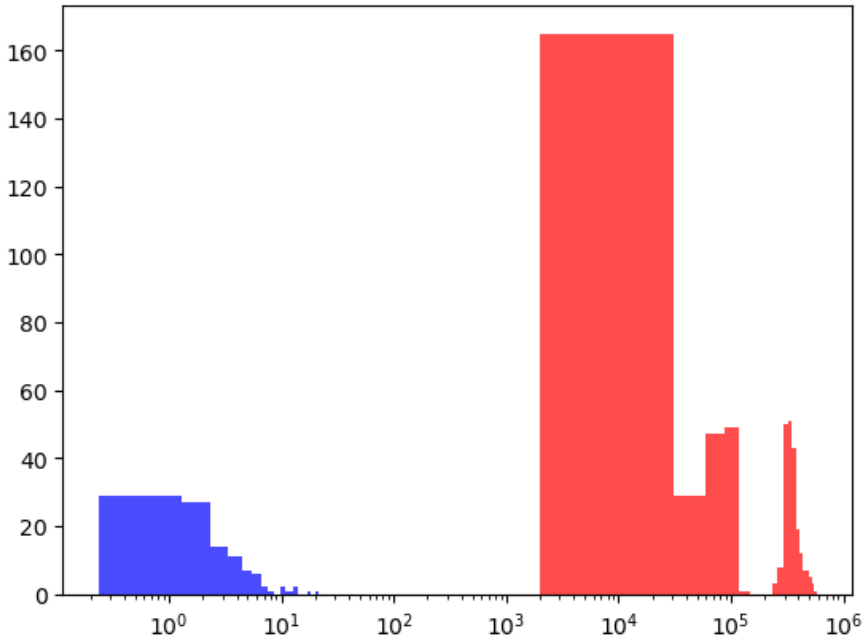

Figura 15: Grafico dei valori delle distanze Mahalanobis di un set di dati normali e contenenti anomalie

Utilizzando questo, possiamo creare una semplice soglia. Qualsiasi nuovo campione la cui distanza di Mahalanobis è superiore a 20 è un “anomalia” e tutto il resto è “normale”. Possiamo utilizzare il nostro set di test che abbiamo messo da parte all’inizio del programma per generare questa matrice di confusione:

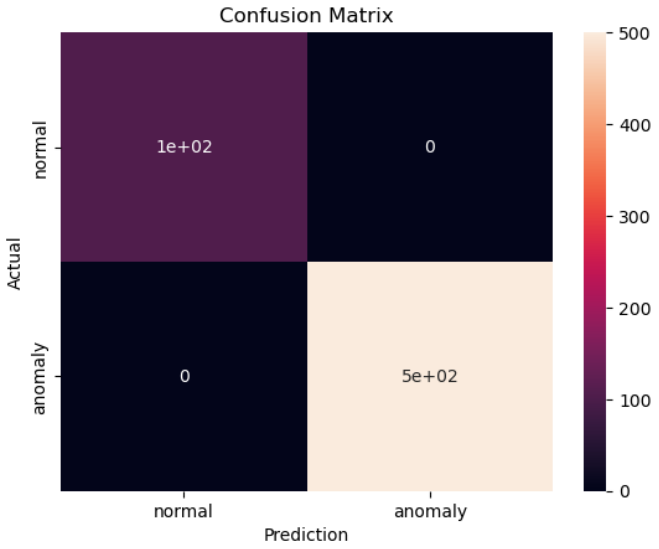

Figura 16: Matrice di confusione del modello con distanza mahalanobis

Per distribuire il modello basterà utilizzare le matrici di media e covarianza calcolate scritte in un file. Si noti che questo modello descrive solo questo singolo particolare caso di studio. Avremmo bisogno di molti più dati per caratterizzare altri scenari o vorremmo creare un modello unico per ogni.

7.2 Rete neurale di tipo autoencoder

Un autoencoder è un tipo di rete neurale utilizzata nell'apprendimento non supervisionato in cui la rete è composta da sottoreti di codifica e decodifica. La sottorete di codifica forza una rappresentazione compressa dell'input in dimensioni più piccole e la sottorete di decodifica tenta di ricreare l'input dalla versione compressa fornita dalla sottorete di codifica. Gli autoencoder vengono applicati a molti problemi, dal riconoscimento facciale alla rilevazione delle caratteristiche e alla riduzione del rumore dei dati. Rappresentano i dati all'interno di più strati nascosti ricostruendo i dati di input, apprendendo efficacemente una funzione identità. Quando addestrati esclusivamente su istanze di dati normali, non riescono a ricostruire i campioni di dati anomali producendo un grande errore di ricostruzione. Questi punti associati ad un alto errore residuo sono considerati anomalie. La scelta dell'architettura dell'autoencoder dipende dalla natura dei dati, le reti convoluzionali sono preferite per i dataset di immagini mentre i modelli basati su memoria a lungo termine (LSTM) sono in grado di catturare la dipendenza temporale nei dati sequenziali. La profondità di un autoencoder

dipende dalla dimensione dei dati di input. Più dimensioni ci sono, più strati sono necessari per estrarre tutte le informazioni rilevanti durante l'addestramento. Il tipo di apprendimento è non supervisionato perché il modello non richiede alcuna informazione sulle etichette, rendendolo molto popolare e ampiamente utilizzato nella letteratura.

Uno dei motivi per cui gli autoencoder hanno attirato così tanti ricercatori e attenzione è che sono stati a lungo considerati una potenziale via per risolvere i problemi senza la necessità di etichette. Gli autoencoder non sono una tecnica di apprendimento veramente non supervisionata (che implicherebbe un processo di apprendimento completamente diverso), ma una tecnica auto-supervisionata, un'istanza specifica di apprendimento supervisionato in cui i target sono generati dai dati di input. Sebbene gli autoencoder siano architetture semplici ed efficaci per la rilevazione degli outlier, le prestazioni possono essere degradate a causa dei dati di addestramento rumorosi.

Il codice seguente mostra come creare un autoencoder con il pacchetto TensorFlow, l'autoencoder ha tre neuroni di input che accettano i valori delle distanze medie assolute per ogni asse dell'accelerometro, uno strato denso di due neuroni, un dropout di due neuroni ed uno strato denso di tre neuroni. Questa architettura è stata ricavata tramite un processo di ottimizzazione.

```
encoding_dim = 2        # Number of nodes in first layer
model = models.Sequential([
    layers.InputLayer(input_shape=sample_shape),
    layers.Dense(encoding_dim, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(*sample_shape, activation='relu')
])
```

Figura 17: Codice generazione rete neurale di tipo autoencoder

Per utilizzare un autoencoder per la rilevazione di anomalie, addestriamo l'autoencoder solo sui campioni normali. Se fatto correttamente, l'errore di ricostruzione per qualsiasi nuovo campione normale dovrebbe essere basso, poiché l'autoencoder dovrebbe essere in grado di capire le relazioni e le caratteristiche necessarie per riprodurre gli stessi valori MAD di input come i valori MAD di input.

Tuttavia, se vengono forniti valori MAD anomali, l'autoencoder dovrebbe avere difficoltà a riprodurre gli stessi valori, risultando in un errore di ricostruzione più elevato tra l'input e l'output. Con il giusto modello di rete neurale e addestramento,

dovremmo vedere valori di errore bassi per campioni normali e valori di errore elevati per anomalie.

Per l'addestramento della rete, è stato scelto l'errore quadratico medio come metrica di errore. Vista questa scelta, lo stesso errore è stato utilizzato anche per la rivelazione delle anomalie con il calcolo dell'errore quadratico medio tra l'ingresso e l'uscita della rete.

Se rappresentiamo l'errore quadratico medio o Mean Square Error (MSE) dei normali rispetto a quelli anomali, dovremmo vedere una certa separazione, che indica che l'autoencoder può ricreare i campioni MAD normali più facilmente rispetto ai campioni MAD anomali.

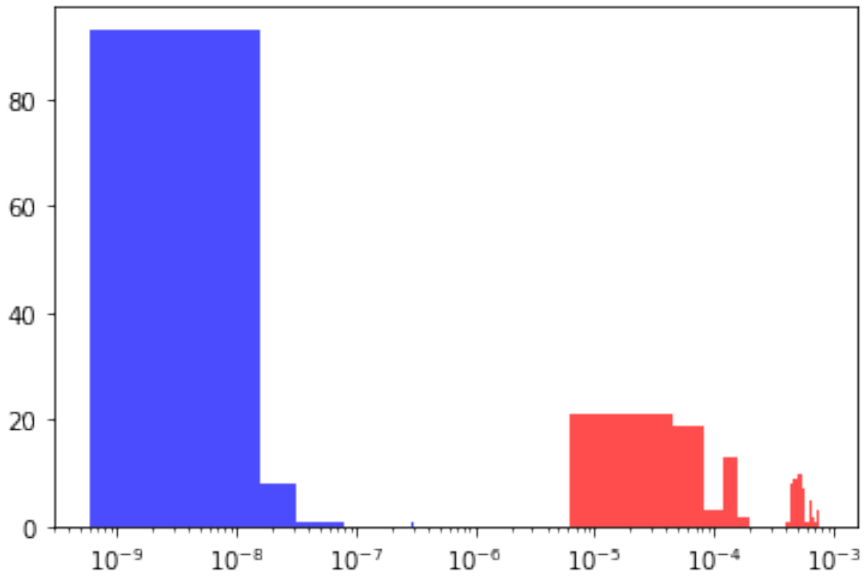

Figura 18: Grafico degli errori quadratici medi su validation set

Come nella distanza di Mahalanobis, possiamo creare un semplice classificatore da utilizzare sul nostro set di test e creare una matrice di confusione per determinare quanto bene il nostro autoencoder funziona nella rilevazione di anomalie.

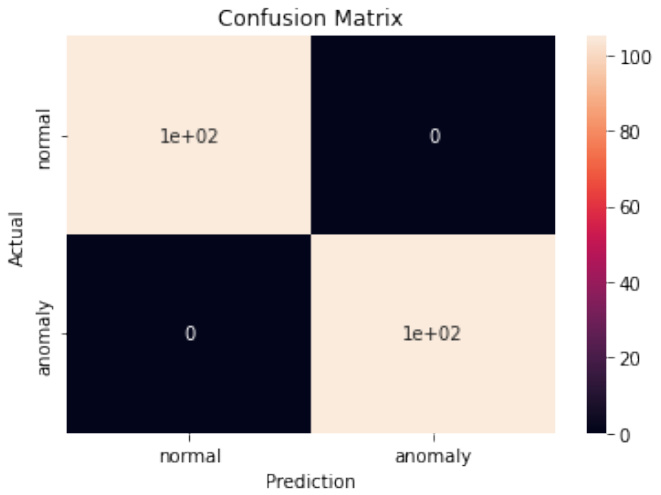

Figura 19: Matrice di confusione del modello autoencoder

7.3 Rete neurale LSTM

Le reti LSTM (Long Short-Term Memory) sono un tipo di reti neurali ricorrenti (RNN) che possono apprendere e memorizzare relazioni e dipendenze a lungo termine tra i dati sequenziali. Le RNN sono reti neurali artificiali che hanno connessioni di feedback, cioè possono usare l'output precedente come input successivo. Questa proprietà consente alle RNN di elaborare intere sequenze di dati, come testo, parlato o serie temporali, senza trattare ogni punto della sequenza in modo indipendente.

Tuttavia, le RNN tradizionali soffrono di problemi che causano il non mantenere una memoria a lungo termine delle informazioni precedenti nella sequenza, e quindi perdono il contesto necessario per elaborare i nuovi dati.

Le reti LSTM sono state progettate per superare questi problemi, introducendo una struttura speciale chiamata cella di memoria. Una cella di memoria è costruita da tre porte: una porta di input e una porta di output. Queste porte sono delle piccole reti neurali che decidono quali informazioni conservare, aggiungere o rimuovere dallo stato della cella, che è la memoria a lungo termine della rete. In questo modo, le reti LSTM possono regolare il flusso delle informazioni nella sequenza e preservare le dipendenze a lungo termine.

Le reti LSTM sono in grado di catturare modelli complessi e sottili nei dati sequenziali e di generare output coerenti e pertinenti.

Per questo modello, abbiamo usato i valori grezzi come input per la rete, raggruppati in lotti di trentadue. L'immagine seguente mostra il codice che definisce la struttura della rete:

```
model = Sequential()
model.add(InputLayer(input_shape=(x_train.shape[1], x_train.shape[2])))
model.add(LSTM(32, activation='relu', return_sequences=True))
model.add(LSTM(16, activation='relu', return_sequences=False))
model.add(RepeatVector(x_train.shape[1]))   # Repeats the inputs time_samples times
model.add(LSTM(16, activation='relu', return_sequences=True))
model.add(LSTM(32, activation='relu', return_sequences=True))
model.add(TimeDistributed(Dense(x_train.shape[2]))) # Get the output
```

Figura 20: Codice generazione rete neurale LSTM di tipo autoencoder

Anche in questo caso rappresentiamo l'errore quadratico medio normale rispetto a quelli anomali e vediamo una chiara separazione.

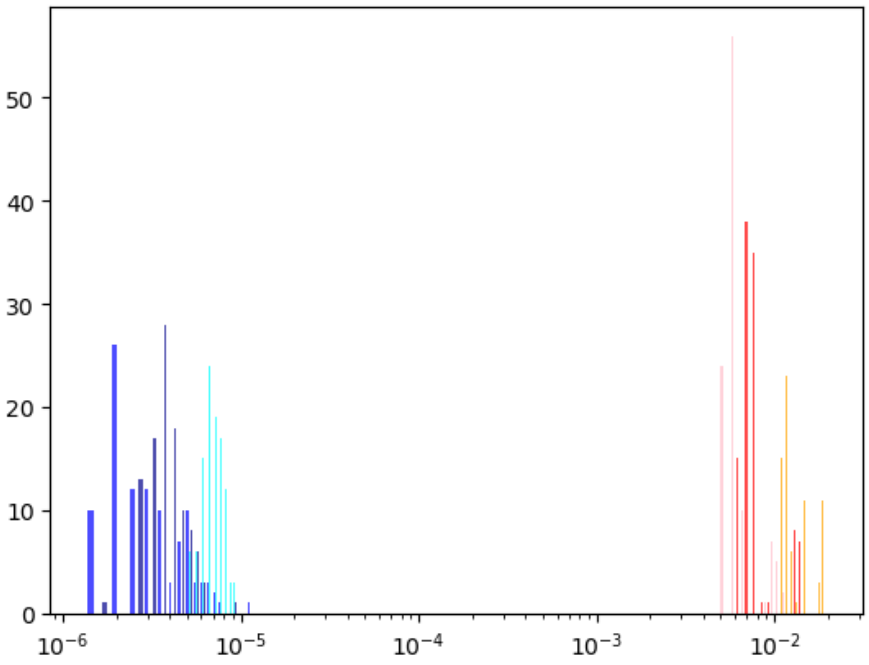

Figura 21: Grafico degli errori quadratici medi su validation set

Utilizzando il classificatore sul nostro set di test e creando una matrice di confusione ricaviamo la bontà del nostro modello nella rilevazione di anomalie.

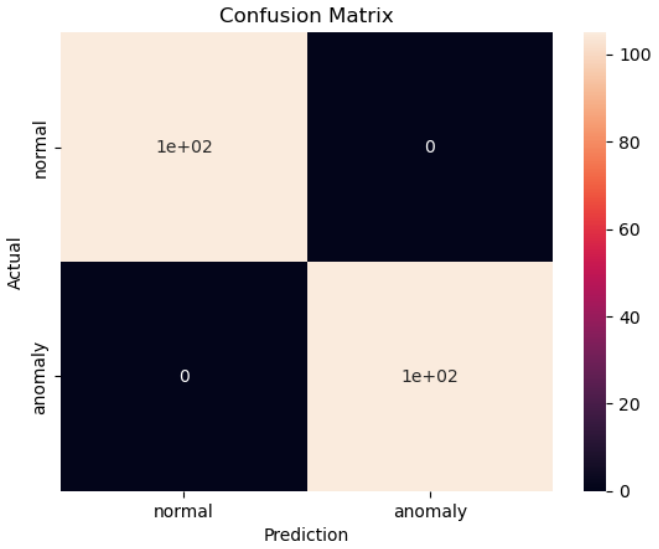

Figura 22: Matrice di confusione del modello LSTM autoencoder

8 Implementazione su stm32

Poiché non abbiamo accesso a molte delle librerie di alto livello in C, come Numpy, SciPy e TensorFlow, abbiamo utilizzato una conversione che permettesse l'implementazione dei modelli nel microcontrollore strm32.

Nel caso del modello della distanza Mahalanobis è sufficiente importare nel codice C le matrici di media e covarianza calcolate precedentemente, implementare le funzioni per il calcolo della distanza Mahalanobis, della deviazione media assoluta, e dell'errore quadratico medio, illustrate nelle immagini in appendice 11.

Ogni funzione creata è stata utilizzata con un campione di dati e comparata con le funzioni usate precedentemente in Python. In questo modo si è ricavato un errore di calcolo che, una volta implementato nel micro, corregge i calcoli e li rende identici a quelli effettuati precedentemente in python.

Per i modelli di reti neurali è stato utilizzato il pacchetto STM Cube AI per convertire i file di TensorFlow in file C ottimizzati e integrati nel micro STM32, in particolare, è stata fatta una conversione da modelli Keras a modelli TensorFlow lite poi convertiti in C.

Il software STM Cube consente di ottenere un indice di complessità computazionale dei modelli di reti neurali in termini di MAC (multiply accumulate). Il modello

autoencoder ha una complessità di 22 MAC, mentre la LSTM ha una complessità di 522336 MAC. La differenza è data dal grande numero di parametri nella rete LSTM rispetto all'autoencoder. Inoltre, per la rilevazione delle anomalie per l'autoencoder, è necessario calcolare anche le MAD che non sono tenute in conto dall'indice precedente.

Le funzioni principali che implementano i modelli quindi prendono in ingresso i dati in real-time dal sensore accelerometro e rilevano la presenza di anomalie tramite l'accensione di un led.

I risultati della prova hanno dimostrato che il modello della distanza Mahalanobis ha un minor consumo energetico e computazionale rispetto agli altri due modelli, ma con meno precisione nella rilevazione pratica di alcune anomalie. Il modello dell'autoencoder si è mostrato il più bilanciato in termini di memoria occupata (simile al modello Mahalanobis), basso consumo energetico e computazionale e alta affidabilità nelle rilevazioni. Il modello LSTM è risultato capace di rilevare più anomalie contestuali ma ad un costo alto energetico, computazionale e di memoria. Inoltre, è importante sottolineare che i risultati ottenuti sono stati validati su un set di dati specifico e potrebbero non essere generalizzabili ad altri contesti.

9 Conclusioni e sviluppi futuri

In questo documento si descrivono i risultati ottenuti applicando diversi algoritmi di machine learning per l'identificazione di anomalie in serie temporali real time. Dopo aver esaminato vari approcci e tecniche per il problema, si sono selezionati alcuni metodi da implementare e confrontare. Si è creato un dataset con dati normali e anomali, e si sono esplorate le sue proprietà.

Si sono sviluppati tre modelli: uno basato sulla distanza Mahalanobis, una rete neurale autoencoder e una LSTM.

I modelli hanno dimostrato di essere efficaci nei casi di studio scelti.

In futuro si collezioneranno altri dati e si sperimentaranno modelli simili in diversi contesti dove sarà impiegato il datalogger finale.

I modelli saranno integrati come task in un sistema operativo real time.

10 Riferimenti bibliografici

[1] Nico Görnitz, Luiz Alberto Lima, Klaus-Robert Müller, Marius Kloft, and Shinichi Nakajima. Support vector data descriptions and k-means clustering: one class? *IEEE transactions on neural networks and learning systems*, 29 (9):3994–4006, 2017. 3

[2] Jiaping Zhao and Laurent Itti. Classifying time series using local descriptors with hybrid sampling. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):623–637, 2015. 3

[3] Zahra Zamanzadeh Darban, Geoffrey I Webb, Shirui Pan, Charu C Aggarwal, and Mahsa Salehi. Deep learning for time series anomaly detection: A survey. *arXiv preprint arXiv:2211.05244*, 2022. 3

[4] Oleksandr I Provotar, Yaroslav M Linder, and Maksym M Veres. Unsupervised anomaly detection in time series using lstm-based autoencoders. In *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*, pages 513–517. IEEE, 2019. 3

[5] Tolga Ergen and Suleyman Serdar Kozat. Unsupervised anomaly detection with lstm neural networks. *IEEE transactions on neural networks and learning systems*, 31(8):3127–3141, 2019. 3

[6] Alessio Siciliano. *Deep learning models for anomaly detection in time series*. PhD thesis, Politecnico di Torino, 2021. 3

[7] Jiuqi Elise Zhang, Di Wu, and Benoit Boulet. Time series anomaly detection via reinforcement learning-based model selection. In *2022 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 193–199. IEEE, 2022. 3

[8] Venelin Valkov. Credit card fraud detection using autoencoders in keras–tensorflow for hackers (part vii), 2017.

[9] Zsigmond Benkő, Tamás Bábel, and Zoltán Somogyvári. Model-free detection of unique events in time series. *Scientific Reports*, 12(1):227, 2022.

[10] Chaoli Zhang, Tian Zhou, Qingsong Wen, and Liang Sun. Tfad: A decomposition time series anomaly detection architecture with time-frequency analysis. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 2497–2507, 2022.

11 Appendice

```
// Find median in a group of numbers
// WARNING: sorts array in place
float median(float *arr, int arrLen) {

  float median;

  // Use stdlib's qsort to sort array in place
  qsort(arr, arrLen, sizeof(float), compare_floats);

  // If even number of elements take average of two middle elements
  if (arrLen % 2 == 0) {
    median = (arr[(arrLen - 1) / 2] + arr[arrLen / 2]) / 2.0;
  } else {
    median = arr[arrLen / 2];
  }

  return median;
}
```

Figura 23: Codice implementato nel stm32 per il calcolo della media

```
// Calculate Median Absolute Deviation (MAD) of array
// WARNING: manipulates array in place
float calcMad(float *arr, int arrLen) {

  // Get the median of the array
  float med = median(arr, arrLen);

  // Calculate absolute deviation from the median for each element
  for (int i = 0; i < arrLen; i++) {
    arr[i] = fabs(arr[i] - med);
  }

  // Find the median of the deviations
  return median(arr, arrLen);
}
```

Figura 24: Codice implementato nel stm32 per il calcolo della deviazione media assoluta

```
// Calculate the Mahalanobis distance based on mean and inverse covariance
float mahalanobis(const float *x, const float *mu,
                  const float *invCov, int len) {

  float xMinusMu[1][len];
  float xMinusMuT[len][1];
  float leftTerm[1][len];
  float mahal[1][1];

  // Subtract each element in X from the mean
  for (int i = 0; i < len; i++) {
    xMinusMu[0][i] = x[i] - mu[i];
  }

  // Compute product of prev term and inverse covariance
  matrix_multiply(*xMinusMu, invCov, 1, len, len, len, *leftTerm);

  // Transpose matrix
  for (int i = 0; i < len; i++) {
    xMinusMuT[i][0] = xMinusMu[0][i];
  }

  // Matrix multiply prev term and difference
  matrix_multiply(*leftTerm, *xMinusMuT, 1, len, len, 1, *mahal);

  return mahal[0][0];
}
```

Figura 25: Codice implementato nel stm32 per il calcolo della distanza mahalanobis

```
// Calculate the mean squared error between two arrays
float calcMse(const float *x, const float *xHat, const int len) {

  float mse = 0;

  // Square difference between each set of elements
  for (int i = 0; i < len; i++) {
    mse += pow(x[i] - xHat[i], 2);
  }

  return mse / len;
}
```

Figura 26: Codice implementato nel stm32 per il calcolo del errore quadratico medio