
UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

---

Corso di Laurea Magistrale

in Ingegneria Informatica e dell'Automazione

# Studio e implementazione di un modulo diagnostico data-driven per il rilevamento di danni alle eliche di un velivolo multirotore

## Study and development of a data-driven fault detection module for propeller damage in multirotor vehicles

Relatore:
**Prof. Alessandro Freddi**

Tesi di laurea di:
**Michele Graziano de Virgilio**

Correlatori:
**Prof. Adriano Mancini**
**Dr. Jack Dyson**

---

Anno Accademico 2023/2024

# Sommario


Introduzione ..... 3

1. Manutenzione preventiva applicata ai droni ..... 5

1.1 Concetti preliminari ..... 5

1.1.1 Supervisione di processo ..... 5

1.1.2 Terminologie della diagnosi ..... 7

1.1.3 Classificazione della diagnosi ..... 8

1.2 I droni e l'importanza della fault detection ..... 9

1.3 Guasti sui droni ..... 10

1.4 Rilevamento guasti ..... 11

2. Descrizione del dataset e preprocessamento dati ..... 13

3. Estrazione feature ..... 15

3.1 Diagnostic feature designer ..... 15

3.2 Importazione dataTable e generazione feature ..... 15

3.3 Preparazione dati ..... 18

4 Diagnosi basata su classificazione ..... 20

4.1 Estrazione codici dal CL ..... 20

4.2 Addestramento all features ..... 21

4.3 Addestramento classificatori con top 14 feature ..... 22

4.4 Test con features vibrazioni ..... 24

4.5 Holdout randomici ..... 25

4.6 Test variazione FR, FS ..... 25

5 Analisi in tempo-frequenza ..... 29

5.1 Estrazione dati aggiuntivi ..... 29

5.2 Spettrogramma ..... 32

5.2.1 Fourier Transform ..... 32

5.2.2 Discrete Time Fourier Transform (DTFT) ..... 33

5.2.3 Fast Fourier Transform (FFT) ..... 33

5.2.4 Short Time Fourier Transform (STFT) ..... 33

5.2.5 funzione Spectrogram e settaggio dei parametri ..... 34

5.3 Scalogramma ..... 34

5.3.1 Continuous wavelet transform ..... 35

5.3.2 Funzione CWT e settaggio dei parametri ..... 36

5.4 Segnali di interesse ..... 36

5.5 Confronto Spectrogram – CWT con accelerazioni e traiettorie ..... 38

5.6 Media potenze spettrogrammi ..... 40
5.7 Addestramento classificatori ..... 43
5.7.1 Volo intero ..... 43
5.7.2 Spectral binning ..... 44
5.7.3 Partizionamento voli ..... 48
Conclusioni e sviluppi futuri ..... 50
Appendice ..... 51
A.1 - Plot confronti spectrogram – CWT ..... 51
Bibliografia ..... 60

# Introduzione

In questo lavoro di tesi sono state testate e messe a confronto varie strategie di diagnosi guasti su droni. Garantire la sicurezza diventa fondamentale data la diffusione esponenziale che stanno avendo negli ultimi anni e nei campi più disparati.

La sfida che verrà affrontata è quella di riuscire a fare rilevamento e isolamento guasti, mediante tecniche data-driven, addestrando classificatori che riescano a riconoscere le classi di guasto.

Successivamente si utilizzeranno tecniche signal-based, analizzando le singole frequenze di segnali selezionati, cercando di estrarre feature nel dominio congiunto tempo-frequenza, costruire un algoritmo di classificazione in base alla tipologia di fault, tratte da opportuni spettrogrammi. L'idea è di andare a suddividere eventualmente i dati dei voli a disposizione, in porzioni più piccole, così da aumentare la dimensione del dataset per poi fare classificazione. Il workflow del lavoro svolto è riassunto graficamente in *Fig. 1*.

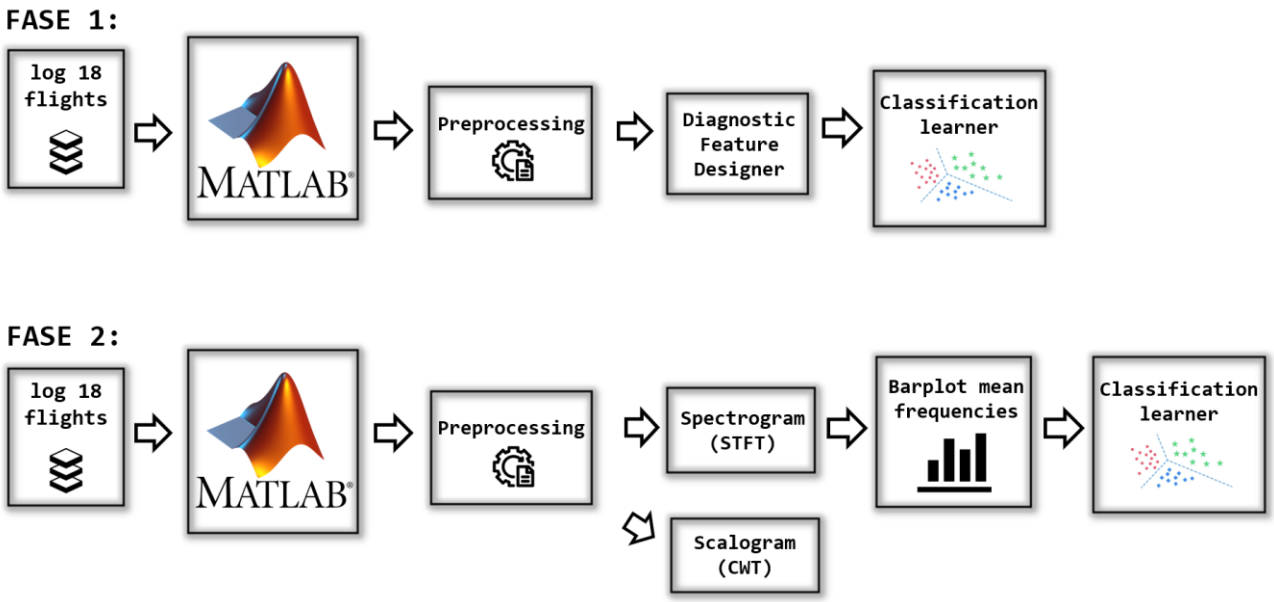


*Figura 1: Workflow del lavoro svolto*

La struttura è così formata, con il lavoro diviso in cinque capitoli:

Nel *primo capitolo*, vengono trattati alcuni concetti preliminari nell'ambito della manutenzione preventiva e nell'ambito della diagnosi guasti. Successivamente è stata posta l'attenzione sui droni e sull'importanza di fare rilevamento guasti su di essi, enunciando i vari campi di impiego in cui sono utilizzati, per poi passare alle tecniche di rilevamento guasti e ai loro vantaggi e svantaggi.

Nel *secondo capitolo*, viene descritto il dataset utilizzato, le modalità di sincronizzazione dati ed estrazione delle variabili, con l'aggiunta delle etichette relative alla tipologia di fault.

Il *terzo capitolo* fa una breve panoramica sul Diagnostic Feature Designer di Matlab, per poi parlare di come importare il dataset, le modifiche al codice realizzate e la preparazione del dataset per essere classificato.

Il *quarto capitolo* contiene la parte relativa alla diagnosi basata su classificazione, inserimento dei dati processati nel Classification Learner e vari test in base a differenti casi, per riportare i risultati sotto forma di tabella. Inoltre contiene dei test che agiscono sul *Frame Rate* e *Frame Size* del campionamento del volo, per valutare l'impatto di questa scelta.

Il *quinto capitolo* contiene la parte relativa all'analisi tempo-frequenza. Nella prima parte c'è una breve digressione sulla geodesia e i passi per giungere a determinate formule di conversione di latitudine e longitudine che verranno utilizzate, dopo aver estratto queste variabili dai dati grezzi, per poi continuare con l'utilizzo della STFT e trasformata wavelet continua per analizzare opportuni segnali selezionati. Questo capitolo si conclude con risultati relativi a test ulteriori, sotto opportune considerazioni che si è costretti a fare data la scarsità del dataset.

# 1. Manutenzione preventiva applicata ai droni

In questo primo capitolo si tratterà il tema della manutenzione preventiva, la sua importanza per i sistemi e alcune definizioni chiave note in letteratura, propedeutiche agli argomenti trattati, tratte da [1].

## 1.1 Concetti preliminari

### 1.1.1 Supervisione di processo

I processi tecnologici moderni divengono sempre più complessi e integrati, con diversi sistemi di controllo i quali sono organizzati su livelli differenti. Ci si porrà al di sopra del classico livello di controllo di processo, in un livello detto di "supervisione".

La supervisione dei processi tecnici ha l'obiettivo di mostrare lo stato attuale, indicare stati indesiderati o non consentiti e adottare le azioni appropriate per evitare danni o incidenti. Le deviazioni dal normale comportamento dei processi derivano da guasti ed errori, che possono essere attribuiti a molteplici cause. Questi possono comportare malfunzionamenti o guasti per periodi più o meno lunghi, se non vengono prese contromisure. Uno dei motivi principali per fare supervisione è prevenire tali malfunzionamenti o guasti [1].

Gli strumenti di supervisione spesso integrano sistemi SCADA (Supervisory Control and Data Acquisition) e algoritmi di machine learning per identificare trend e intervenire prima che si manifestino guasti o degrado sul sistema.

Nelle sezioni seguenti vengono descritti brevemente i compiti fondamentali della supervisione. Successivamente, viene fatto un approfondimento su alcuni termini e sulla terminologia utilizzata in questo campo.

Si può affermare, in sintesi, che la supervisione è l'azione volta a garantire che l'automazione si svolga nel modo desiderato [1].

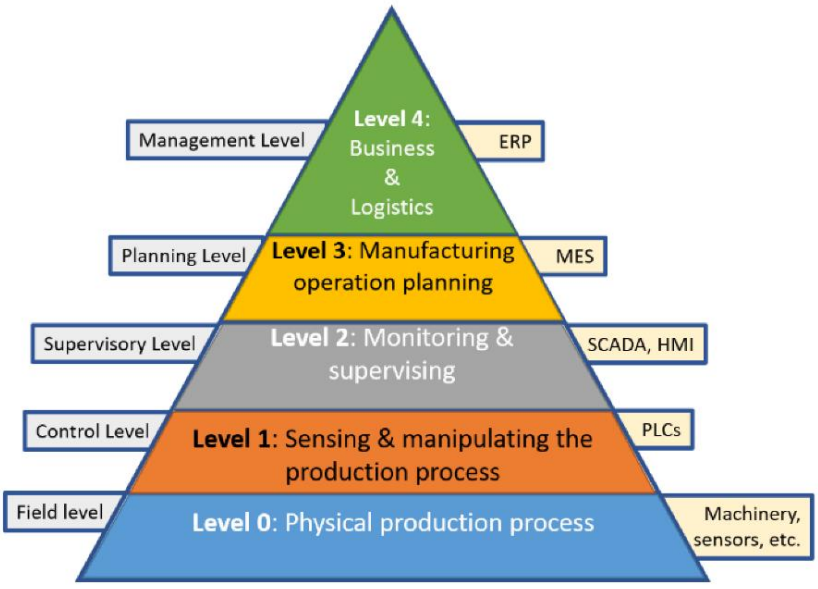


*Figura 1.1: Piramide dell'automazione [11]*

Nella *Fig. 1.1* viene riportata la piramide dell'automazione, che schematizza un impianto automatizzato suddividendolo in vari livelli. Il livello di interesse per questo lavoro di tesi è il livello 2, che si pone sopra il livello di controllo e il livello di campo.

La supervisione cresce di pari passo con l'automazione: viene richiesta sempre meno la presenza dell'operatore umano dal controllo di processo, quindi si ha la necessità che la supervisione divenga sempre più efficiente e sicura [1].

È possibile dividere la supervisione in due categorie, supervisione **classica** e **avanzata**.

- La supervisione **classica** si basa su politiche di "limit checking", analizzando variabili di processo: vengono generati allarmi in caso di deviazioni significative. Questo metodo è apprezzato per la sua semplicità, ma risulta limitato nella capacità di rilevare guasti minori o graduali, specialmente in presenza di sistemi complessi o operanti in condizioni transitorie. In tali contesti, possono sussistere ritardi nell'individuazione o ci può essere indeterminabilità per il guasto.
- La supervisione **avanzata** utilizza modelli analitici e metodi di rilevamento più sofisticati per identificare anomalie. Questo approccio viene realizzato combinando più tecniche: l'analisi dei residui, l'utilizzo di modello matematici per processi e segnali, l'identificazione delle relazioni causa-effetto (inferenza) o algoritmi di classificazione, per individuare e classificare i guasti in componenti (sensori, attuatori o parti del processo). Tale supervisione consente sia il rilevamento tempestivo, sia la messa in atto di una risposta proattiva, riducendo i tempi di inattività e migliorando l'affidabilità complessiva del sistema.

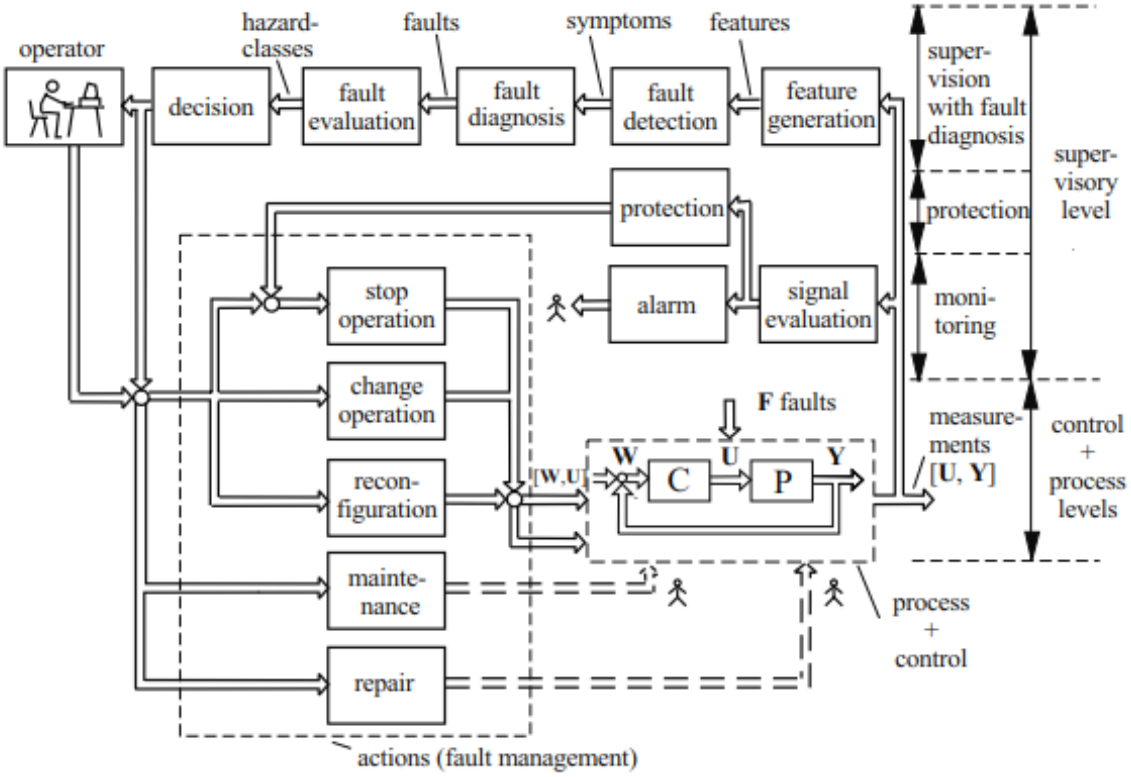


*Figura 2.2: Schema generale dei differenti metodi di supervisione con gestione guasti (supervisory loop) [1]*

La supervisione avanzata si compone delle seguenti fasi di monitoraggio e gestione, come rappresentato anche graficamente in *Fig. 1.2*

**Feature generation**: segnale caratteristico dalla cui analisi è possibile estrapolare l'informazione diagnostica (stato stimato, parametro identificato, etc.)

**Fault detection**: analisi dei segnali caratteristici per capire se il sistema è in stato di funzionamento corretto oppure è affetto da guasto.

**Fault diagnosis**: utilizzo di tecniche diagnostiche per determinare la tipologia, grandezza e posizione del guasto.

**Fault evaluation**: classificazione del guasto nelle diverse classi di rischio.

**Decision**: in base alla classe di rischio del fault classificato e del suo possibile grado di pericolo, è possibile scegliere la contromisura (manuale o automatica) da adottare.

Le *azioni di contromisura* che possono essere prese dal decisore tipicamente sono:

**Stop operation**: arresto in caso di un imminente pericolo per il processo o per l'ambiente.

**Change operation**: limitare l'espansione di un guasto tramite un cambiamento delle condizioni operative del sistema.

**Reconfiguration**: utilizzare altri sensori, attuatori o componenti ridondanti per mantenere il processo sotto controllo tramite una struttura "riconfigurata".

**Maintenance:** modificare i parametri di processo o sostituire le parti danneggiate istantaneamente o alla prima occasione disponibile.

**Repair**: rimuovere completamente la causa del guasto istantaneamente o alla prima occasione disponibile.

### 1.1.2 Terminologie della diagnosi

**Il guasto (fault)** può essere definito come qualsiasi deviazione non desiderata dallo stato nominale di funzionamento di un sistema o di un suo componente. Si tratta di uno stato in cui evolve il sistema che, sebbene non sempre immediatamente critico, rappresenta una condizione anormale che può evolvere nel tempo, compromettendo progressivamente le prestazioni del sistema.

Può essere di difficile individuazione se piccolo o nascosto. È possibile avere un guasto repentino (gradino) o una deriva (rampa).

Infine un guasto può portare a una rottura (*failure*) o malfunzionamento (*malfunction*).

**La rottura (failure)** è l'incapacità di un sistema di svolgere una funzione specifica richiesta sotto specifiche condizioni operative. È la conseguenza più critica di un guasto non gestito o ignorato. Mentre un guasto rappresenta un stato potenzialmente gestibile, la rottura segna il punto in cui la funzionalità del sistema è definitivamente compromessa. Questo concetto evidenzia l'importanza di rilevare e intervenire sui guasti prima che evolvano in rotture, dal momento che un'interruzione permanente delle funzioni di un sistema può avere conseguenze gravi.

I *failure* possono classificarsi:

**Per il numero:**

- Singoli;
- Multipli.

**Per la possibilità di essere predetti:**

- failure casuale;
- failure deterministico (predicibile sotto certe condizioni);
- failure sistematico (dipende da condizioni note).

**Il malfunzionamento (malfunction)** è un'irregolarità intermittente nel soddisfacimento di una funzione desiderata del sistema. Rappresenta un comportamento anomalo del sistema che, pur non traducendosi immediatamente in una rottura, segnala comunque un problema operativo. Spesso il malfunzionamento è la fase intermedia tra il guasto e la rottura e può essere caratterizzato da prestazioni degradate, errori sporadici o irregolarità nei dati di output.

Questo può verificarsi dopo l'avvio delle operazioni oppure per via di richieste eccessive sul processo.

Comprendere queste differenze semantiche e operative tra guasto, malfunzionamento e rottura è essenziale per implementare correttamente strategie di manutenzione.

### 1.1.3 Classificazione della diagnosi

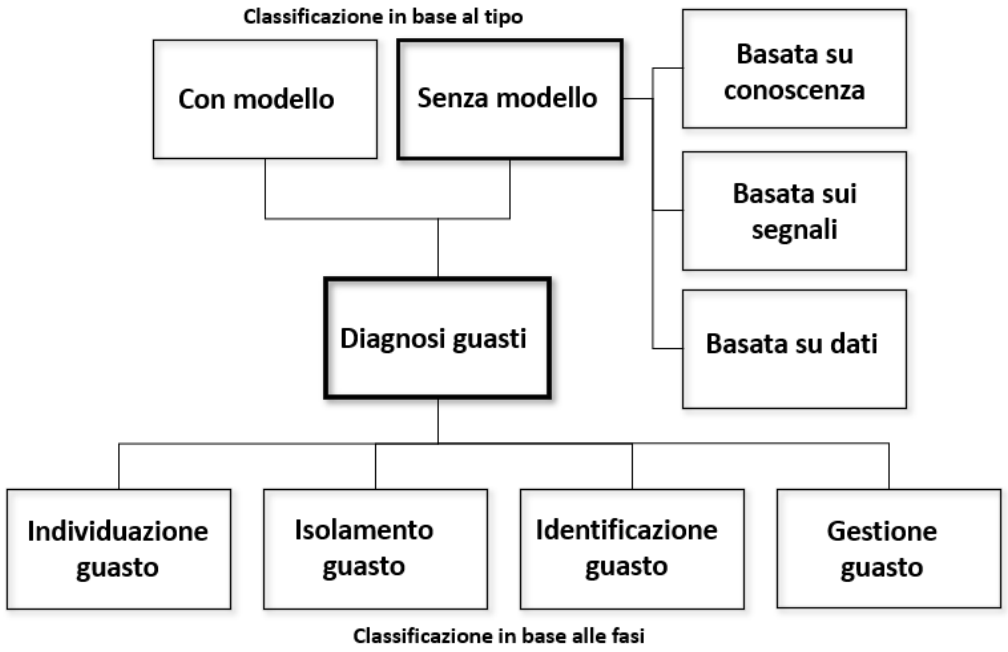


*Figura 3.3: Classificazione della diagnosi guasti*

La diagnosi guasti è comunemente classificata in base al dominio applicativo (*Fig. 1.3*) e si compone di 3 fasi:

- **Fault Detection** (FD – individuazione guasto): determinazione dei fault presenti nel sistema e dei momenti in cui essi sono individuati.
- **Fault Isolation** (FI – isolamento guasto: determinazione della tipologia, posizione e tempo di individuazione del fault; *segue la fault detection*.
- **Fault Identification** (FId – identificazione guasto): determinazione della dimensione e del comportamento tempo variante di un fault; *segue la fault isolation*.

Complessivamente la **Fault Diagnosis** comprende quindi la determinazione della tipologia, dimensione, posizione e tempo di individuazione del fault; *include la fault detection, isolation e identification* [1].

## 1.2 I droni e l'importanza della fault detection

I **droni** (noti anche come UAV - Unmanned Aerial Vehicles) sono dispositivi aerei senza pilota a bordo, progettati per operare autonomamente o essere controllati a distanza. Negli ultimi decenni hanno avuto un'evoluzione straordinaria trasformandosi, da strumenti di nicchia, a tecnologie essenziali, con un impatto significativo sia nel settore militare sia in quello civile. In origine sono stati concepiti per scopi militari, più recentemente in contesti civili e industriali, grazie alle innovazioni tecnologiche che ne hanno aumentato efficienza, versatilità e accessibilità.

Nel campo militare, i droni sono impiegati per missioni di sorveglianza e ricognizione in aree di conflitto, garantendo informazioni preziose senza mettere a rischio vite umane. Sono utilizzati per identificare minacce, monitorare il movimento delle truppe nemiche, e raccogliere dati strategici. Gli UAV possono svolgere anche attacchi mirati grazie all'integrazione di sistemi d'arma di precisione, riducendo la necessità di interventi di terra. Recentemente, il loro ruolo si è esteso anche al supporto logistico, con la capacità di trasportare rifornimenti in aree difficilmente accessibili, contribuendo alla resilienza operativa delle forze armate. L'efficienza e l'accuratezza dei droni nelle operazioni militari li rendono oggi un pilastro delle moderne strategie di difesa, alimentando però anche il dibattito su temi come l'etica dell'uso della forza a distanza e i rischi legati alla guerra asimmetrica [2].

In ambito civile, i droni hanno mostrato un potenziale ugualmente rivoluzionario. Impiegati nel monitoraggio ambientale, sono strumenti essenziali per la raccolta di dati su cambiamenti climatici, biodiversità e prevenzione di incendi. Questi offrono un'alternativa rapida ed economica ai metodi tradizionali, nella mappatura topografica, producendo modelli 3D dettagliati del territorio, sono utili in edilizia, urbanistica e gestione delle risorse naturali [2].

Nel settore dei trasporti, l'uso dei droni per le consegne è già in fase di sperimentazione in molti Paesi, potrebbero rappresentare un punto di svolta soprattutto in aree dove l'accesso ai servizi di trasporto è limitato. Aziende globali e start-up stanno investendo sugli stessi per sviluppare modelli di consegna rapidi, sostenibili e sicuri, mentre nel campo fotografico e video, i droni hanno rivoluzionato il modo di catturare immagini aeree, semplificando il lavoro per il cinema, il giornalismo e la promozione turistica [3].

Con l'evolversi della tecnologia, i droni sono stati introdotti pian piano in ambiti nuovi e disparati, come l'agricoltura di precisione, dove vengono impiegati per monitorare le coltivazioni, identificare infestazioni o stress idrico e per ottimizzare l'uso di risorse come l'acqua e i fertilizzanti. Per quanto riguarda la ricerca scientifica, i droni contribuiscono a progetti che vanno dallo studio delle migrazioni animali, alla mappatura

dei fondali marini. La loro versatilità li rende cruciali anche nelle operazioni di soccorso, soprattutto in caso di catastrofi naturali o zone difficilmente accessibili via terra, possono infatti effettuare ricognizioni rapide su vaste aree, identificare i superstiti e trasportare rifornimenti di emergenza.

Parallelamente a quanto affermato, insorgono problemi relativi alla sicurezza: la possibilità che i droni vengano usati per fini poco etici o terroristici, che non è da sottovalutare, di conseguenza c'è la necessità di sviluppare contromisure e regolamenti specifici per ovviare a questi problemi. Sul fronte della privacy, l'utilizzo di droni per la sorveglianza e raccolta di dati crea preoccupazioni legate alla riservatezza e alla gestione dei dati personali. Attuare una giusta regolamentazione diventa cruciale per garantire un uso etico degli UAV, cercando di porsi a metà tra innovazione e sicurezza per rispettare tutta la popolazione [4].

Considerato il largo impiego che i droni stanno avendo ultimamente, diventa indispensabile attuare strategie di *fault detection*, ovvero la determinazione dei *fault* presenti nel sistema e dei momenti in cui essi sono individuati. La *rilevazione* nel più breve tempo possibile è fondamentale, per garantire l'integrità e l'efficacia delle missioni, poiché un *malfunzionamento* imprevisto può compromettere l'intera operazione o farli precipitare.

Per lo studio contenuto in questo lavoro di tesi non si è utilizzato un drone, ma il dataset pubblico *UAV-FD*, creato per migliorare la *rilevazione di guasti* nei droni multirotore (esarotore nel caso in esame), in applicazioni dove la sicurezza è fondamentale, basti pensare al volo sopra aree affollate o voli per quanto riguarderà droni per passeggeri (quest'ultimo caso se si vuole dare uno sguardo al futuro). Il dataset contiene dati di volo reali, raccolti da un *esarotore* che ha subito danni a una delle eliche (danno artificiale). Questa situazione può verificarsi di solito a seguito di collisioni o atterraggi bruschi. Utilizzando un controller ArduPilot standard con firmware personalizzato, il sistema aumenta la frequenza di registrazione dei segnali, raccogliendo dati ad alta frequenza. In questo lavoro di tesi seguiranno dimostrazioni circa l'utilità del dataset, dove sono state applicate strategie di *fault detection* sfruttando il toolbox di MATLAB, Diagnostic Feature Designer, evidenziando come sia possibile individuare efficacemente i danni alle eliche [5].

## 1.3 Guasti sui droni

I guasti nei droni sono generalmente suddivisi in due categorie: *guasti agli attuatori* e *guasti ai sensori*. Sebbene esistano studi che trattano, ad esempio danni ai circuiti stampati, alla struttura del drone o alla superficie del veicolo volante stesso, questi non rappresentano in maniera usuale problemi di ricerca [6], in quanto sono problemi strutturali rilevabili a priori.

Nell'ambito del dataset che verrà analizzato in questo lavoro, si parlerà di guasti sugli attuatori.

In *Fig. 1.4* viene riportata un'immagine riepilogativa delle tipologie di fault presenti sui droni [6].

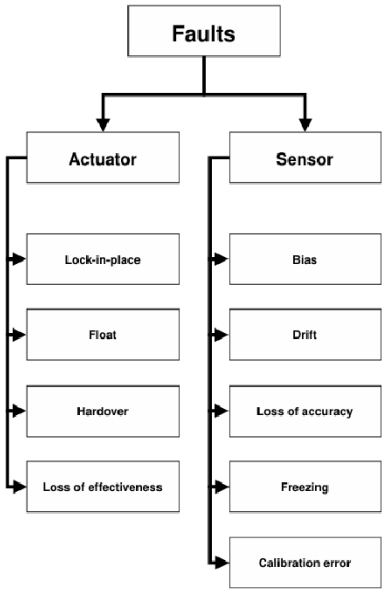


*Figura 4.4: Schematizzazione dei fault nei droni [6]*

Si passano ora in rassegna brevemente altre tipologie di guasto che possono essere presenti sui droni:

*Guasti alle eliche e ai motori*: Le eliche sono componenti soggetti a danneggiamenti causati da urti o stress durante il volo. Un'elica danneggiata può compromettere la stabilità del drone.

I motori possono surriscaldarsi o usurarsi nel tempo. Questi guasti possono derivare da carichi eccessivi o difetti di fabbricazione.

*Problemi al sistema di alimentazione*: Le batterie, tipicamente agli ioni di litio, sono sensibili a usura, temperature estreme e sovraccarichi. Una batteria esausta o danneggiata può ridurre significativamente l'autonomia del drone.

*Danni strutturali*: Condizioni atmosferiche avverse (vento forte, pioggia, basse temperature) possono indebolire il telaio o danneggiare i materiali del drone, compromettendone l'affidabilità e la resistenza.

*Errore software*: Benché rari, ci potrebbero essere errori nel software di controllo del volo o nella configurazione del firmware che possono causare malfunzionamenti durante il volo.

## 1.4 Rilevamento guasti

Nel campo del rilevamento dei guasti, le tecniche possono classificarsi principalmente in tre macrocategorie: *model-based*, *data-driven* e *tecniche ibride*.

Le tecniche *model-based* si basano sull'utilizzo di un modello matematico per andare a descrivere il comportamento fisico di un determinato sistema. I modelli sono solitamente ottenuti tramite leggi fisiche fondamentali (termodinamica, meccanica o elettronica) e rappresentano una descrizione teorica e ideale. La difficoltà principale risiede nella difficoltà a creare un modello preciso. Il modello risulta inoltre sensibile alle variazioni del sistema reale, sono presenti non linearità o i componenti stessi possono degradarsi.

Le tecniche *data-driven* analizzano I dati raccolti dalle letture dei sensori. Tra questi si trovano solitamente l'unità di misura inerziale (IMU), il GPS, o le misure della velocità (tubo di Pitot) [17]. Questi metodi sfruttano algoritmi di machine learning o tecniche di analisi di segnale, per estrarre le informazioni necessarie ai fini

della diagnosi. Il tutto può avvenire online od offline. Un esempio riportato nel [18] è l'analisi dei segnali acustici provenienti dai motori di un drone. Combinando i dati raccolti da microfoni ad algoritmi di classificazione (ad.es. *SVM*, o *Classification Tree*) è possibile monitorare le vibrazioni e i suoni emessi dai motori per rilevare guasti comuni, ovvero problemi nei cuscinetti o alle pale dei rotori. Questi metodi hanno raggiunto accuratezze superiori al 90%, quindi si sono dimostrati estremamente efficaci per identificare precocemente guasti nel sistema [18].

Le tecniche data-driven non richiedono una conoscenza del modello fisico del sistema, il che fa in modo che possano essere usate su droni differenti e in diverse condizioni operative.

La loro efficacia tuttavia dipende fortemente dalla qualità e dalla quantità di dati disponibili per l'addestramento di modelli e spesso, non si ha una disponibilità immediata di una mole di dati sufficiente.

Le tecniche ibride, d'altra parte, combinano elementi delle due categorie precedenti [6].

È necessario ora fare un'ultima distinzione nell'ambito del rilevamento guasti, nello specifico si può distinguere in rilevamento online e offline,

- **Rilevamento online**: utilizza sensori collegati alle macchine che monitorano continuamente i parametri operativi in tempo reale. Grazie all'integrazione con sistemi IoT e intelligenza artificiale, consente un'analisi automatizzata e tempestiva, riducendo i tempi di risposta ai guasti. È ideale per asset critici o difficilmente accessibili, garantendo una diagnosi più accurata e continuativa. Tuttavia c'è bisogno di unità di calcolo sufficientemente veloci, poiché, specie quando la mole di dati diventa alta, non è possibile perdere troppo tempo per l'elaborazione degli stessi.
- **Rilevamento offline:** prevede l'acquisizione manuale dei dati durante intervalli programmati, con log dove vengono raccolte misurazioni sul sistema. In questo caso, i dati raccolti vengono analizzati separatamente, spesso dopo la fine dell'acquisizione [1].

# 2. Descrizione del dataset e preprocessamento dati

*Figura 2.1: Esarotore UAV utilizzato per acquisire i dati*

Il dataset a cui si fa riferimento [16] è composto da 18 file *.mat*, acquisiti con il velivolo riportato in *Fig. 2.1*

Per realizzare questi voli è stata danneggiata artificialmente una pala del motore, prima di una porzione pari al 5% dell'elica, poi di una porzione pari al 10% dell'elica. La pala danneggiata, è stata spostata di volta in volta su uno dei sei motori diversi con l'avanzare delle prove. L'identificativo del nome del file difatti, ha all'interno anche l'informazione relativa a quale motore ha la pala danneggiata.


*Figura 2.2: Guasti sulla pala*

I dati grezzi raccolti sono distribuiti in 18 file *.mat*:

- Da 1) al 6) senza alcun fault, "*NO_FAULT#.mat*"
- Dal 7) al 12) è presente un fault del 5% (*Fig. 2.1*) su ognuno dei 6 motori, "*FAULT_M#_5.mat*"
- Dal 12) al 18) è presente un fault del 10% (*Fig. 2.1*) su ognuno dei 6 motori, "*FAULT_M#_10.mat*"

Da questi sono state estratte soltanto alcune variabili di interesse per il caso in esame, in particolare:

- 3x misure dall'accelerometro (misure dell'accelerazione lineare lungo gli assi X, Y e Z) – *acc_x, acc_y, acc_z*

- 3x misure dal giroscopio (misure della velocità angolare) – *gyr_x, gyr_y, gyr_z*

- 6x PWM – *pwm_1, pwm_2, pwm_3, pwm_4, pwm_5, pwm_6*

- 6x Velocità motori – *esc_1, esc_2, esc_3, esc_4, esc_5, esc_6*

- 6x Correnti – *cur_1, cur_2, cur_3, cur_4, cur_5, cur_6*

- 3x Roll, Pitch, Yaw stimati (posizioni angolari stimate) – *roll, pitch, yaw*

- 3x Roll,Pitch, Yaw desiderati (posizioni angolari desiderate) – *roll_des, pitch_des, yaw_des*

- 3x Velocità stimata – *vn, ve, vd*

- 3x Vibrazioni – *vibe_x, vibe_y, vibe_z*

Nel dataset sono presenti ulteriori variabili, come spiegato in [7], ma queste non verranno utilizzate perché non sono utili per il problema in esame.

A questo punto è stato possibile, come fatto in [5], pre-processare i dati andando a fare un resampling dei dati con un rate fisso impiegando una *zero order hold*, operando un ricampionamento a 350 Hz.

Alla fine del processo sono stati ottenuti dati in una singola dataTable, contenenti i 18 voli con le 36 variabili misurate. Oltre a queste variabili, sono state introdotte tre variabili aggiuntive*, roll_e*, *pitch_e* e *yaw_e*, ottenute per differenza tra le rispettive variabili stimate e misurate. Il motivo di questa scelta, è dovuto al fatto che si è supposto che queste misurazioni fossero influenzate fortemente dalla traiettoria del volo, quindi si è voluto valutare anche l'impatto dell'errore di queste misurazioni.

Oltre a questi passaggi, sono stati effettuati dei tagli sui voli: sono stati rimossi i primi 2 secondi di volo e gli ultimi 2 secondi, questo perché sono dati influenzati dall'alta velocità dei motori prima di raggiungere la quota desiderata, quindi poco adatti a rilevare guasti durante il normale funzionamento del drone.

Successivamente è stata aggiunta un'ulteriore colonna relativa al *faultCode*, dato categorico che può essere di 3 tipologie:

- 0 (no fault)
- 5 (fault al 5%)
- 10 (fault al 10%)

Questa sarà la colonna che andrà a categorizzare i dati presenti nella *dataTable* ai fini della classificazione.

# 3. Estrazione feature

## 3.1 Diagnostic feature designer

Il Diagnostic Feature Designer (DFD) di MATLAB è un'applicazione specifica per estrarre, analizzare e valutare le caratteristiche diagnostiche (feature) utilizzabili nella manutenzione predittiva e nella diagnosi dei guasti. Questa app permette di fare varie operazioni sui dataset o segnali importati al suo interno, Attraverso questo strumento si possono anche importare e suddividere dati, reali o simulati, in insiemi di addestramento e test per un eventuale algoritmo di machine learning.

L'applicazione offre una serie di strumenti per le seguenti fasi:

- Estrazione di feature in diversi domini: è possibile generare feature nei domini del tempo o della frequenza. Ad esempio si possono generare feature per macchine rotanti, per serie temporali, feature non lineari etc.
    - Feature temporali: parametri statistici come media, RMS, varianza, kurtosis e skewness, utili per rilevare variazioni nel segnale che possono indicare guasti.
    - Feature spettrali: come bande di frequenza specifiche e picchi spettrali, spesso usati per identificare fenomeni di degrado o di usura.
- Selezione e ranking delle feature: una volta estratte, le features possono essere valutate e ordinate in base alla loro capacità di distinguere efficacemente tra diverse classi in un sistema. L'applicazione offre strumenti come l'indice di Fisher, la mutual information o il test di Anova per calcolare l'importanza delle feature, permettendo di identificare quelle che meglio separano i dati tra classe normale e classi di guasto.
- Esportazione per classificazione: una volta estratte, le feature più efficaci possono essere trasferite direttamente nel Classification Learner, uno strumento che permette di addestrare modelli di classificazione, dove vengono utilizzate per addestrare modelli di machine learning, come alberi decisionali, SVM o reti neurali. Questo consente di sviluppare modelli di diagnosi accurati e automatizzati, sfruttando le feature estratte per ottimizzare l'identificazione dei guasti.
- Automazione e generazione di codice: una delle caratteristiche più importanti del Diagnostic Feature Designer è la possibilità di generare codice MATLAB per automatizzare l'intero processo di estrazione e selezione delle feature. Il codice generato può essere riutilizzato su nuovi set di dati e rendere il sistema adattabile e scalabile per ambienti produttivi che richiedono monitoraggio continuo. È possibile generare codice anche per integrazioni in Simulink, supportando flussi di dati in streaming per l'analisi in tempo reale [8].

## 3.2 Importazione dataTable e generazione feature

Una volta che il dataset è pronto, è stato possibile importarlo nel DFD tramite interfaccia grafica cliccando su "New Session" e selezionando la *dataTable* contenente tutti i segnali menzionati precedentemente. Ci si accerta che la colonna relativa a *faultCode* sia settata su *categorical* e si procede.

Si è impostata una finestra di 1 sec per il calcolo delle feature (Frame Size, *FS*), senza overlapping tra le finestre (Frame Rate, *FR*, pari al *FS*); il dettaglio grafico è riportato in *Fig. 3.1*

Questo ha una duplice utilità: si vogliono avere frame da 1 sec per poterli usare in seguito come elementi per il train o il test di un classificatore, e valutare se è possibile utilizzare questo approccio di FD online [5].

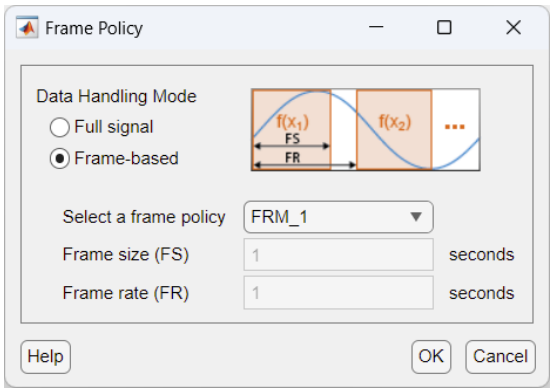


*Figura 3.1: Frame policy scelta*

Si è proceduto poi a generare le feature temporali per i primi due segnali (*acc_x, acc_y*), per analizzare il codice esportato dal DFD e cercare di replicarlo per gli altri segnali. Le feature temporali che verranno generate sono 13, nello specifico:

*Mean, RMS, Standard Deviation, Shape Factor, Kurtosis, Skewness, Crest Factor, Impulse Factor, Clearance Factor, Peak Value.*

| | |
|---|---|
| **Mean (Media)** | $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ |
| **Root Mean Square** | $\mathrm{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$ |
| **Standard Deviation** | $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (x_i - \bar{x})^2}$ |
| **Shape Factor (Fattore di Forma)** | $\text{Shape Factor} = \frac{\mathrm{RMS}}{\frac{1}{N}\sum_{i=1}^{N} \lvert x_i \rvert}$ |
| **Kurtosis (Curtosi)** | $\text{Kurtosis} = \frac{\frac{1}{N}\sum_{i=1}^{N} (x_i - \bar{x})^4}{\left(\frac{1}{N}\sum_{i=1}^{N} (x_i - \bar{x})^2\right)^2}$ |
| **Skewness (Asimmetria)** | $\text{Skewness} = \frac{\frac{1}{N}\sum_{i=1}^{N} (x_i - \bar{x})^3}{\left(\sqrt{\frac{1}{N}\sum_{i=1}^{N} (x_i - \bar{x})^2}\right)^3}$ |
| **Crest Factor (Fattore di Cresta)** | $\text{Crest Factor} = \frac{\max(\lvert x_i \rvert)}{\mathrm{RMS}}$ |
| **Impulse Factor (Fattore di Impulso)** | $\text{Impulse Factor} = \frac{\max(\lvert x_i \rvert)}{\frac{1}{N}\sum_{i=1}^{N} \lvert x_i \rvert}$ |

| | |
|---|---|
| **Clearance Factor (Fattore di Chiarezza)** | $\text{Clearance Factor} = \frac{\max(\lvert x_i \rvert)}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}}$ |
| **Peak Value (Valore di Picco)** | $\text{Peak Value} = \max(\lvert x_i \rvert)$ |
| **SINAD (Signal-to-Noise and Distortion Ratio)** | $\text{SINAD} = 10\log_{10}\left(\frac{P_{signal}}{P_{noise} + P_{distortion}}\right) \text{dB}$ |
| **SNR (Signal-to-Noise Ratio)** | $\text{SNR} = 10\log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) \text{dB}$ |
| **THD (Total Harmonic Distortion)** | $\text{THD} = 100 \cdot \frac{\sqrt{\sum_{n=2}^{N} P_{harmonic,n}}}{P_{fundamental}} \backslash\%$ |

A questo punto si è generato lo spettro dei due segnali in analisi, utilizzando un modello *AR* di ordine *10*, con un approccio Forward – Backward e senza Windowing. Per estrarre le features conviene considerare solamente l'intervallo di frequenze che comprende i picchi di potenza, quindi si è scelto di usare la banda di frequenza tra 10 e 150 Hz (Figura 3.2).

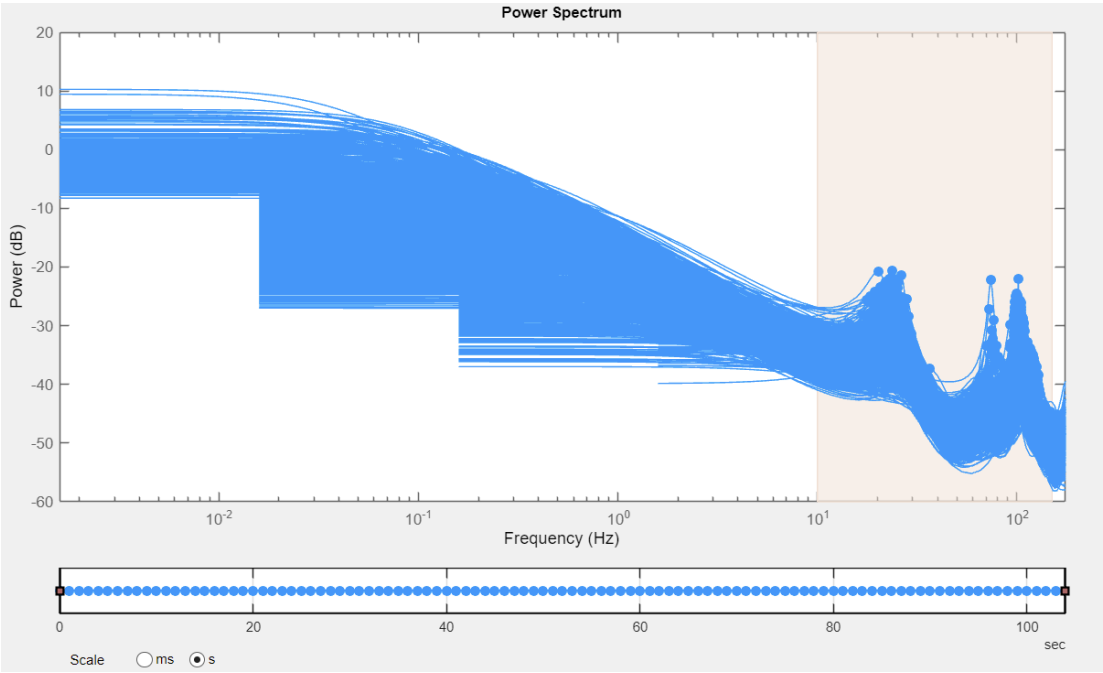


*Figura 3.2: Spettro di acc_y (lo spettro di acc_x è analogo)*

Ora è possibile generare le feature spettrali (5), in particolare: *First Peak Amplitude, First Peak Frequency, Second Peak Amplitude, Second Peak Frequency, Band Power.*

| | |
|---|---|
| First Peak Amplitude | $A_1 = \max_{f \in F_{range}} \lvert X(f) \rvert$ |
| First Peak Frequency | $f_1 = \arg \max_{f \in F_{range}} \lvert X(f) \rvert$ |
| Second Peak Amplitude | $A_2 = \max_{f \in F_{range}, f \neq f_1} \lvert X(f) \rvert$ |
| Second Peak Frequency | $f_2 = \arg \max_{f \in F_{range}, f \neq f_1} \lvert X(f) \rvert$ |
| Band Power (Potenza di Banda) | $P_{band} = \int_{f_{min}}^{f_{max}} \lvert X(f) \rvert^2 \, df$ |

A questo punto è stato possibile esportare il codice. Avendo notato che il DFD genera un nuovo blocco di codice per ogni segnale analizzato, è stato riformulato il codice, inserendo un ciclo *parfor* che iterasse attraverso tutti i segnali, che sono stati racchiusi in un vettore così fatto:

```
signals =

["acc_x","acc_y","acc_z","gyr_x","gyr_y","gyr_z","pwm_1","pwm_2","pwm_3","pwm_4",…
"pwm_5","pwm_6","esc_1","esc_2","esc_3","esc_4","esc_5","esc_6","cur_1","cur_2",…
"cur_3","cur_4","cur_5","cur_6","roll","pitch","yaw","roll_des","pitch_des","yaw_des",…
"roll_e", "pitch_e", "yaw_e", "vn","ve","vd","vibe_x","vibe_y","vibe_z"];
```

Lo scopo di questa scelta è duplice, si vuole rendere il codice più snello e più efficace con l'esecuzione in parallelo. Successivamente sono stati resi input della funzione anche i valori *FR* e *FS*. L'obiettivo è rendere il codice adatto ad effettuare test successivi, dove saranno fatte iterazioni attorno a questi due valori, per generare un numero maggiore di Feature Table, più velocemente.

## 3.3 Preparazione dati

Prima di procedere con la fase successiva, cioè l'importazione delle feature nel Classification Learner, è necessario effettuare una pulizia del dataset, quindi una volta esportata la tabella sono stati ripuliti i dati, rimuovendo valori NaN e Inf. In particolare ciò che è stato fatto è:

- eliminazione colonne con valori *NaN* superiori al 25% nella colonna stessa;

- sostituzione *NaN* col valore precedente nelle colonne dove questi siano minori del 25%;

- sostituzione *NaN* presenti come primo valore della colonna con *0.*

Così facendo si è passati da 706 colonne a 626, come riportate in *Fig. 3.3*


*Figura 3.3: Pulizia feature_Table*

Adesso è possibile dividere la *feature_Table*, omogeneamente rispetto al campo *EnsembleID_*, che rappresenta il tipo di volo (*nofault, fault 5%, fault 10%*) in una *feature_Table_Train* per l'addestramento e una *feature_Table_Test*, per testare successivamente i classificatori.

# 4 Diagnosi basata su classificazione

In questa sezione si farà uso di una seconda applicazione di Matlab, il Classification Learner (CL), adibito all'addestramento e al test di classificatori.

Il Classification Learner di MATLAB è un'applicazione sviluppata per semplificare l'addestramento e l'analisi di modelli di classificazione in contesti di machine learning supervisionato. È integrata nel toolbox "Statistics and Machine Learning" e si propone come uno strumento interattivo per affrontare problemi complessi, permettendo di esplorare i dati, selezionare le variabili più significative e ottimizzare i modelli.

Tipicamente viene importato un dataset, per poi addestrare diversi algoritmi di classificazione e valutare le performance dei modelli generati. L'app consente di selezionare manualmente i classificatori (Decision Tree, SVM, ensemble etc.) oppure di utilizzare la funzione *Quick-To-Train*, che permette di testare rapidamente più opzioni. È possibile analizzare i risultati mediante strumenti grafici come matrici di confusione o curve ROC, per fornire un'indicazione visiva della qualità delle predizioni del modello.

È possibile inoltre personalizzare il processo di addestramento. Questo include l'ottimizzazione degli iperparametri e l'uso di tecniche di validazione come la *cross-validation* o *l'holdout*, per garantire che il modello sia generalizzabile su dati nuovi. Inoltre, il software integra strumenti per l'interpretabilità, come le analisi di dipendenza parziale e i valori SHAP, utili per comprendere il contributo delle variabili indipendenti alle predizioni.

Dal punto di vista operativo, l'applicazione offre due vantaggi principali: la generazione automatica di codice MATLAB che riproduce l'intero workflow, facilitando la ripetizione e la modifica del processo, e la possibilità di esportare i modelli per l'uso in ambienti come Simulink o MATLAB Production Server. Questo rende il Classification Learner adatto sia per applicazioni accademiche che industriali, dove è necessario integrare modelli predittivi in pipeline più ampie [9].

## 4.1 Estrazione codici dal CL

Come specificato sopra, uno dei punti di forza del CL è l'esportazione del modello addestrato per permettere di fare test su altri dati, e valutarne le prestazioni per dati di test diversi.

Si è quindi esportato il codice per quanto riguarda i seguenti modelli:

- (Ensemble) Subspace discriminant
- (Ensemble) Boosted trees
- SVM (quadratic)
- SVM (cubic)
- SVM (linear)

Si riporta una breve panoramica generale per ogni tipologia di classificatore [9].

- *Ensemble Subspace Discriminant*: Utilizza l'ensemble learning, addestrando classificatori discriminanti basati sul metodo "*subspace*". Ogni classificatore viene addestrato su un sottoinsieme casuale delle feature predittive. È efficace con dataset con numerose feature ed è tipicamente usato per ridurre il rischio di overfitting grazie all'uso di più sottoinsiemi di feature. Questo metodo combina i risultati dei diversi classificatori per migliorare la precisione complessiva.

- *Ensemble Boosted Trees*: Combina alberi decisionali addestrati sequenzialmente usando il metodo "*boosting*". Nel *boosting* ogni modello viene costruito in maniera sequenziale apprendendo dagli errori del modello precedente. Utilizza algoritmi come *AdaBoost*, che assegnano maggiore peso agli esempi classificati erroneamente per il successivo ciclo di apprendimento, migliorando progressivamente l'accuratezza. Funziona bene per dataset complessi, ma richiede più attenzione alla scelta dei parametri per evitare overfitting.
- *SVM Quadratic*: È un modello di Support Vector Machine (SVM) che utilizza un kernel polinomiale di secondo grado (quadratico) per trasformare i dati in uno spazio ad alta dimensionalità. Questo permette di separare dataset non linearmente separabili nei dati originali.
- *SVM Cubic*: È simile all'SVM quadratico, ma con un kernel polinomiale di terzo grado (cubico). È adatto per modelli con relazioni non lineari più complesse tra le feature, ma può richiedere più risorse computazionali.
- *SVM Linear*: È un SVM con un kernel lineare che cerca di trovare l'iperpiano ottimale per separare i dati in due classi. È ideale per dataset che sono linearmente separabili o quasi separabili. È meno flessibile rispetto ai kernel non lineari, ma computazionalmente più efficiente.

Si è proceduto a modificare i codici per poter ricevere in input tutte le tipologie di feature che verranno trattate successivamente, inserendo un ulteriore dato in input (*predictor_names*). Quando le funzioni dei seguenti modelli verranno richiamate, basterà passare alla funzione i nomi dei predittori presi di volta in volta dalle intestazioni della tabella (*featureTable.Properties.VariableNames*), escludendo le colonne "EnsembleID_" (le quali indicano quale volo stiamo trattando) e le due colonne relative a FRM1 che indica quale frame si sta trattando.

## 4.2 Addestramento all features

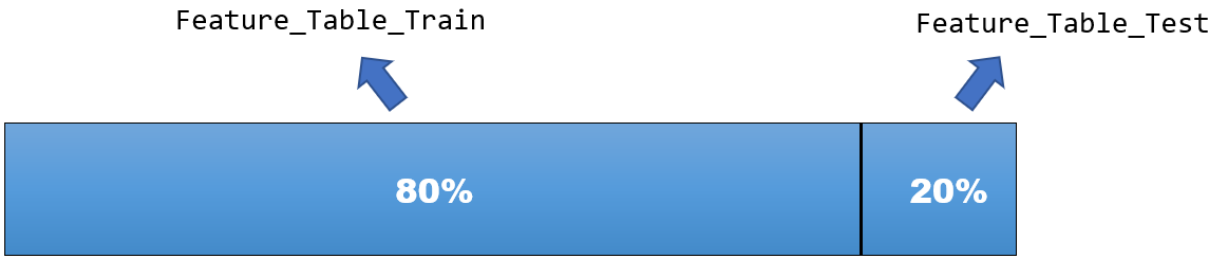


*Figura 4.1: Holdout*

In questa sezione sono riportati i primi test relativi all'addestramento dei classificatori. Si sono svolti dapprima prove relative a tutte le features *(Tab 4.1)*, con il relativo calcolo della matrice di confusione *(Fig. 4.2)*

| Classificatore | Accuracy (Test) |
|---|---|
| Ensemble (subspace discriminant) | **97.4%** |
| Ensemble (boosted trees) | 97.0% |
| SVM (quadratic) | 96.2% |
| SVM (cubic) | 95.1% |
| SVM (linear) | 94.0% |

*Tabella 4.1: Accuracy dei classificatori addestrati utilizzando tutta la tabella delle feature (622)*

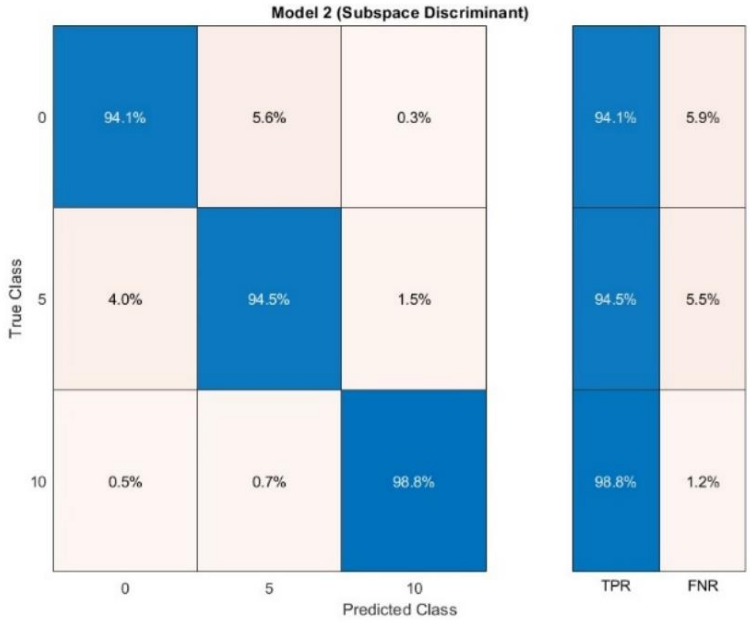


*Figura 4.2: Matrice di confusione di un classificatore **Subspace Discriminant** utilizzando tutta la tabella delle feature (622)*

Si può notare come il Subspace Discriminant sia stato quello che ha ottenuto un punteggio migliore utilizzando l'intero dataset.

## 4.3 Addestramento classificatori con top 14 feature

Si riportano di seguito i risultati per quanto riguarda l'addestramento utilizzando le migliori 14 features. La selezione è stata fatta con un test one-way ANOVA per realizzare un ranking delle feature, andando a selezionare successivamente le feature con un ANOVA score > 100.

Per quanto riguarda le feature *temporali*, verranno impiegate:

- Mean *(vibe_x, vibe_y, vibe_z, acc_x)*
- RMS *(vibe_x , vibe_y, vibe_z)*
- PeakValue *(vibe_x, vibe_y, vibe_z)*
- Std *(pwm_5)*

Invece le feature nel dominio della *frequenza*:

- BandPower *(vibe_y)*
- PeakFreq1 *(acc_y)*
- PeakAmp2 *(acc_z)*

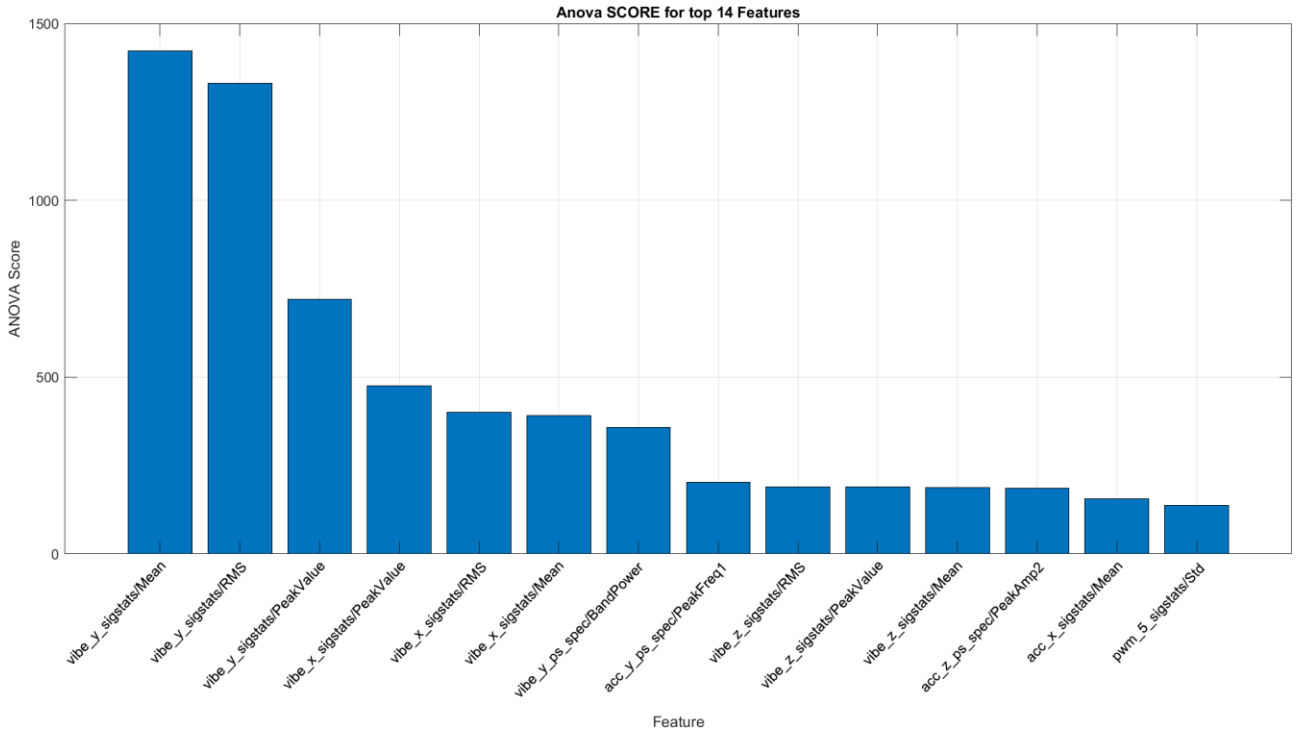


*Figura 4.3: Anova Test*

| Classificatore | Accuracy (Test) |
|---|---|
| Ensemble (subspace discriminant) | 90.9% |
| Ensemble (boosted trees) | 92.8% |
| SVM (quadratic) | **95.1%** |
| SVM (cubic) | 94.7% |
| SVM (linear) | 90.6% |

*Tabella 4.2: Accuracy dei classificatori addestrati utilizzando le migliori 14 feature (Anova)*

Si è poi proceduto a svolgere i medesimi test svolti per quanto riguarda tutta la tabella delle feature, addestrando gli stessi classificatori visti precedentemente:

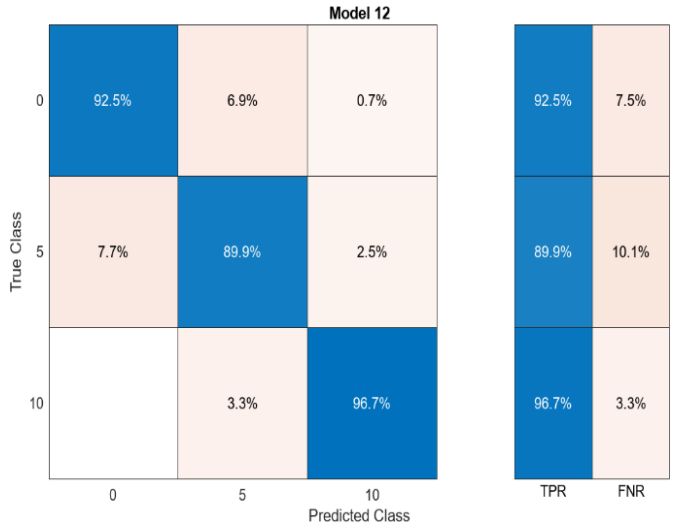


*Figura 4.3: Matrice di confusione di un classificatore **SVM (**Quadratic) utilizzando le migliori 14 feature (Anova)*

In questo caso si evince che i risultati migliori sono stati ottenuti da un classificatore SVM (quadratic)

## 4.4 Test con features vibrazioni

Durante I test eseguiti si è notato si è notato che nell'ANOVA test eseguito sulla tabella delle feature, la maggior parte delle feature sono relative ai segnali *vibe* lungo le tre direzioni, un segnale aggregato [5], di conseguenza si è voluto testare e valutare cosa accade utilizzando solo feature calcolate dai segnali di *vibe*.

È stato realizzato uno script Matlab che estrae dalla tabella soltanto le feature, nel dominio del tempo e della frequenza, relative a tali segnali. Successivamente sono stati addestrati e testati i medesimi classificatori della sezione precedente (*Tab 4.3*):

| Classificatore (only vibe features) | Accuracy (Test) |
|---|---|
| Ensemble (subspace discriminant) | 89.1% |
| Ensemble (boosted trees) | **92.8%** |
| SVM (quadratic) | 91.3% |
| SVM (cubic) | **92.8%** |
| SVM (linear) | 89.8% |

*Tabella 4.3: Accuracy dei classificatori addestrati utilizzando*

In questo caso i risultati migliori sono stati ottenuti con il Boosted Trees e l'SVM (cubic).

## 4.5 Holdout randomici

Al fine di valutare la sensibilità dei classificatori ai set di train e test, sono stati eseguiti dei test randomici. Nello specifico, si è scelto il calssificatore *Subspace Discriminant*, andando poi ad iterare 10 volte il processo di addestramento e test del classificatore, calcolando poi la media e la deviazione standard.

I risultati sono riportati in *Tab 4.4 e 4.5*

| **Validation Accuracy - subspace discriminant** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| It.1 | It.2 | It.3 | It.4 | It.5 | It.6 | It.7 | It.8 | It.9 | It.10 |
| 0.9557 | 0.9492 | 0.9652 | 0.9567 | 0.9605 | 0.9586 | 0.9614 | 0.9680 | 0.9642 | 0.9614 |
| **Mean** | **0.9601** | | | | **Std** | **0.0054** | | | |

*Tabella 4.4: Validation Accuracy Subspace Discriminant*

| **Test Accuracy - subspace discriminant** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **It.1** | **It.2** | **It.3** | **It.4** | **It.5** | **It.6** | **It.7** | **It.8** | **It.9** | **It.10** |
| **0.9849** | 0.9774 | 0.9736 | 0.9660 | **0.9849** | 0.9736 | 0.9623 | 0.9547 | 0.9623 | 0.9698 |
| **Mean** | **0.9709** | | | | **Std** | **0.0099** | | | |

*Tabella 4.5: Test Accuracy Subspace Discriminant*

## 4.6 Test variazione FR, FS

Successivamente si è voluto testare quanto i risultati venissero influenzati dalle variazioni di *Frame Rate*, *Frame Size* e di conseguenza l'overlap tra le finestre.

È stato realizzato quindi uno script Matlab, *overlaps.m* che itera due direzioni (*overlap_steps, FS_steps*) ottenendo una tabella *featureTableCell* contenente a sua volta 18 *FeatureTable*, e per ognuna di queste un'etichetta, che ne indica il caso che si sta considerando (*Fig. 4.6, Fig. 4.7*)

```
overlap_steps = 0:0.1:0.5; % Step per sovrapposizione (da 0% a 50% - step 10%)
  FS_steps = 0.512:0.256:1.024; % Frame Size (da 0.512 a 1.024 – step 0.256)
```

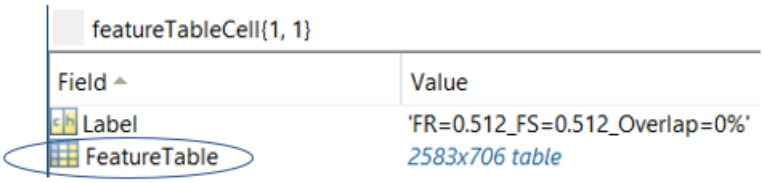


*Figura 4.6: Feature_Table FR=0.512, FS=0.512*

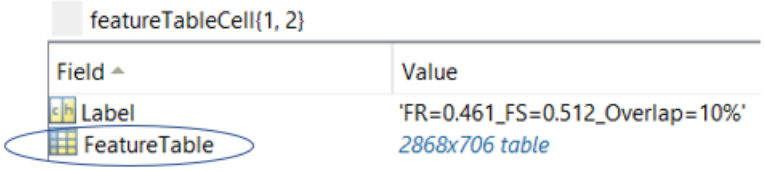


*Figura 4.7: Feature_Table FR=0.461, FS=0.512*

A questo punto è stato possibile addestrare 18 classificatori (è stato usato ancora il *Subspace Discriminant*), per vedere quanto la scelta del *Frame Rate* e del *Frame Size* influenzasse i risultati dell'accuracy in fase di validation o test.

I risultati sono riportati in *Tab 4.6 e 4.7*.

| Accuracy (validation) | Label | Accuracy (validation) | Label |
|---|---|---|---|
| 95.16% | 'FR=0.512_FS=0.512_Overlap=0%' | 96.88% | 'FR=0.358_FS=0.512_Overlap=30%' |
| 96.23% | 'FR=0.768_FS=0.768_Overlap=0%' | 97.61% | 'FR=0.538_FS=0.768_Overlap=30%' |
| 96.72% | 'FR=1.024_FS=1.024_Overlap=0%' | 97.77% | 'FR=0.717_FS=1.024_Overlap=30%' |
| 95.60% | 'FR=0.461_FS=0.512_Overlap=10%' | 97.18% | 'FR=0.307_FS=0.512_Overlap=40%' |
| 97.46% | 'FR=0.691_FS=0.768_Overlap=10%' | 97.82% | 'FR=0.461_FS=0.768_Overlap=40%' |
| 96.79% | 'FR=0.922_FS=1.024_Overlap=10%' | 97.91% | 'FR=0.614_FS=1.024_Overlap=40%' |
| 96.59% | 'FR=0.410_FS=0.512_Overlap=20%' | 97.04% | 'FR=0.256_FS=0.512_Overlap=50%' |
| 96.93% | 'FR=0.614_FS=0.768_Overlap=20%' | 97.75% | 'FR=0.384_FS=0.768_Overlap=50%' |
| 97.37% | 'FR=0.819_FS=1.024_Overlap=20%' | **98.21%** | 'FR=0.512_FS=1.024_Overlap=50%' |

*Tabella 4.6: Validation Accuracy test FR,FS*

| Accuracy (test) | Label | Accuracy (test) | Label |
|---|---|---|---|
| 96.51% | 'FR=0.512_FS=0.512_Overlap=0%' | 96.74% | 'FR=0.358_FS=0.512_Overlap=30%' |
| 97.09% | 'FR=0.768_FS=0.768_Overlap=0%' | **98.58%** | 'FR=0.538_FS=0.768_Overlap=30%' |
| 97.30% | 'FR=1.024_FS=1.024_Overlap=0%' | 97.84% | 'FR=0.717_FS=1.024_Overlap=30%' |
| 97.03% | 'FR=0.461_FS=0.512_Overlap=10%' | 96.74% | 'FR=0.307_FS=0.512_Overlap=40%' |
| **98.17%** | 'FR=0.691_FS=0.768_Overlap=10%' | 97.56% | 'FR=0.461_FS=0.768_Overlap=40%' |
| **98.26%** | 'FR=0.922_FS=1.024_Overlap=10%' | 96.75% | 'FR=0.614_FS=1.024_Overlap=40%' |
| 95.50% | 'FR=0.410_FS=0.512_Overlap=20%' | 97.48% | 'FR=0.256_FS=0.512_Overlap=50%' |
| 97.45% | 'FR=0.614_FS=0.768_Overlap=20%' | 97.82% | 'FR=0.384_FS=0.768_Overlap=50%' |
| **98.14%** | 'FR=0.819_FS=1.024_Overlap=20%' | **98.06%** | 'FR=0.512_FS=1.024_Overlap=50%' |

*Tabella 4.7: Test Accuracy test FR,FS*

Non sono stati rilevate particolari differenze tra gli addestramenti del *Subspace Discriminant* considerando i 18 casi in esame. In grassetto vengono riportati i casi in cui l'accuracy test è stata superiore al 98%.

Infine è stato eseguito un test simile al precedente, fissando il *Frame Rate* e ponendolo uguale al *Frame Size* di ogni finestra che verrà considerata per la generazione delle feature. Questa scelta comporta la creazione di un numero di *feature Table* diverso ogni volta, in base alla lunghezza del vettore *FS_steps*, una volta impostato lo step.

I valori minimi e massimi selezionati per questo test, sono gli stessi precedentemente usati, ovvero *0.512* e *1.024 sec*.

L'obiettivo è quello di andare a verificare se la scelta dello step influenza direttamente il risultato finale di Accuracy test. Per fare ciò è stato utilizzato l'indice di correlazione lineare tra un vettore contenente gli FS_steps e un vettore contenente i risultati di Accuracy Test.

Successivamente è stato graficato il risultato. In *Fig. 4.8* è mostrato uno schema che rappresenta il processo realizzato:

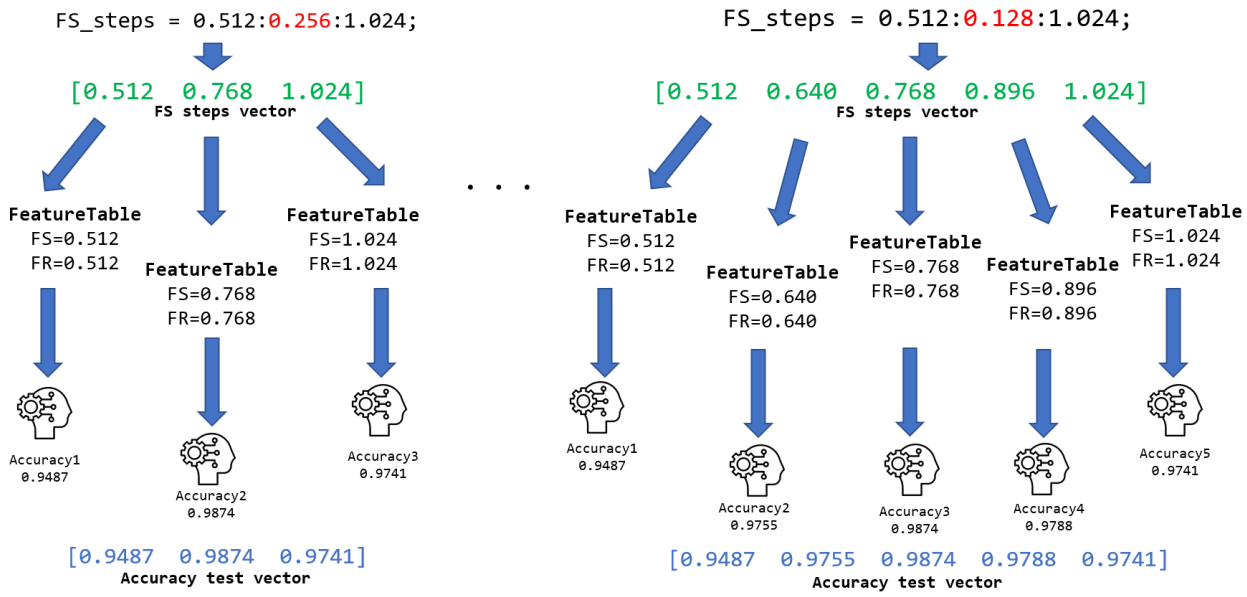


*Figura 4.8: Workflow test variazione FS*

Come si nota, di volta in volta, tra ogni iterazione e la successiva, la differenze è nel valore dello step che influenza il vettore *Fs_steps (in rosso),* questo fa sì che si generi un numero di *featureTable*, con un *Frame Size* e *Frame Rate* differenti per ognuna di esse.

Successivamente, viene diviso il dataset in Train e Test, (80% - 20%), e si addestra un classificatore Subspace Discriminant.

Una volta fatto ciò, si calcola l'indice di correlazione tra i due vettori, per poi graficare gli indici in relazione al numero di elementi presenti nel vettore di FS_steps per ogni ciclo.

I risultati sono riportati in *Fig. 4.9*.

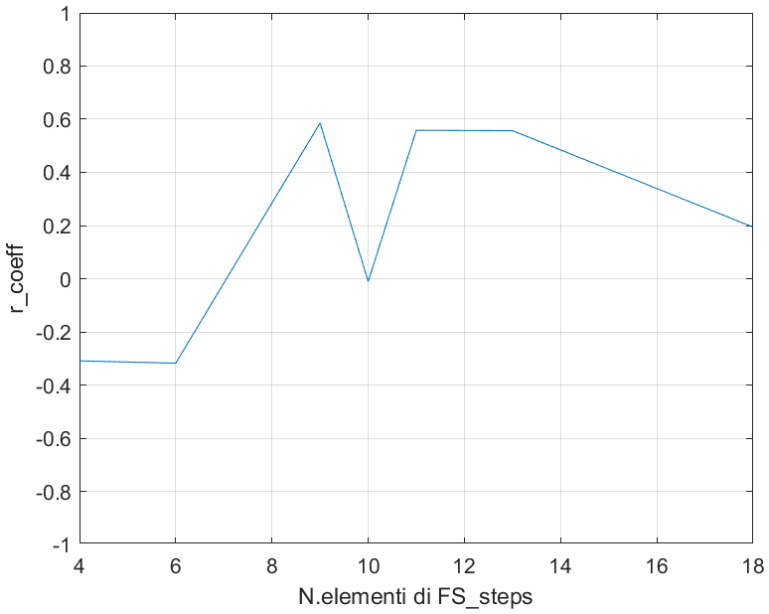


*Figura 4.9: Correlazione*

Osservando il grafico si evince che non c'è particolare correlazione tra le due variabili, questo si pensa potrebbe essere dovuto al fatto che non si è potuto considerare un valore massimo del *Frame size* superiore a *1.024,* infatti i dati di volo a disposizione sono tutti inferiori ai due minuti. Probabilmente settando un *FS* massimo superiore, i risultati sarebbero stati diversi.

# 5 Analisi in tempo-frequenza

## 5.1 Estrazione dati aggiuntivi

Al fine di proseguire questa analisi, è stato necessario estrarre dati aggiuntivi dai dati grezzi dei 18 voli a disposizione. Nello specifico sono stati utilizzati i dati della traiettoria del drone:

- Latitudine
- Longitudine
- Altezza
- RPM (dei 6 motori)

I primi due dati, sono registrati nei log di volo in gradi. L'altezza invece è direttamente estraibile dal dataset, in quanto proviene da una misura stimata dall'accelerometro e dal barometro presente a bordo. (misurata in cm) Le rotazioni dei sei motori invece sono espresse in *giri/min* (*RPM*, revolutions per minute). I primi 3 sono estraibili dalla variabile *GPS_0*, mentre le sei velocità dei motori sono presenti nelle variabili *ESC_0, ESC_1, ESC_2, ESC_3, ESC_4, ESC5* [7].

Sono di seguito riportate le formule di conversione geodetiche e i dati di volo.

La geodesia, è una disciplina relativa alle scienze della Terra che esplora la forma, le dimensioni e il campo gravitazionale del pianeta. Questa scienza nasce dalla necessità di misurare e rappresentare con precisione la superficie terrestre, con applicazioni che spaziano dalla cartografia alla navigazione satellitare, fino al monitoraggio delle deformazioni crostali per scopi scientifici e ingegneristici. La sua importanza è evidente nel mondo contemporaneo, dove sistemi di posizionamento globale come il GPS o il Galileo dipendono interamente da modelli geodetici accurati.

Il punto di partenza della geodesia è il riconoscere che la Terra non è una sfera perfetta, ma un ellissoide, schiacciato ai poli e rigonfio all'equatore. Questo modello ellissoidale è ulteriormente complicato dalla realtà fisica della superficie terrestre, che è irregolare a causa di montagne, valli e altre variazioni geografiche. Per rappresentare matematicamente la superficie, si utilizzano modelli geometrici standard, come l'ellissoide di riferimento (ad esempio il WGS84), e modelli fisici, come il geoide, che approssima il livello medio del mare.

Le coordinate geografiche sono il sistema di riferimento fondamentale per descrivere la posizione di un punto sulla superficie terrestre. Esse comprendono:

- Latitudine ($\phi$): è la distanza angolare di un punto rispetto al piano equatoriale, misurata in gradi, varia da 0° all'equatore a $\pm 90°$ ai poli.
- Longitudine ($\lambda$): è la distanza angolare rispetto al meridiano di Greenwich, considerato come origine, e varia da $0°$ a $\pm 180°$.
- Altitudine ($h$)**:** è l'altezza verticale di un punto rispetto a un riferimento, solitamente il livello del mare o il geoide.

Queste coordinate angolari possono essere convertite in distanze lineari (metri) attraverso formule che tengono conto della forma ellissoidale della Terra. Il calcolo preciso delle lunghezze di un grado di latitudine e di longitudine richiede di considerare il modello matematico dell'ellissoide e, in particolare, le variazioni locali del raggio terrestre.

Per calcolare la lunghezza di un grado di latitudine ($\Delta\phi$) e longitudine ($\Delta\lambda$) in metri, si parte dalla rappresentazione ellissoidale della Terra. Un ellissoide è definito dai suoi due semiassi principali: il **semiasse**

**maggiore** ($a$), pari a circa 6,378,137 metri, e il **semiasse minore** ($b$), pari a circa 6,356,752 metri (valori riferiti al WGS84). L'eccentricità dell'ellissoide (e) è data da:

$$e = \sqrt{1 - \frac{b^2}{a^2}}$$

Le formule per le distanze lineari di un grado di latitudine ($\Delta y$) e longitudine ($\Delta x$) sono riportate di seguito.

- Distanza per grado di latitudine:

$$\Delta y = \frac{\pi}{180} a(1 - e^2)\ (1 - e^2\ sin^2\phi)^{-3/2}$$

questa formula tiene conto della variazione del raggio di curvatura lungo il meridiano, che dipende dalla latitudine del punto (ϕ)

- Distanza per grado di longitudine

$$\Delta y = \frac{\pi}{180} a\ cos\phi\ (1 - e^2)\ (1 - e^2\ sin^2\phi)^{-3/2}$$

la lunghezza di un grado di longitudine è influenzata dalla convergenza dei meridiani verso i poli, il che comporta una diminuzione della distanza man mano che ci si sposta dall'equatore ai poli.

Se le distanze da approssimare sono relativamente brevi, quindi in un caso affine a quello in esame, è possibile considerare la terra localmente piatta; le formule si semplificano notevolmente ed è possibile basarsi su un raggio terrestre medio (R) di circa 6.371 km.

**Distanza per grado di latitudine**: La lunghezza di un grado di latitudine può essere considerata costante:

$$\Delta y = 111.32km\ =\ 111320m$$

**Distanza per grado di longitudine**: La lunghezza di un grado di longitudine varia con la latitudine secondo:

$$\Delta x = 111.32 \cdot cos\phi\ km\ =\ 111320\ cos\phi\ \cdot m$$

Questo approccio semplificato è particolarmente utile in applicazioni pratiche, come la navigazione terrestre o il calcolo di distanze approssimative su piccola scala, dove le differenze introdotte dall'approssimazione sono trascurabili [10].

Detto ciò, per quanto riguarda il volo di un drone su un territorio ristretto, queste formule semplificate sono ottime e verranno impiegate per convertire i dati estratti dai log di volo dell'esarotore.

Sono stati modificati i codici per l'estrazione dei dati dai 18 dati di volo grezzi, ripetendo le medesime operazioni di taglio e sincronizzazione fatte per i precedenti dati. Alla fine del processo, è stata ottenuta una nuova dataTable per questa sezione, contenente nove colonne in più, sei di queste relative ai dati di rotazione dei sei motori e tre di queste contenenti i dati relativi a latitudine, longitudine e altitudine.

I dati di latitudine e longitudine sono stati convertiti come indicato sopra. L'approccio che è stato adottato è stato quello di settare come latitudine e longitudine di riferimento, (*lat0* e *lon0*) le prime due di ogni log di volo, per poi calcolare la differenza rispetto a quel punto e convertirlo secondo le formule viste sopra, derivanti dall'approssimazione localmente piana della Terra. Un procedimento simile è stato fatto per l'altitudine, calcolando lo scostamento dal primo dato estratto da ogni log di volo, che fungerà da riferimento iniziale.

In *Fig. 5.1-5.3* sono riportati tre plot dei voli dei droni, uno per ogni classe di guasto. Il punto iniziale menzionato sopra (*lat0, lon0, z0*), è riportato in rosso nel grafico 3D.

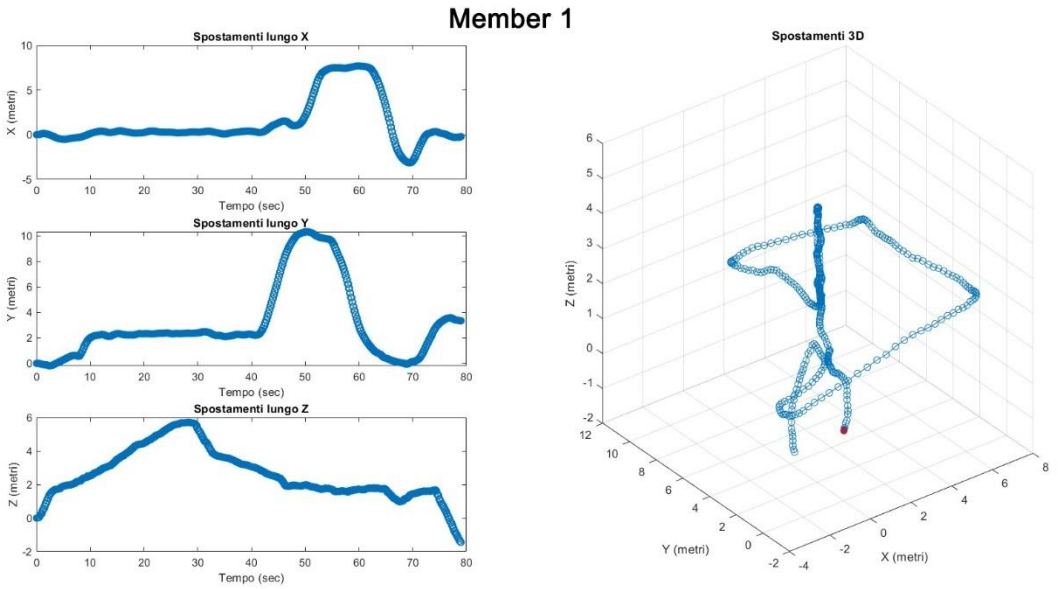


*Figura 5.1: Traiettoria Member 1*

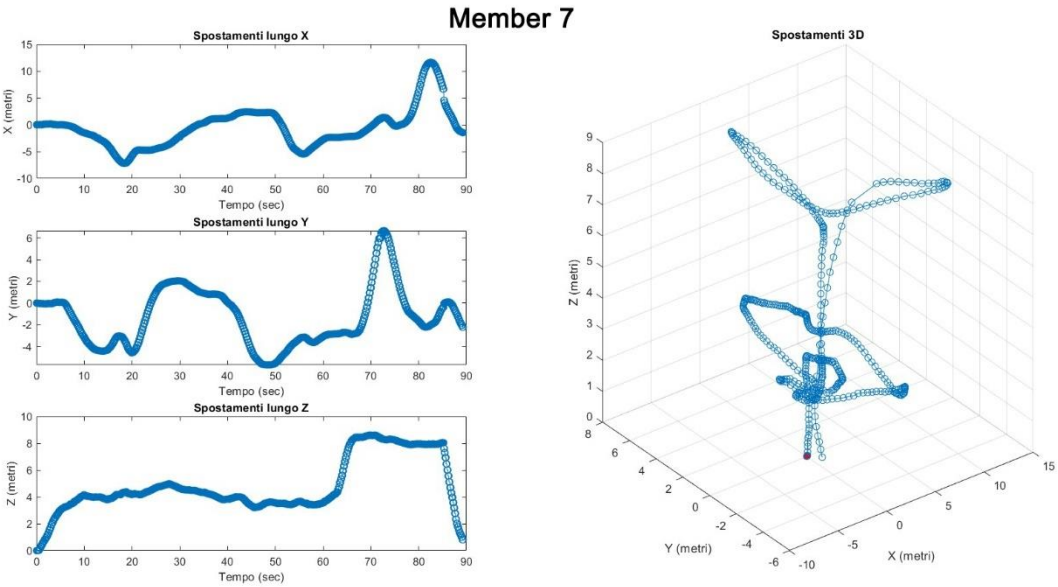


*Figura 5.2: Traiettoria Member 7*

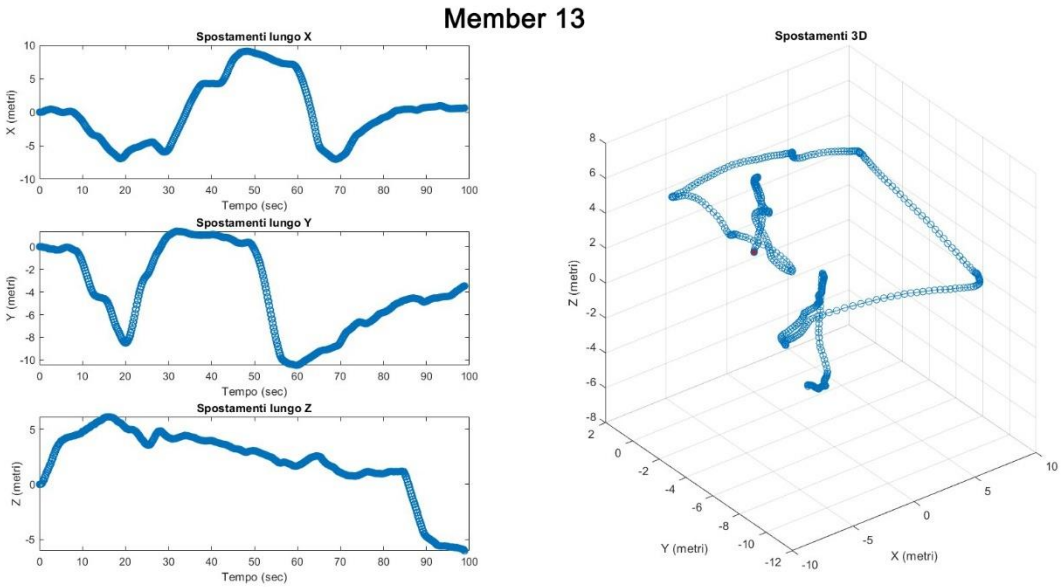


*Figura 5.3: Traiettoria Member 13*

## 5.2 Spettrogramma

Uno **spettrogramma** è una rappresentazione visiva del contenuto spettrale di un segnale in funzione del tempo. Mostra come l'energia di diverse frequenze evolve nel tempo, fornendo una descrizione combinata della distribuzione temporale e frequenziale del segnale.

Lo spettrogramma si basa sulla **Short-Time Fourier Transform (STFT)**, una versione "finestrata" della trasformata di Fourier. La STFT divide il segnale in segmenti sovrapposti (tramite una finestra temporale) e calcola la trasformata di Fourier per ciascun segmento. Questo approccio consente di ottenere informazioni locali sulle frequenze. Questa tecnica verrà approfondita meglio nel seguito. Lungo gli assi si trovano:

Ascisse (asse $x$): Tempo.

Ordinate (asse $y$): Frequenza. Mostra le componenti frequenziali del segnale, solitamente in Hertz ($Hz$).

Colore: Rappresenta l'intensità. La scala cromatica indica l'intensità o l'ampiezza (spesso espressa in termini di potenza o decibel) delle componenti frequenziali presenti a un dato istante [1].

Prima di procedere verrà fatta una breve digressione sulla Trasformata di Fourier fino a giungere alla Trasformata di Fourier Short, che è quella che viene utilizzata dalla funzione *spectrogram* di Matlab, utilizzata in questo lavoro di tesi.

### 5.2.1 Fourier Transform

La trasformata di Fourier, sviluppata dal matematico francese Jean-Baptiste Joseph Fourier nel XIX secolo, e ha rivoluzionato il modo in cui vengono interpretati e manipolati i dati nel dominio della frequenza. Questo strumento permette la decomposizione di un segnale nel dominio temporale nelle sue componenti sinusoidali. Questa operazione consente di analizzare le componenti frequenziali di segnali complessi, rendendola indispensabile in numerosi campi come l'elaborazione dei segnali, le comunicazioni, la fisica, l'ingegneria, l'elaborazione del suono e l'analisi delle immagini.

La trasformata continua di Fourier di una funzione f(t) è definita formalmente come:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t}\, dt$$

$F(\omega)$ rappresenta lo spettro del segnale $f(t)$ ed è individuato da modulo e fase. $\omega$ è la pulsazione angolare, $j$ indica l'unità immaginaria, mentre $e^{-j\omega t}$ rappresenta la componente esponenziale complessa, che permette la trasformazione nel dominio delle frequenze.

### 5.2.2 Discrete Time Fourier Transform (DTFT)

C'è da considerare che la maggior parte dei segnali che vengono trattati, in tutti i campi, sono segnali campionati, di conseguenza si introduce la trasformata di Fourier discreta (DFT): è una versione della trasformata di Fourier applicata a segnali campionati. Questa permette di calcolare lo spettro di frequenza di un numero finito di punti di dati. La DFT è una delle tecniche più utilizzate nell'elaborazione digitale dei segnali. L'equazione della DFT per una sequenza di $N$ campioni $x[n]$ è:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn}$$

dove $X[k]$ rappresenta il valore della trasformata al k-esimo componente di frequenza. I vantaggi principali della DFT includono la capacità di analizzare segnali discreti e l'efficacia computazionale quando il numero di campioni è gestibile.

### 5.2.3 Fast Fourier Transform (FFT)

La trasformata di Fourier discreta richiede, $2N$ moltiplicazioni e $2N - 1$ addizioni il che implica un'onerosità a livello computazionale che cresce esponenzialmente con la dimensione del dataset. Al fine di ovviare a questo problema è stata introdotta la trasformata di Fourier veloce.

La trasformata di Fourier veloce (FFT) è un algoritmo efficiente per calcolare la DFT e le sue varianti. La FFT riduce drasticamente la complessità computazionale della DFT, rendendo praticabile l'analisi di segnali con un gran numero di campioni. Un tipico algoritmo FFT riduce il numero di operazioni da $O(N^2)$ a $O(N\, logN)$. Questo rende la FFT ideale per applicazioni in tempo reale e per l'elaborazione di grandi dataset.

L'inconveniente di questo metodo è che richiede un numero limitato di campioni che deve corrispondere a una potenza di 2 $(2^N)$. Per ovviare a questo problema, solitamente, si adotta un troncamento del dataset o lo zero padding (riempimento del set di dati disponibile, con un numero di $0$ in modo da ottenere un numero di campioni complessivo che sia una potenza di $2$).

### 5.2.4 Short Time Fourier Transform (STFT)

La STFT è una tecnica per analizzare i segnali non stazionari (segnali il cui contenuto in frequenza varia nel tempo). Quando si ha a che fare con segnali che presentano queste caratteristiche, o quando un guasto introduce nel sistema una nuova frequenza che varia nel tempo si utilizza questa tecnica. Si tratta di una generalizzazione della trasformata di Fourier che consente di rappresentare un segnale nel dominio **tempo-frequenza**.

$$\mathrm{STFT}\{\mathrm{x(t)}\}(\tau,\omega) \equiv \mathrm{X}(\tau,\omega) = \int_{-\infty}^{\infty} \mathrm{x(t)w(t-\tau)e^{-j\omega t}}\ \mathrm{dt}$$

$x(t)$ è il segnale originale.

$w(\tau - t)$ è la funzione finestra centrata in $t$, che seleziona una porzione del segnale. Solitamente una finestra può essere rettangolare, di Hamming, di Hann, o Gaussiana.

La **lunghezza della finestra** influenza la risoluzione temporale. La scelta della durata della finestra temporale è la conseguenza di un compromesso tra risoluzione del segnale nel tempo e quella dello spettro del segnale.

Si può affermare che la scelta della finestra influisce su:

- *Risoluzione temporale:* Precisione nel determinare eventi nel tempo.
- *Risoluzione in frequenza*: Precisione nel distinguere componenti vicine in frequenza.

Una finestra più corta permette di catturare variazioni rapide nel tempo ma riduce la precisione nelle frequenze, mentre una finestra più lunga migliora la precisione nelle frequenze ma rischia di perdere dettagli temporali.

*Overlap:* le finestre possono sovrapporsi per migliorare la continuità tra i segmenti e ridurre le perdite d'informazione, tuttavia una sovrapposizione elevata aumenta il costo computazionale dell'operazione [1].

### 5.2.5 funzione Spectrogram e settaggio dei parametri

Si farà riferimento alla funzione *spectrogram* di Matlab [12], capace di calcolare la STFT di un segnale, settati alcuni parametri. In particolar modo questi risultano essere:

- *signal:* segnale in analisi;
- *windowSize:* dimensione della finestra;
- *overlap:* numero di campioni da sovrapporre;
- *nfft:* specifica il numero di punti per la Trasformata di Fourier, influenzando la risoluzione in frequenza.
- *Fs*: frequenza di campionamento, necessaria per rappresentare correttamente le frequenze del segnale.

Questi parametri sono settati per il caso in esame come:

```
windowSize =256;
overlap = windowSize / 2;
nfft = 256;
Fs=350.
```

## 5.3 Scalogramma

Uno scalogramma è una rappresentazione grafica che mostra l'evoluzione delle componenti di frequenza di un segnale al variare nel tempo. Questo grafico utilizza la trasformata wavelet diversamente da uno spettrogramma. Lungo gli assi si trovano:

Ascisse (asse $x$): Tempo.

Ordinate (asse $y$): rappresenta la scala (spesso legata inversamente alla frequenza: scale più grandi = frequenze più basse).

Colore: Rappresenta l'intensità Ogni punto dello scalogramma corrisponde alla "potenza" del segnale a un certo tempo e a una certa scala. Nella pratica, fornisce informazioni su quanto è forte il contributo del segnale a quella specifica frequenza/scala in quel momento.

Anche questa rappresentazione risulta utile per segnali non stazionari [14].

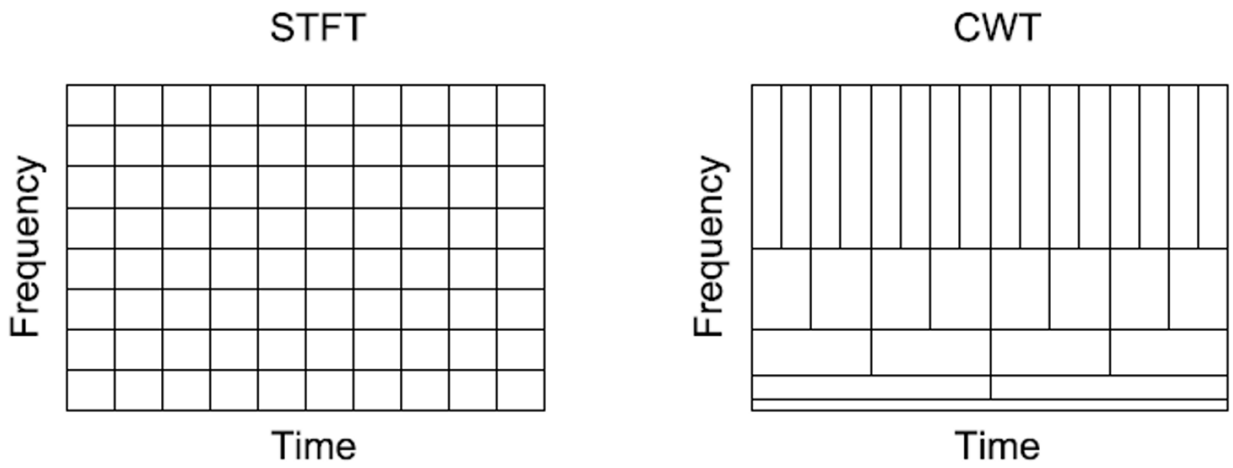


*Figura 5.4: Tiling del piano tempo-frequenza nella STFT (sinistra) e CWT (destra) [15]*

### 5.3.1 Continuous wavelet transform

La trasformata wavelet continua (CWT) è una tecnica analitica che si distingue per la capacità di rappresentare un segnale in una forma tempo-frequenza scalabile. Questo approccio si basa su funzioni matematiche chiamate *wavelet*, progettate per essere localizzate sia nel dominio temporale sia in quello delle frequenze. A differenza della trasformata di Fourier, che fornisce una rappresentazione globale del segnale, la CWT consente un'analisi multi-risoluzione, ideale per segnali non stazionari o che presentano variazioni nel tempo, come quelli osservati in applicazioni fisiche, biomediche o industriali.

La trasformata wavelet continua scompone un segnale utilizzando una funzione wavelet madre, scalata e traslata per esplorarne i dettagli su diversi intervalli di tempo e frequenza. Questa scalabilità è un vantaggio significativo, poiché permette di catturare sia caratteristiche globali che locali.

Le wavelet madre possono subire una dilatazione temporale di un fattore $a$, oppure possono essere traslate nel tempo di un fattore $\tau$:

$$\Psi^*(t, a, \tau) = \frac{1}{\sqrt{a}} \Psi\left(\frac{t-\tau}{a}\right)$$

Il fattore di scala, $1/\sqrt{a}$, è introdotto per ottenere una scalatura corretta della densità di potenza spettrale.

Se le frequenza media di una wavelet è pari a $\omega_0$, allora scalando la wavelet di una quantità pari a $\tau/a$, si ottiene una frequenza media scalata pari $\omega_0/a$ .

La trasformata wavelet continua è definita come:

$$CWT(a,\tau) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} y(t) \Psi \left( \frac{t-\tau}{a} \right) dt$$

Per esempio, dettagli in un segnale possono essere analizzati con funzioni wavelet più concentrate, mentre pattern globali possono essere esaminati con wavelet dilatate. La rappresentazione risultante è un piano tempo-frequenza che permette di osservare come l'energia del segnale è distribuita nei due domini [1].

Le funzioni wavelet comunemente usate includono le wavelet di *Morlet, Daubechies e Haar*, ognuna con caratteristiche specifiche: la wavelet di Morlet è utile per analisi dove il contenuto frequenziale oscillatorio è predominante, mentre le wavelet di Daubechies sono preferite per la loro ortogonalità e supporto compatto, che le rendono adatte a molte applicazioni ingegneristiche e scientifiche.

Restando nell'ambito del rilevamento guasti, la CWT si è affermata soprattutto nei sistemi meccanici e rotanti, come turbine, motori e pompe. Le vibrazioni prodotte da tali sistemi sono tipicamente non stazionarie e contengono informazioni cruciali per identificare problemi come usura o difetti nei cuscinetti. La capacità della CWT di fornire un'analisi tempo-frequenza dettagliata la rende ideale per rilevare segnali deboli o anomalie nascoste nel rumore di fondo.

Nel monitoraggio delle macchine rotanti, la CWT permette di individuare variazioni nelle componenti frequenziali delle vibrazioni, che spesso precedono un guasto meccanico. L'uso della CWT per estrarre i massimi locali dei coefficienti wavelet può rivelare cambiamenti nella dinamica del sistema che non sarebbero rilevabili con metodi tradizionali. Inoltre, tecniche avanzate come la wavelet synchrosqueezing migliorano la risoluzione tempo-frequenza, facilitando una diagnosi più precisa.

Un ulteriore vantaggio significativo dell'applicazione della CWT nel rilevamento dei guasti è la capacità di identificare il contenuto frequenziale specifico associato a determinati tipi di difetti. Ad esempio, guasti nei cuscinetti generano tipiche frequenze di risonanza che possono essere identificate nel dominio wavelet, permettendo interventi tempestivi prima che il problema comprometta l'intero sistema.

### 5.3.2 Funzione CWT e settaggio dei parametri

Si farà riferimento alla funzione *CWT* [13] di Matlab, che dato un segnale ne calcola la trasformata wavelet continua. Grazie a questo comando si è in grado di costruire lo scalogramma di un segnale e mostrare come le componenti in frequenza evolvono nel tempo.

Tra i parametri più significativi che possono essere settati, oltre al segnale, figurano la frequenza di campionamento dello stesso e il tipo di famiglia di funzioni wavelet che devono essere utilizzate. Nel caso in esame è stato omesso, quindi sono state utilizzate le *Morse* wavelet.

## 5.4 Segnali di interesse

Il segnale maggiormente impattante secondo l'Anova test, come si evince dalla *Fig. 4.3 (Anova test)* è il segnale *vibe_y,* tuttavia il segnale *vibe* è un segnale aggregato [5], quindi una volta calcolato lo spettrogramma dello stesso si è notato che risultava essere molto rumoroso.

Fatta questa considerazione il segnale che è stato selezionato per l'analisi è *acc_y*, che mostrava zone nitide a maggiore densità nello spettrogramma. Sono stati fatti anche test con *acc_x* e *acc_z*, ma nessuno dei due restituiva informazioni adeguate nel grafico tempo-frequenza, quindi non verranno riportati.

Nelle *Fig. 5.5-5.7* vengono riportati tre spettrogrammi (*acc_y*), uno per ogni classe di guasto:

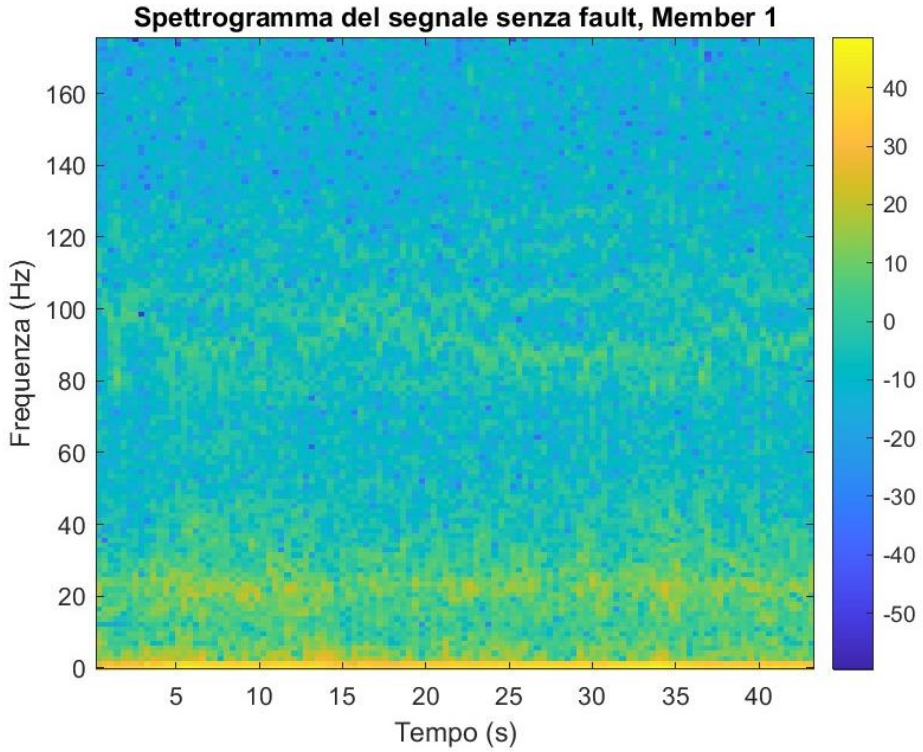


*Figura 5.5: Spettrogramma Member 1*

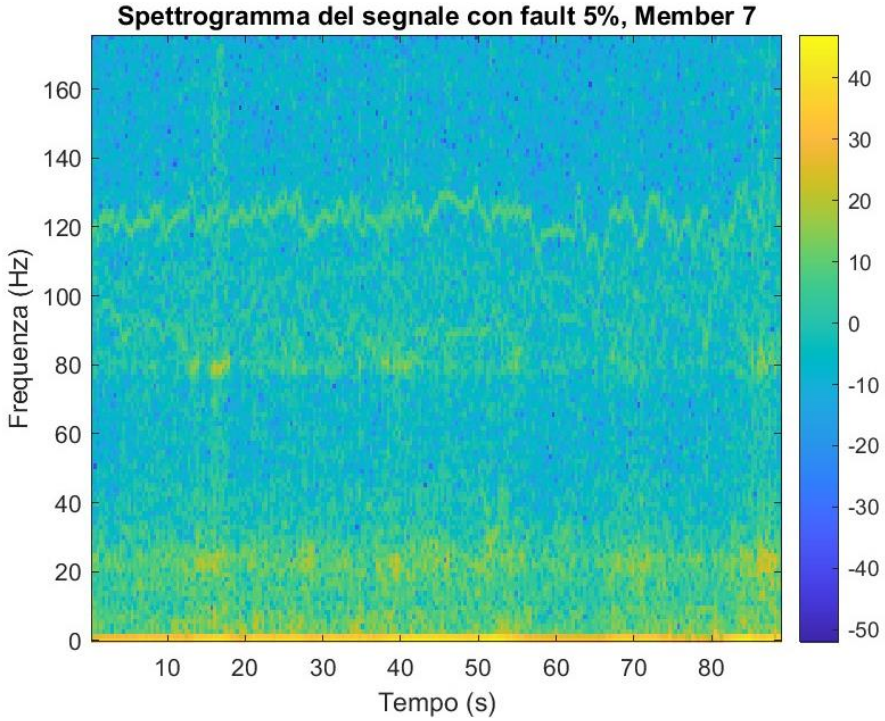


*Figura 5.6: Spettrogramma Member 7*

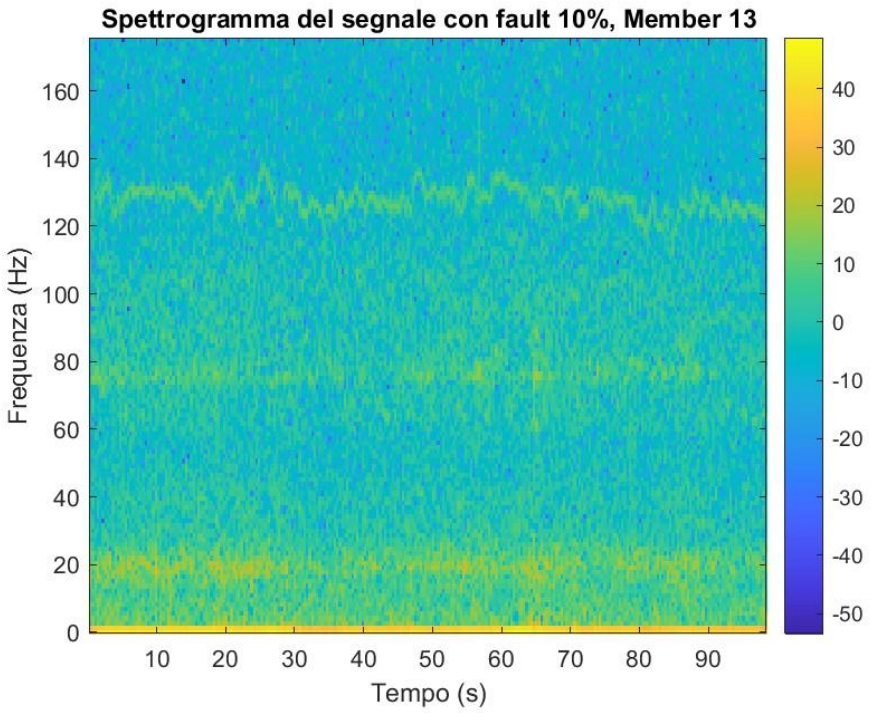


*Figura 5.7: Spettrogramma Member 13*

Da una prima analisi visiva degli spettrogrammi, si nota che le frequenze interessate, tralasciando una componente rumorosa a 20Hz presente in tutti i voli analizzati, oscillano tra i 70 e i 140 Hz. Questo rende molto difficoltoso il processo di rilevamento guasti analizzando la sola immagine: con un dataset composto da 18 voli, si possono generare 18 immagini; pur supponendo di suddividere le immagini in frame più piccoli (ad esempio di 10 secondi), ed etichettando gli stessi, non sarebbe possibile fare classificazione adeguatamente.

In *Appendice* vengono riportati gli spettrogrammi di tutti i voli.

## 5.5 Confronto Spectrogram – CWT con accelerazioni e traiettorie

In questa sezione sono messi a confronto gli spettrogrammi ricavati nella sezione precedente, con gli spostamenti lineari lungo le tre direzioni, le accelerazioni lineari lungo le tre direzioni, e le sei velocità di rotazione dei motori espresse in Hz. Lo stesso verrà poi fatto con gli scalogrammi.

Si vuole analizzare la dipendenza del contenuto in frequenza mostrato nei due grafici, dalla traiettoria che il pilota ha effettuato, dalle accelerazioni, alle velocità dei motori. Sono riportati (*Fig. 5.8 – 5.10*) confronti relativi agli stessi tre voli riportati nella sezione precedente (*Member 1, Member 7, Member 13*). Per consultare i confronti di tutti i 18 voli si rimanda all'*Appendice*.

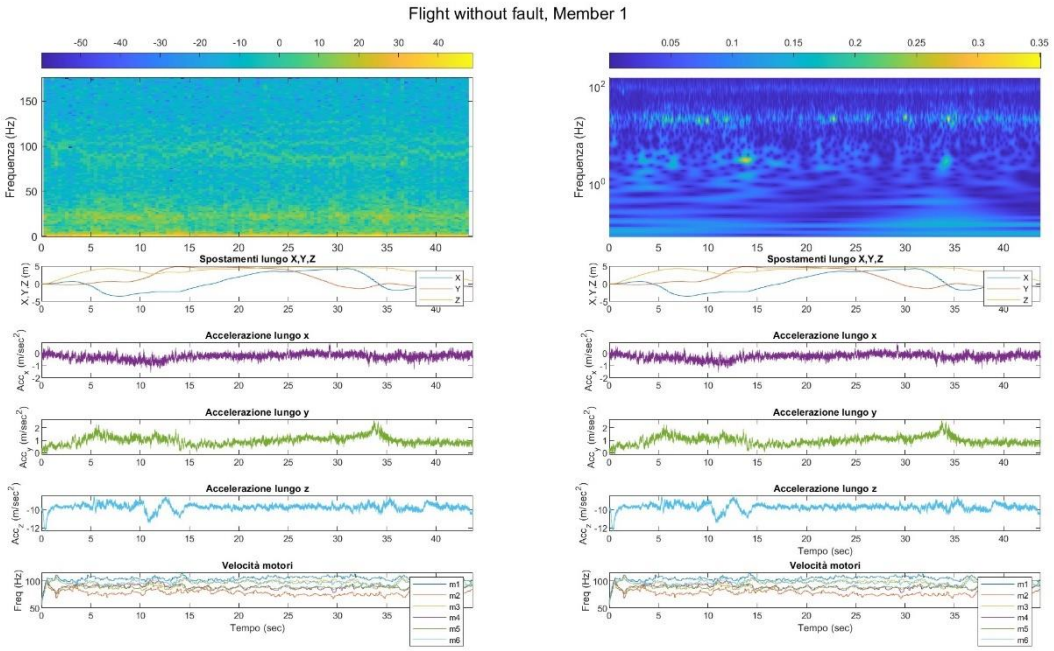


*Figura 5.8: Confronto Member 1*

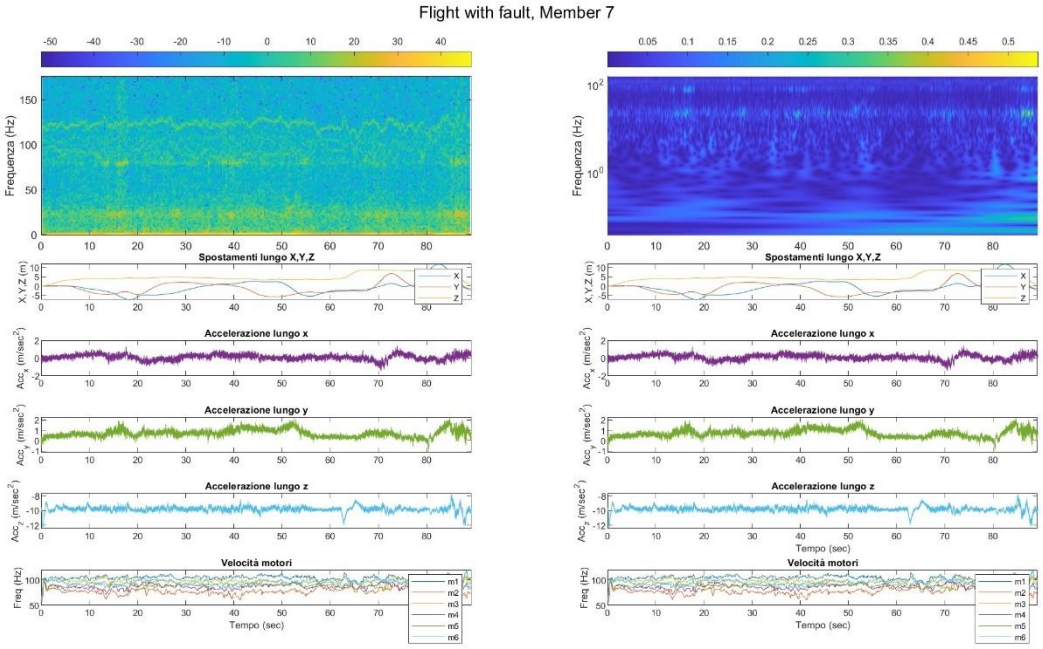


*Figura 5.9: Confronto Member 7*

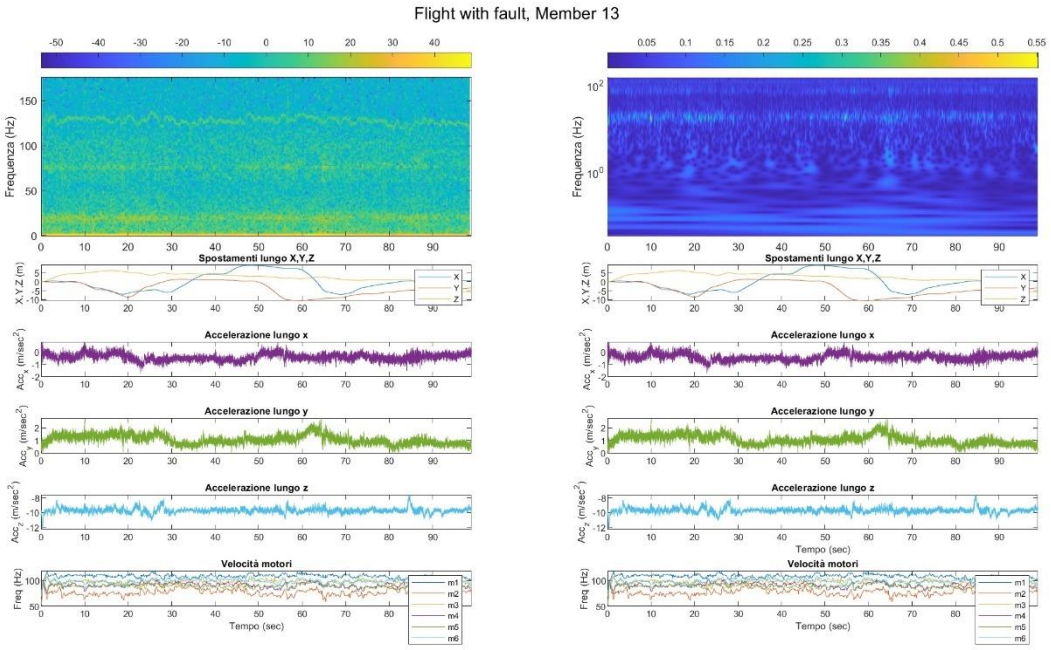


*Figura 5.10: Confronto Member 13*

Si nota che per le zone a maggior densità nel diagramma tempo-frequenza della CWT, sono solitamente relative a strappi di accelerazione o a spostamenti repentini del drone. Si può notare inoltre come nello spettrogramma, sia presente un contributo dipendente dalle sei velocità dei motori, mediato all'interno della finestra scorrevole di calcolo della STFT.

Concludendo che l'analisi delle immagini non porta a risultati, si è scelto di proseguire con una soluzione alternativa che verrà presentata nella prossima sezione.

## 5.6 Media potenze spettrogrammi

L'obiettivo è quello di fare classificazione utilizzando il contenuto informativo ricavato dai grafici precedenti. Nello specifico, sono stati mediati i dati complessi dello spettrogramma rispetto alle frequenze, così da ottenere un unico valore medio che racchiuda il contenuto informativo relativo a una singola frequenza. Dal momento che i campioni presenti nello spettrogramma sono complessi, è stato necessario usare il modulo e successivamente farne la media.

Prima di stampare i dati, si è notato che il contenuto informativo effettivamente interessante è posizionato da una certa frequenza in poi, poiché le basse frequenze sono influenzate da lievi vibrazioni strutturali o fattori ambientali, quindi sono state scartate le frequenze sotto i 50Hz.

I risultati vengono riportati poi in istogrammi (come fatto prima, sono mostrati tre istogrammi relativi ai primi tre voli di ogni tipologia di fault, *Fig. 5.11-5.13*).

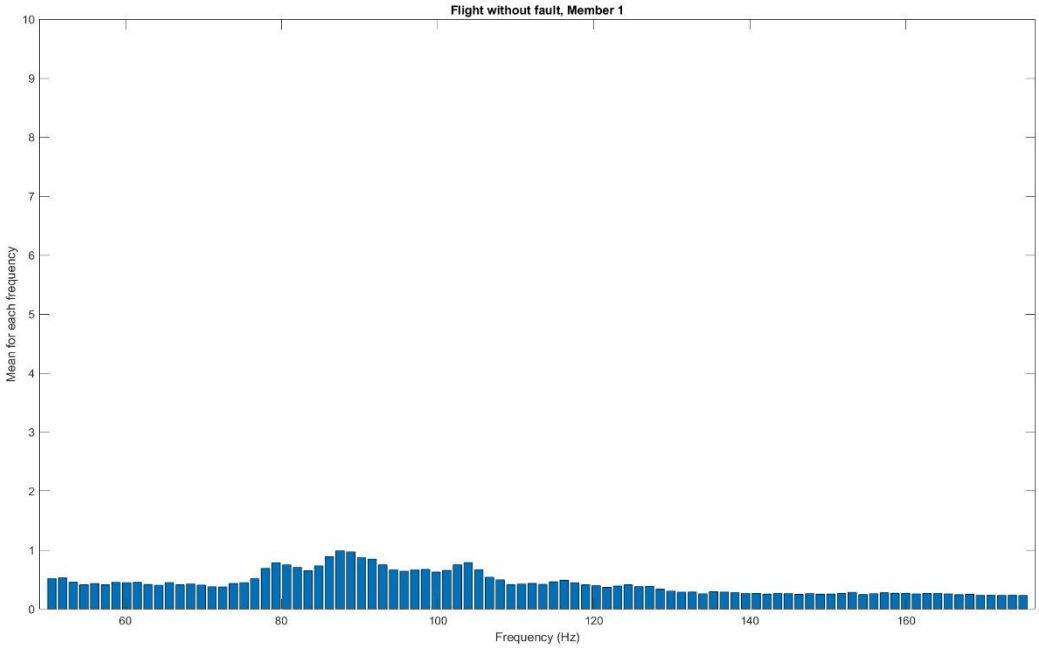


*Figura 5.11: Media frequenze Member 1*

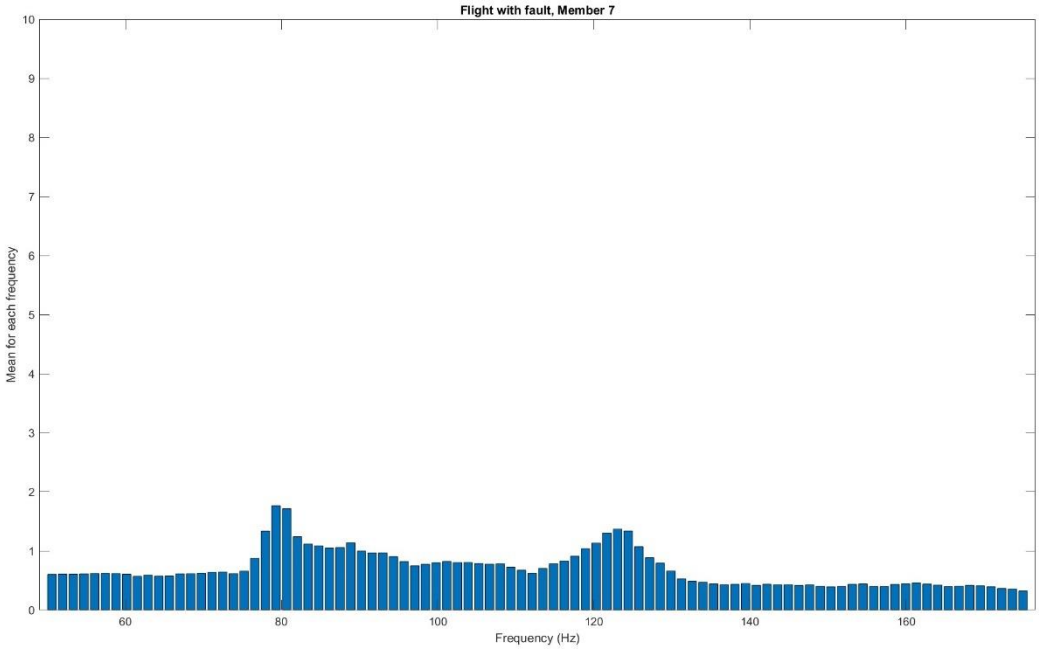


*Figura 5.12: Media frequenze Member 7*

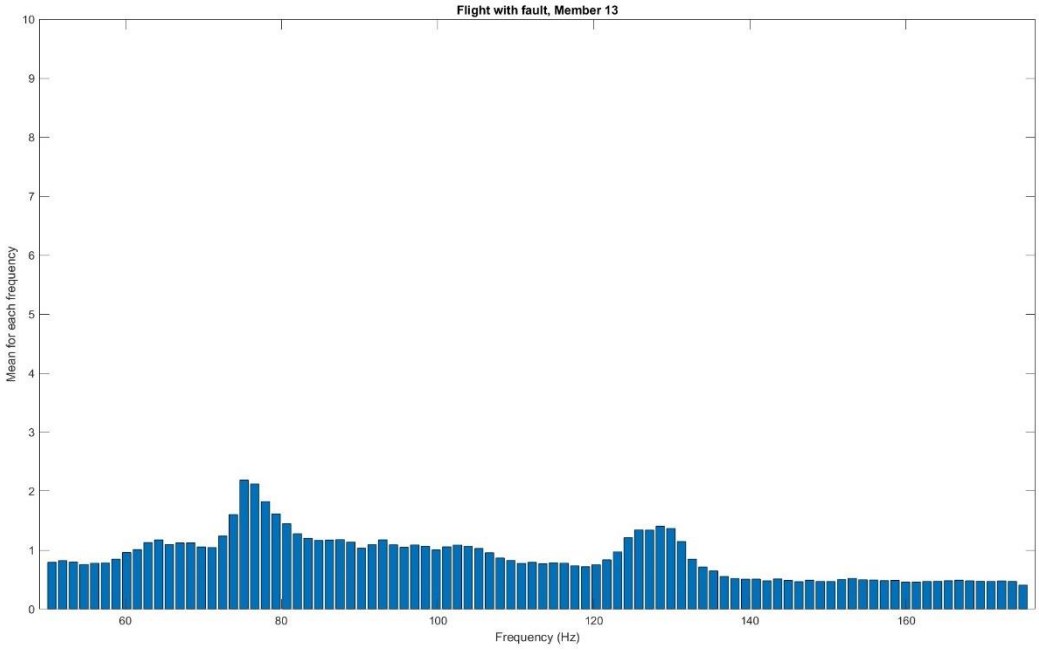


*Figura 5.13: Media frequenze Member 13*

Andando a sovrapporre questi tre grafici si nota chiaramente come i barplot dei Member con fault siano mediamente più alti rispetto a quelli senza fault. Possiamo notare il volo 1 in verde (nofault), il volo 7 (fault 5%) e il volo 13 (fault 10%), come riportato in *Fig. 5.14.*

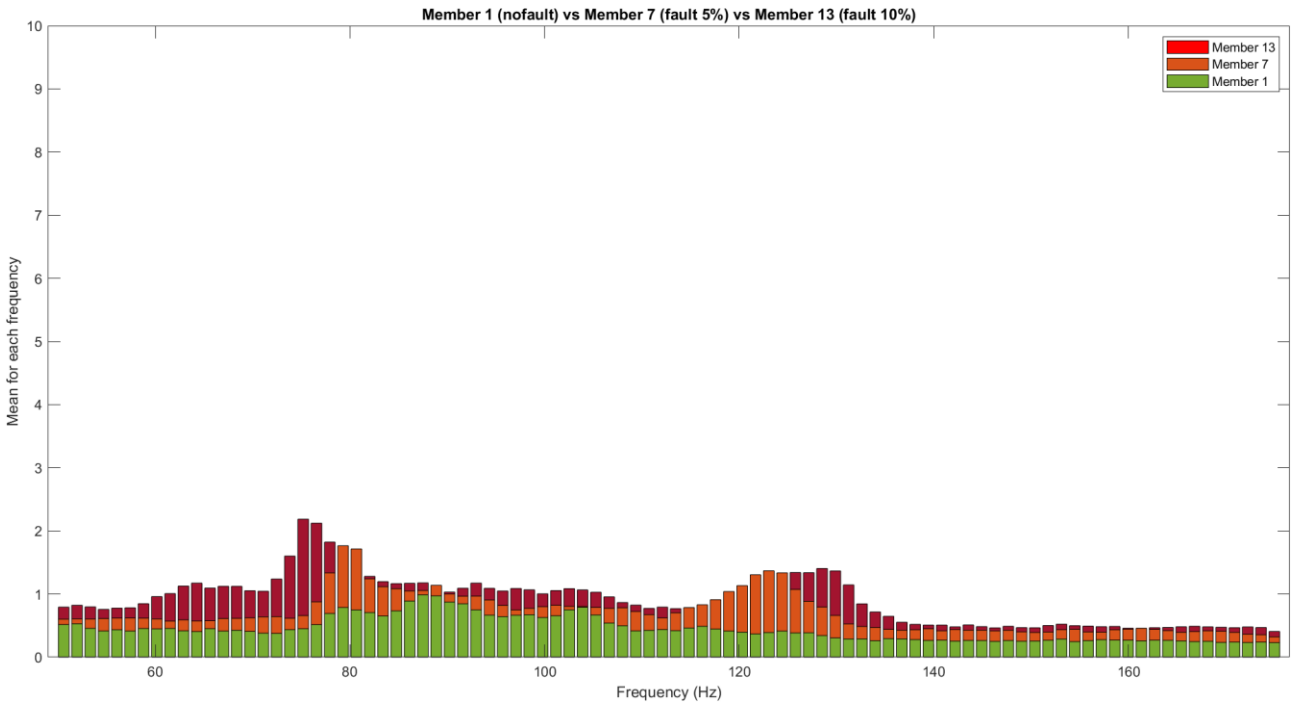


*Figura 5.14: Sovrapposizione Member*

Questo osservazione può essere fatta controllando e sovrapponendo anche altri voli, di cui uno per ogni categoria.

## 5.7 Addestramento classificatori

È stata generata una nuova feature Table, questa volta composta da due soli predittori, ovvero il valore dell'intensità media per ogni frequenza e le frequenze stesse. L'obiettivo è andare a fare classificazione con questi dati e valutare se è possibile addestrare dei *Decision Tree* e ottenere buoni risultati. La scelta di questi algoritmi è dovuta alla loro facile interpretabilità tramite le regole di decisione e potenzialmente idonei per applicazioni in tempo reale, quando si tratta di risorse computazionali limitate.

Verranno addestrati classificatori, considerando dapprima il volo intero, poi effettuando un binning spettrale e infine considerando porzioni di volo.

### 5.7.1 Volo intero

*Nelle Tab. 5.1 e 5.2* vengono riportati i risultati ottenuti addestrando un Fine Tree, un Medium Tree e un Coarse Tree. È stato partizionato il dataset in 80-20% per la fase di train e test, con un approccio k-fold cross validation con k=5.

| Decision Tree fault – 0 - 5 - 10 | |
|---|---|
| **Tree** | **Accuracy (Test)** |
| Fine Tree | **70.0%** |
| Medium Tree | 69.0% |
| Coarse Tree | 57.7% |

*Tabella 5.1: Test Accuracy volo intero, caso 0-5-10*

| Decision Tree fault - nofault | |
|---|---|
| **Tree** | **Accuracy (Test)** |
| Fine Tree | **74.4%** |
| Medium Tree | 72.8% |
| Coarse Tree | **74.4%** |

*Tabella 5.2: Test Accuracy volo intero, caso fault-nofault*

Nelle *Fig. 5.15 e 5.16* sono riportate le rispettive matrici di confusione per il solo *Fine Tree* (Best Case)

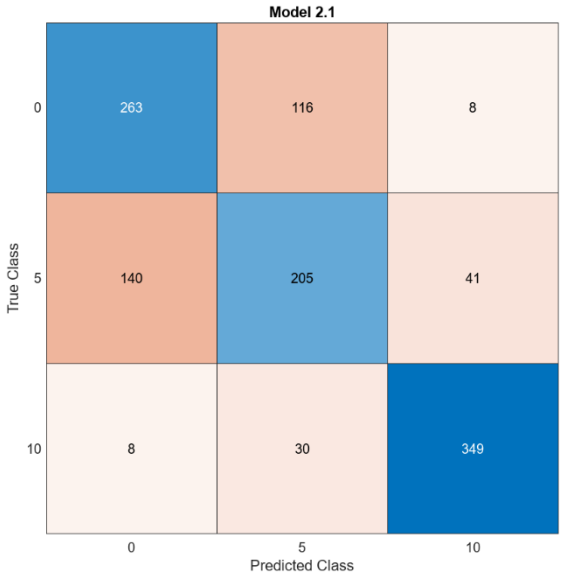


*Figura 5.15: Matrice di confusione Fine Tree*

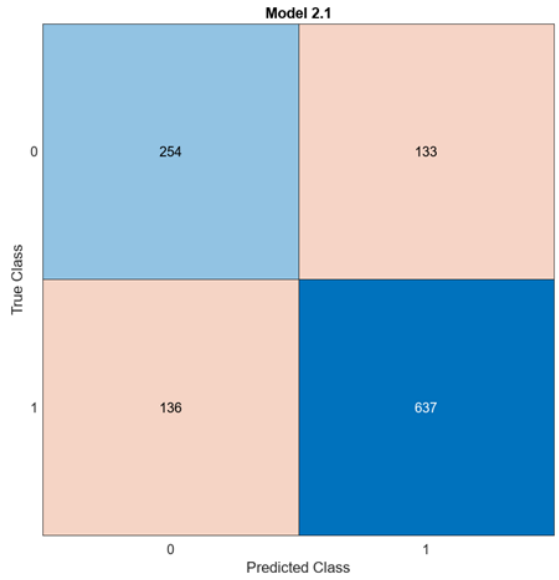


*Figura 5.16: Matrice di confusione Fine Tree*

### 5.7.2 Spectral binning

In questa sezione è stato fatto un ulteriore test, simile al precedente, ma raggruppando le frequenze per 4 e successivamente per 8.

**Raggruppamento per 4**

Il risultato è illustrato in *Fig. 5.17*:

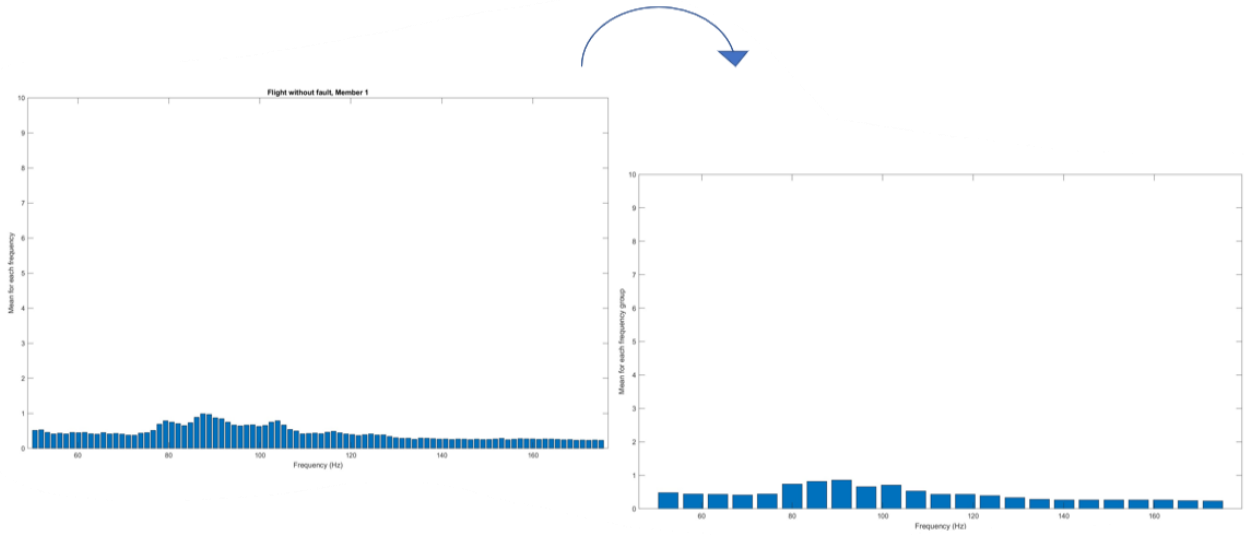


*Figura 5.17: Spectral binning Member 1, raggruppamento per 4*

Adesso è stato possibile addestrare gli stessi alberi del test precedente, con le stesse modalità e approccio.

I risultati sono riportati in *Tab.5.3 e 5.4*:

| Decision Tree fault – 0 - 5 - 10 | |
|---|---|
| **Tree** | **Accuracy (Test)** |
| Fine Tree | 67.7% |
| Medium Tree | **71.0%** |
| Coarse Tree | 58.1% |

*Tabella 5.3: Test Accuracy volo intero, caso 0-5-10*

| Decision Tree fault - nofault | |
|---|---|
| **Tree** | **Accuracy (Test)** |
| Fine Tree | **77.4%** |
| Medium Tree | 76.6% |
| Coarse Tree | 72.6% |

*Tabella 5.4: Test Accuracy volo intero, caso fault-nofault*

Nelle *Fig. 5.18 e 5.19* sono riportate le rispettive matrici di confusione per i soli *Medium Tree* e *Fine Tree* (Best Cases)

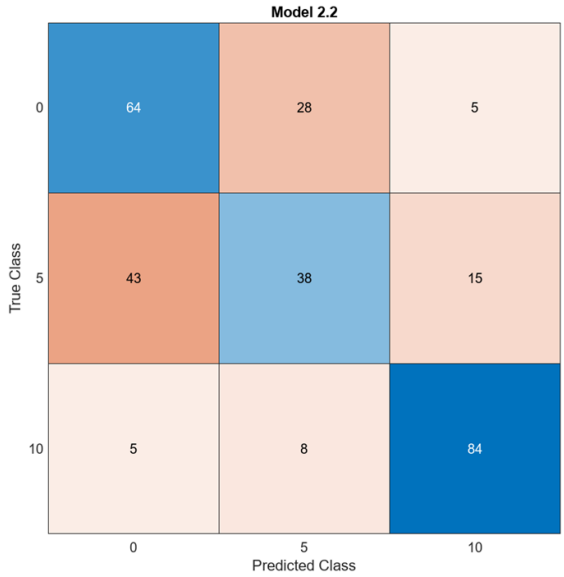


*Figura 5.18: Matrice di confusione Medium Tree*

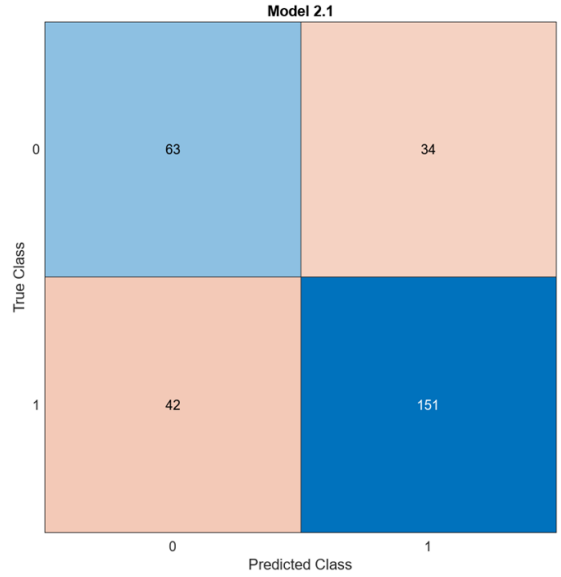


*Figura 5.19: Matrice di confusione Fine Tree*

**Raggruppamento per 8**

Una volta raggruppate le frequenze per 8, è stato possibile addestrare gli stessi alberi del test precedente, con le stesse modalità e approccio.

I risultati sono riportati in *Tab.5.5 e 5.6*:

| Decision Tree fault – 0 - 5 - 10 | |
|---|---|
| **Tree** | **Accuracy (Test)** |
| Fine Tree | **59.7%** |
| Medium Tree | **59.7%** |
| Coarse Tree | 47.5% |

*Tabella 5.5: Test Accuracy volo intero, caso 0-5-10*

| Decision Tree fault - nofault | |
|---|---|
| **Tree** | **Accuracy (Test)** |
| Fine Tree | **78.0%** |
| Medium Tree | **78.0%** |
| Coarse Tree | 72.9% |

*Tabella 5.6: Test Accuracy volo intero, caso fault-nofault*

Nelle *Fig. 5.20 e 5.21* sono riportate le rispettive matrici di confusione per il solo *Fine Tree* (Best Case)

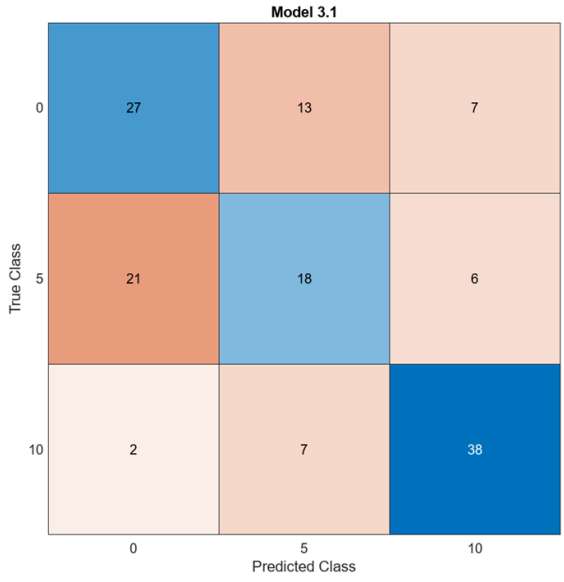


*Figura 5.20: Matrice di confusione Fine Tree*

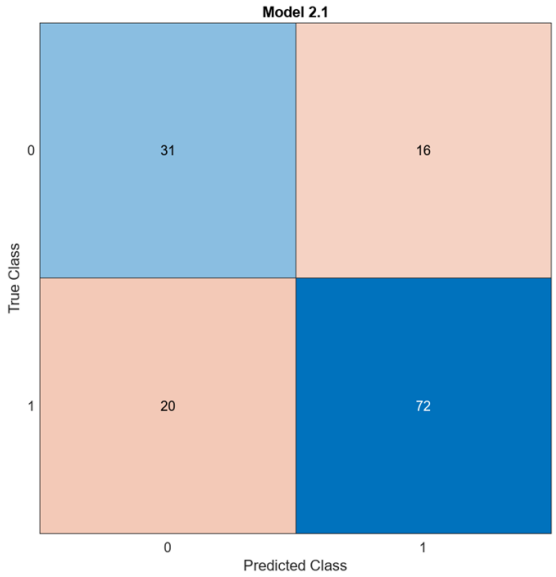


*Figura 5.21: Matrice di confusione Fine Tree*

### 5.7.3 Partizionamento voli

In questa sezioni sono infine valutate le prestazioni degli alberi di decisione facendo riferimento a porzioni di volo. Nello specifico partizionando il volo prima in due parti, poi in quattro.

I partizionamenti sono stati ricavati con uno script Matlab, che ricavasse gli indici per dividere il segnale *acc_y* in due o in quattro parti, arrotondando all'intero più alto in caso di numeri decimali. Successivamente questi indici sono stati usati per frazionare il vettore del segnale e addestrare i tre alberi decisionali.

I risultati sono riportati nelle *Tab. 5.7 e 5.8* per il partizionamento in due parti e nelle *Tab. 5.9 e 5.10* per quello in quattro parti.

| 1/2 | Decision Tree fault – 0 - 5 – 10 | |
|---|---|---|
| | **Tree** | **Accuracy (Test)** |
| | Fine Tree | **64.9%** |
| | Medium Tree | 64.3% |
| | Coarse Tree | 59.3% |

| 2/2 | Decision Tree fault – 0 - 5 – 10 | |
|---|---|---|
| | **Tree** | **Accuracy (Test)** |
| | Fine Tree | **70.6%** |
| | Medium Tree | 67.1% |
| | Coarse Tree | 53.2% |

*Tabella 5.7: Accuracy volo partizionato, caso 0-5-10*

| 1/2 | Decision Tree fault - nofault | |
|---|---|---|
| | **Tree** | **Accuracy (Test)** |
| | Fine Tree | **71.2%** |
| | Medium Tree | 69.6% |
| | Coarse Tree | 70.2% |

| 2/2 | Decision Tree fault - nofault | |
|---|---|---|
| | **Tree** | **Accuracy (Test)** |
| | Fine Tree | 77.0% |
| | Medium Tree | **79.8%** |
| | Coarse Tree | 70.0% |

*Tabella 5.8: Accuracy volo partizionato, caso fault-nofault*

| | Decision Tree fault – 0 - 5 - 10 | |
|---|---|---|
| 1/4 | **Tree** | **Accuracy (Test)** |
| | Fine Tree | 67.7% |
| | Medium Tree | **68.8%** |
| | Coarse Tree | 57.3% |
| 2/4 | **Tree** | **Accuracy (Test)** |
| | Fine Tree | 62.1% |
| | Medium Tree | **67.1%** |
| | Coarse Tree | 57.7% |
| 3/4 | **Tree** | **Accuracy (Test)** |
| | Fine Tree | 66.5% |
| | Medium Tree | **68.3%** |
| | Coarse Tree | 53.4% |
| 4/4 | **Tree** | **Accuracy (Test)** |
| | Fine Tree | **69.0%** |
| | Medium Tree | 66.1% |
| | Coarse Tree | 51.4% |

*Tabella 5.9: Accuracy volo partizionato, caso 0-5-10*

| | Decision Tree fault - nofault | |
|---|---|---|
| 1/4 | **Tree** | **Accuracy (Test)** |
| | Fine Tree | **76.0%** |
| | Medium Tree | 76.0% |
| | Coarse Tree | 71.8% |
| 2/4 | **Tree** | **Accuracy (Test)** |
| | Fine Tree | 68.8% |
| | Medium Tree | 72.2% |
| | Coarse Tree | **72.6%** |
| 3/4 | **Tree** | **Accuracy (Test)** |
| | Fine Tree | 74.8% |
| | Medium Tree | **79.2%** |
| | Coarse Tree | 74.4% |
| 4/4 | **Tree** | **Accuracy (Test)** |
| | Fine Tree | **77.4%** |
| | Medium Tree | 76.8% |
| | Coarse Tree | 73.2% |

*Tabella 5.10: Accuracy volo partizionato, caso fault-nofault*

# Conclusioni e sviluppi futuri

La tesi ha avuto come scopo quello di effettuare fault detection e isolation utilizzando il dataset UAV-FD. I primi test che sono stati svolti, consistevano in un'analisi di feature temporali e nel dominio della frequenza, per poi generare con queste un unico database da dividere in train set e test set. Sono stati addestrati cinque differenti classificatori. Nella seconda parte è stata fatta un'analisi in frequenza generando lo spettrogramma di un segnale selezionato, per poi proseguire mediando i dati complessi rispetto alle singole frequenze. I risultati ottenuti sono stati prima graficati in grafici a barre e successivamente utilizzati come predittori per il train e test di alberi decisionali.

Nella prima parte sono stati raggiunti ottimi risultati, ottenendo valori di accuracy test elevati nella quasi totalità del casi. L'esito finale del lavoro purtroppo si discosta leggermente dalla previsione fatta inizialmente, difatti l'idea iniziale era di tentare l'utilizzo di spettrogrammi / scalogrammi per fare image classification: questa strada è stata accantonata per via della scarsità del dataset e la forte dipendenza dei voli dalle traiettorie e accelerazioni, preferendo poi un approccio in frequenza a partire dall'analisi degli stessi spettrogrammi.

Le differenze tra l'analisi della prima parte, ove sono state generate feature nel dominio del tempo e nel dominio della frequenza contro l'analisi tempo-frequenza sono evidenti, in quanto la prima ha restituito risultati migliori.

Soffermandosi sulla seconda parte, si può affermare che, un problema che ha afflitto l'analisi tempo-frequenza, è sicuramente lo sbilanciamento del dataset nel caso fault-nofault. Questo tuttavia è un problema noto in ambito di manutenzione dove spesso i dati contenenti guasto sono scarni rispetto ai dati di normale funzionamento.

L'idea di troncare il dataset è stata accantonata poiché i dati, come detto, non erano molti, soprattutto dopo aver mediato le frequenze.

Un'altra osservazione, può essere fatta considerando i valori ottenuti per gli accuracy test nella seconda fase: sono stati mediamente più elevati nei casi in cui si è ristretto il problema al caso binario, difatti risulta più complesso classificare le classi di guasto tra 5% e 10%, soffrendo il drone di vibrazioni maggiori in entrambi i casi, non sempre distinguibili nitidamente dall'algoritmo.

Si noti inoltre che, anche utilizzando frazioni di volo, i risultati non sono stati mai troppo diversi in relazione al volo intero. Per quanto riguarda lo spectral binning i risultati sono peggiorati, specialmente per il caso di raggruppamento per 8.

Per quanto riguarda gli sviluppi futuri, si pensa che avendo a disposizione un dataset più ampio, eventualmente generando dati di volo da un simulatore, sarà possibile provare ad utilizzare algoritmi di classificazione direttamente sulle immagini degli spettrogrammi o degli scalogrammi, in quanto si riuscirebbero a coprire molte più situazioni di volo, etichettarle correttamente, e tentare di addestrare un classificatore basato su immagini.

# Appendice

## A.1 - Plot confronti spectrogram – CWT

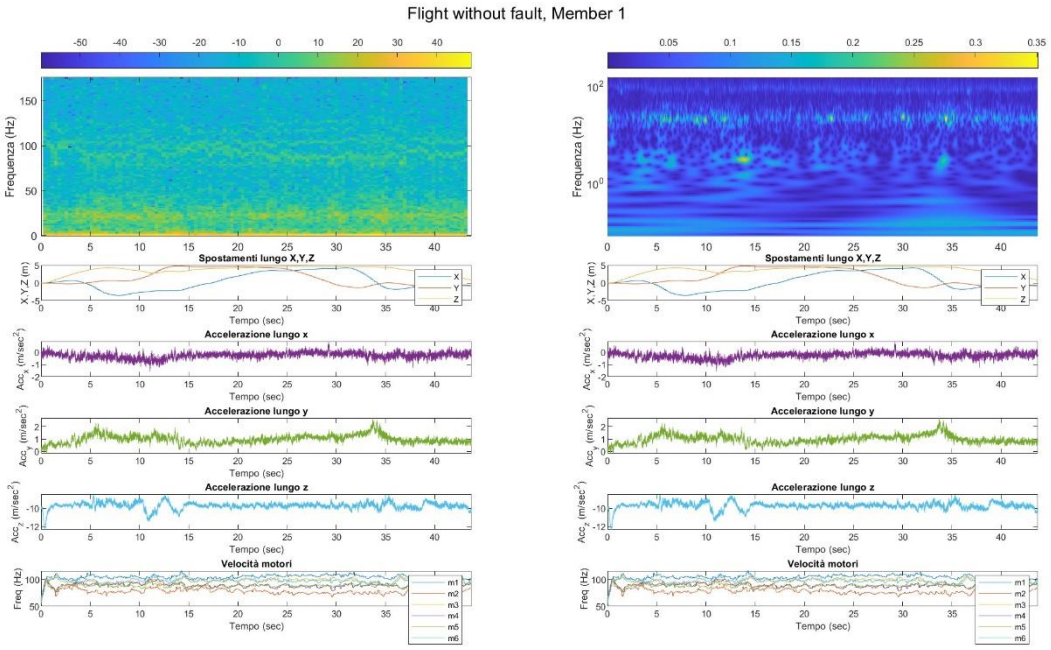


*Figura A.1: confronti spettrogramma – scalogramma, Member 1*

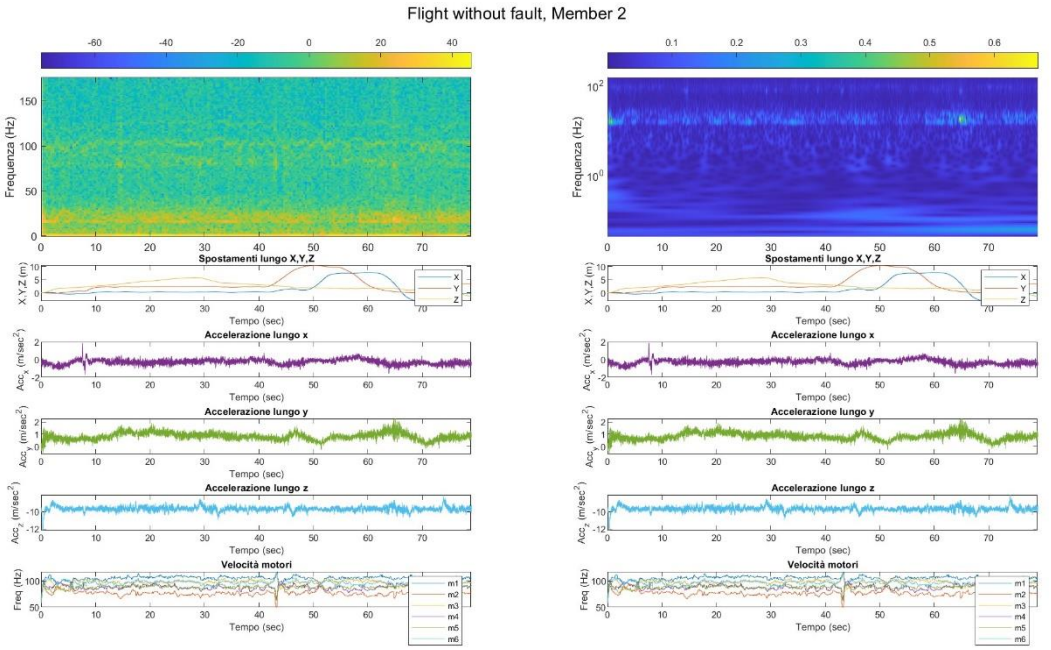


*Figura A.2: confronti spettrogramma – scalogramma, Member 2*

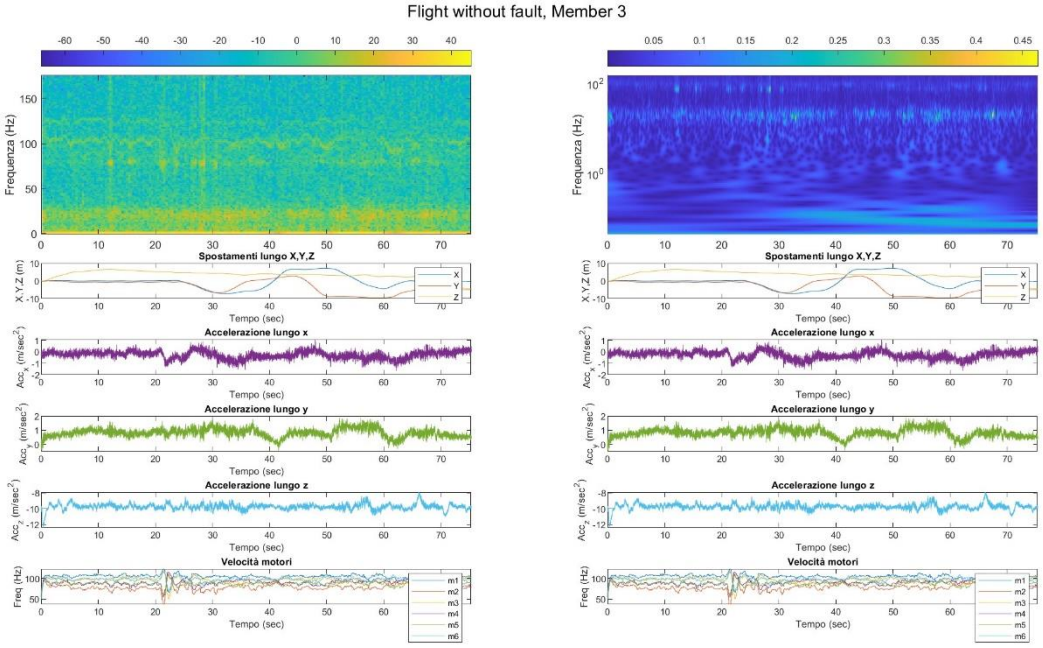


*Figura A.3: confronti spettrogramma – scalogramma, Member 3*

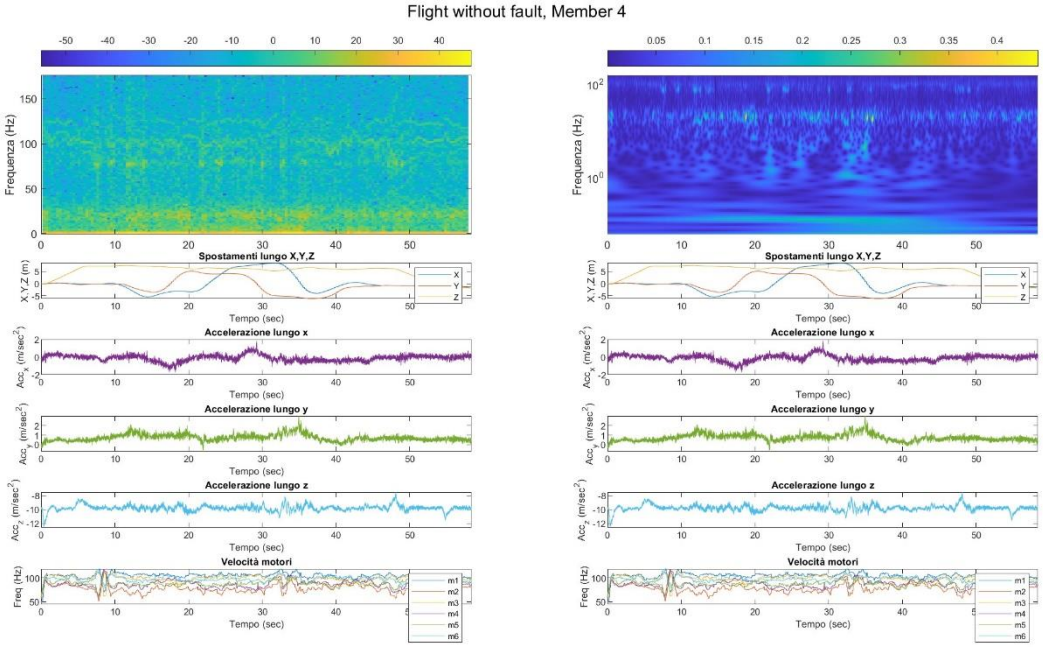


*Figura A.4: confronti spettrogramma – scalogramma, Member 4*

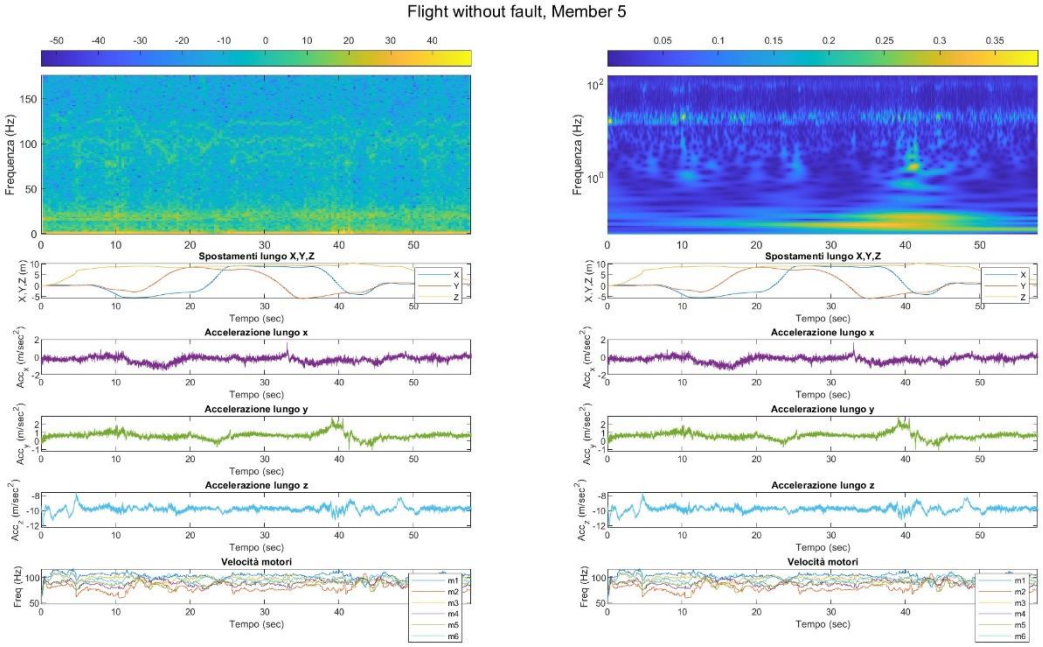


*Figura A.5: confronti spettrogramma – scalogramma, Member 5*

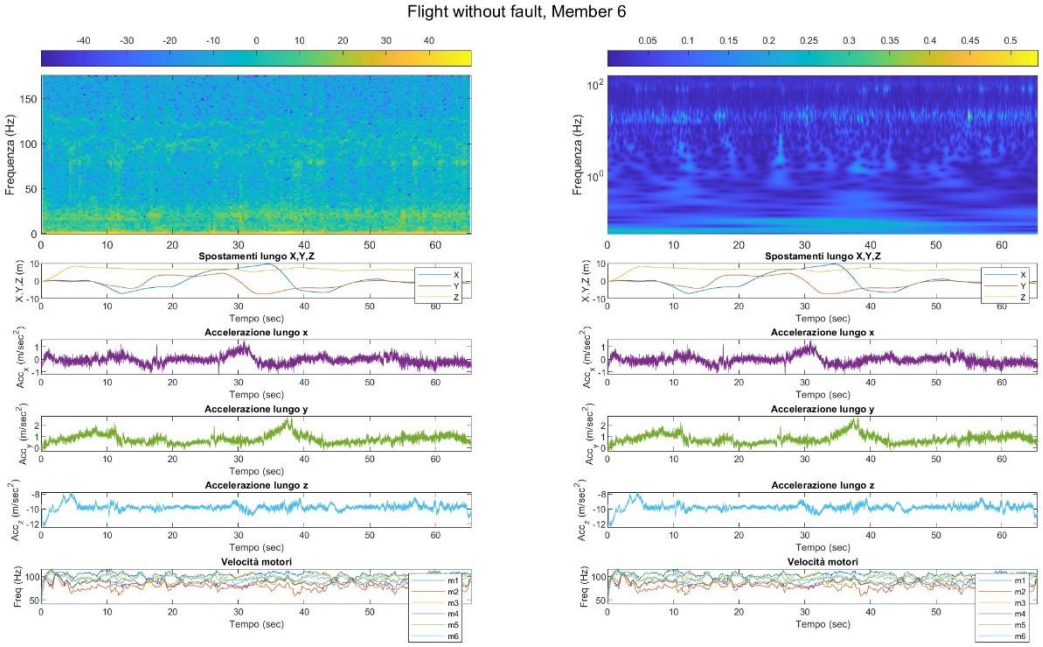


*Figura A.6: confronti spettrogramma – scalogramma, Member 6*

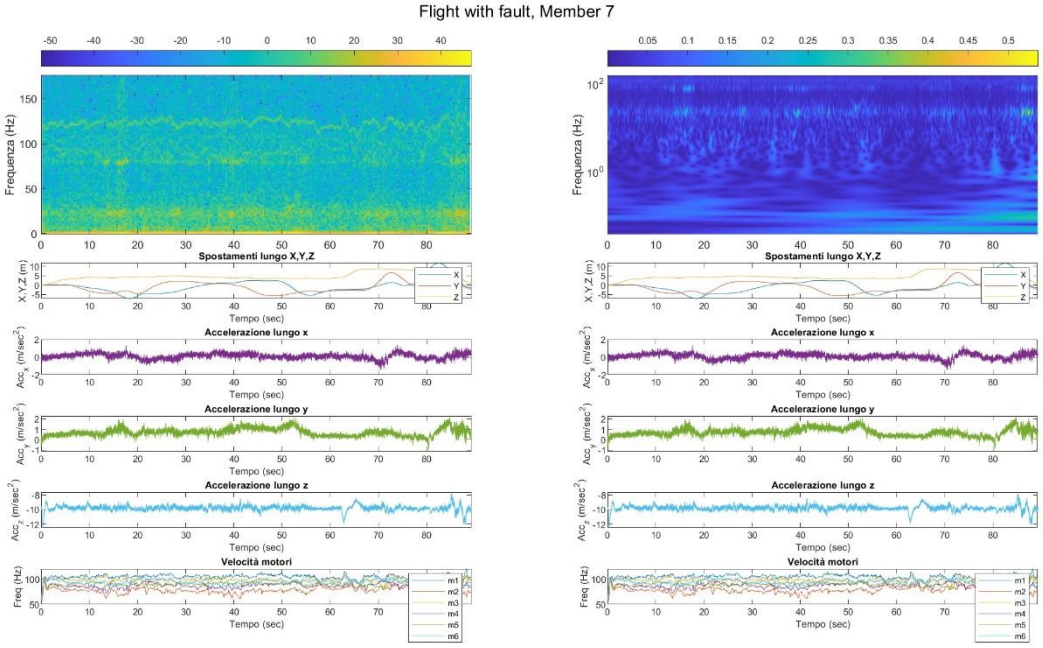


*Figura A.7: confronti spettrogramma – scalogramma, Member 7*

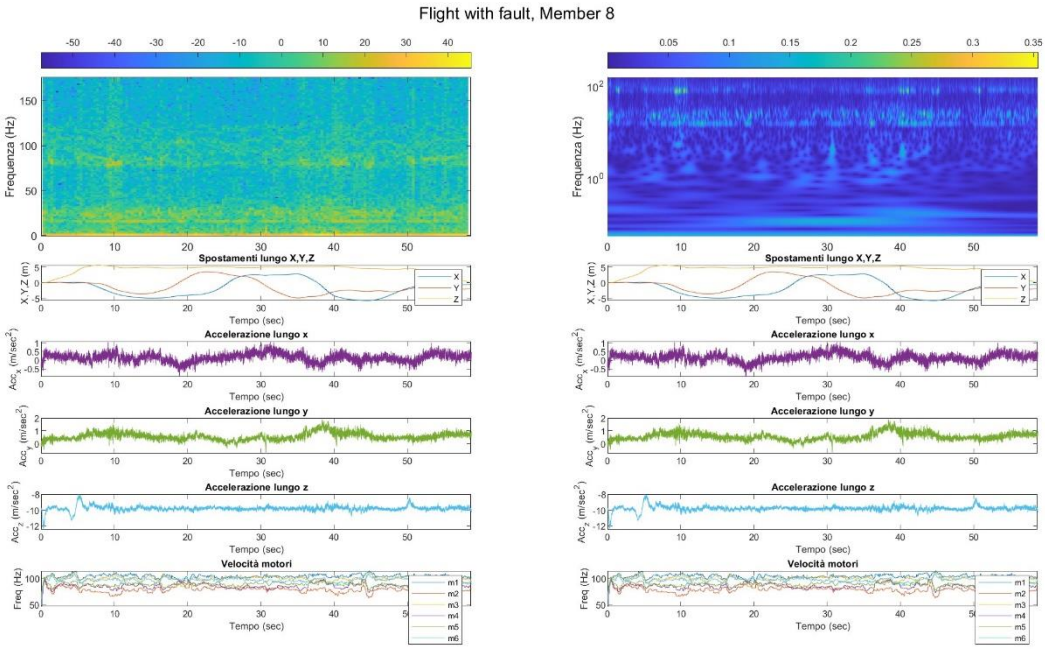


*Figura A.8: confronti spettrogramma – scalogramma, Member 8*

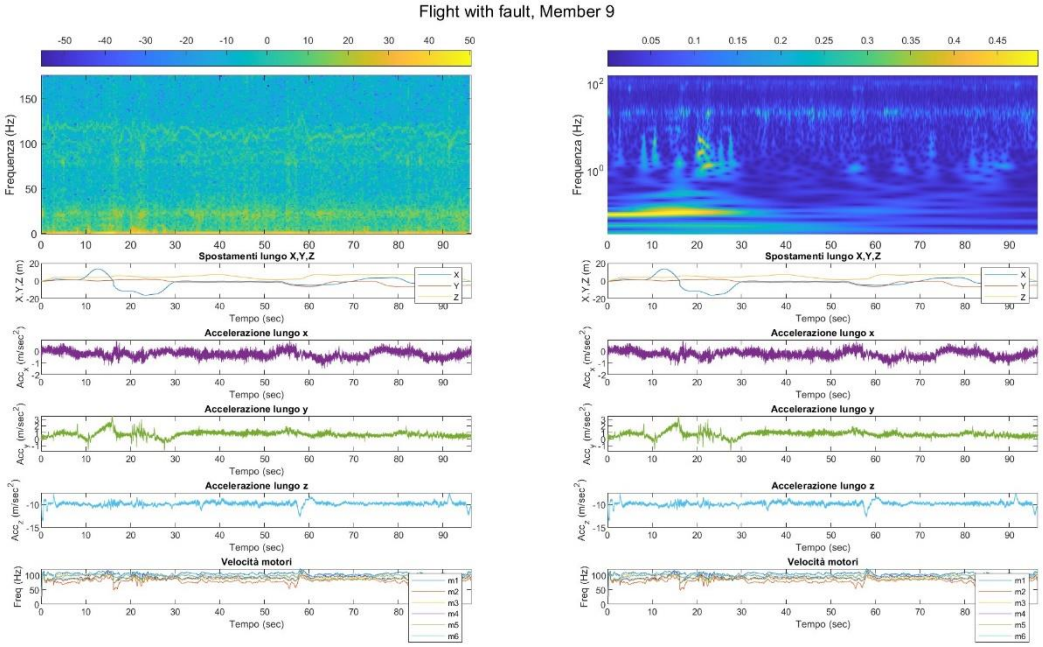


*Figura A.9: confronti spettrogramma – scalogramma, Member 9*

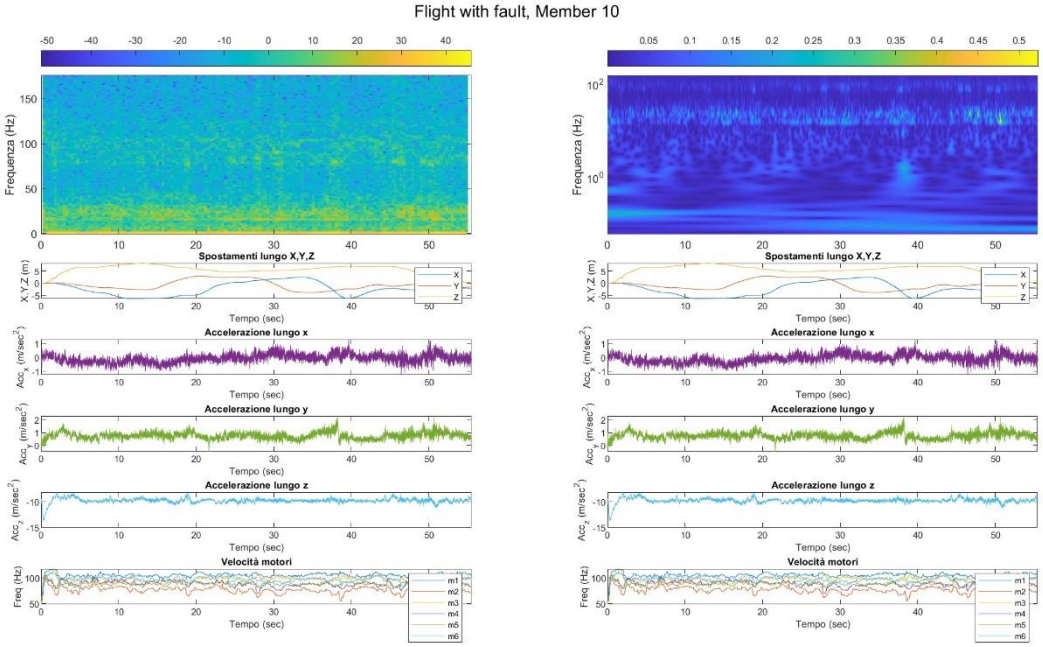


*Figura A.10: confronti spettrogramma – scalogramma, Member 10*

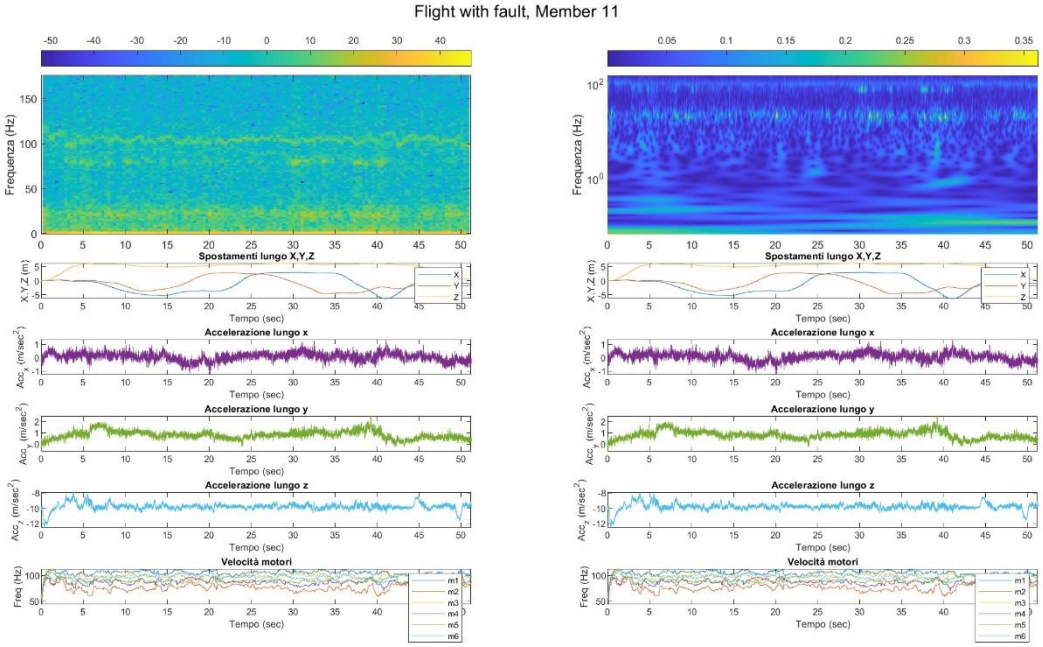


*Figura A.11: confronti spettrogramma – scalogramma, Member 11*

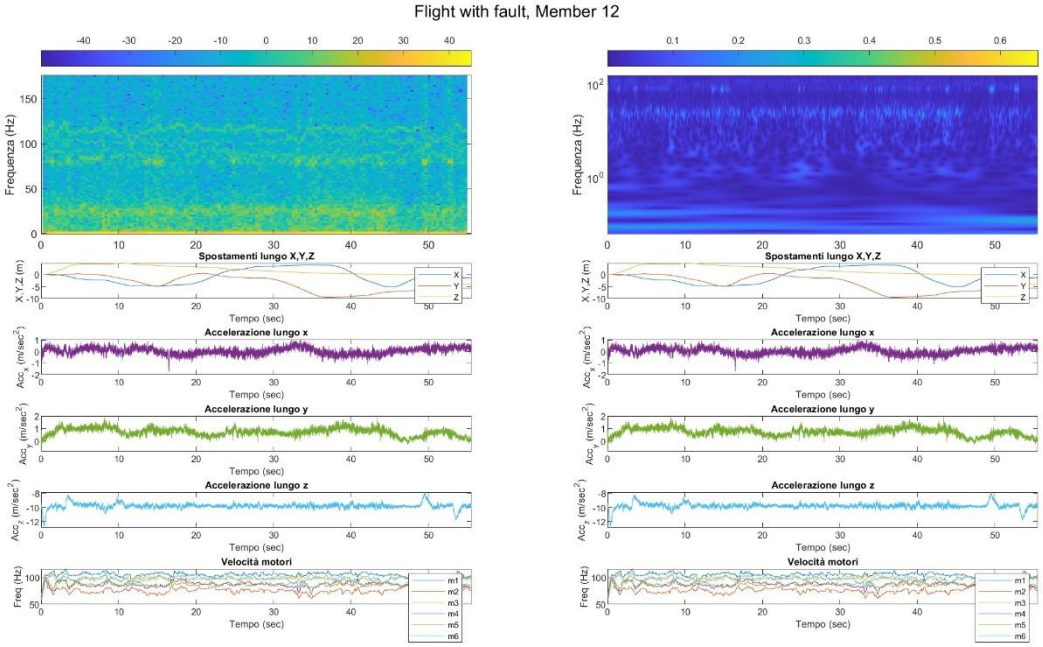


*Figura A.12: confronti spettrogramma – scalogramma, Member 12*

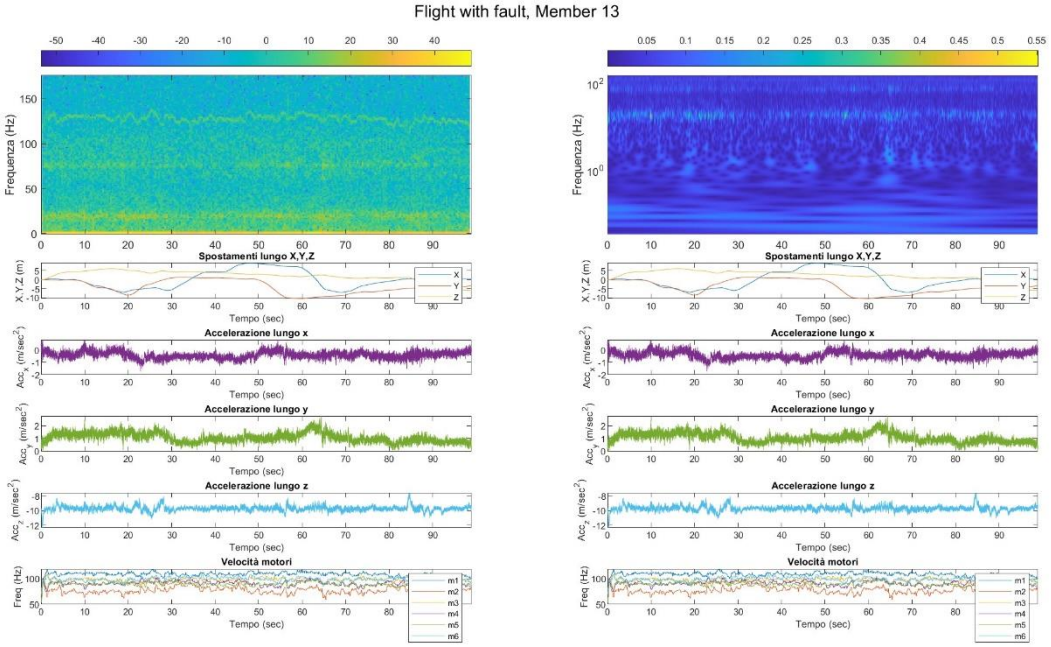


*Figura A.13: confronti spettrogramma – scalogramma, Member 13*

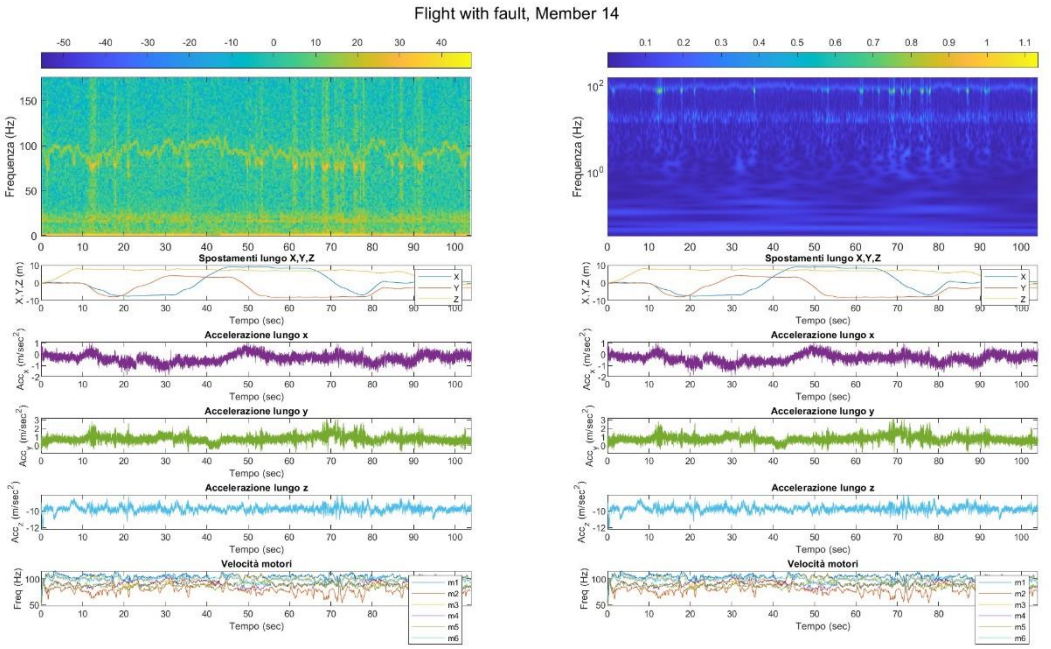


*Figura A.14: confronti spettrogramma – scalogramma, Member 14*

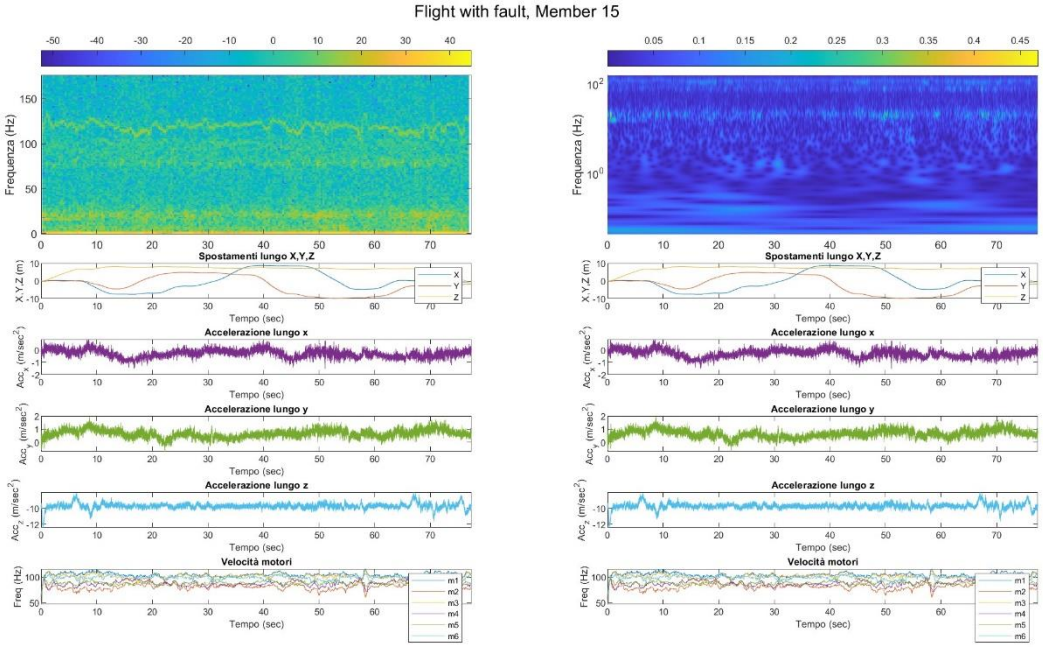


*Figura A.15: confronti spettrogramma – scalogramma, Member 15*

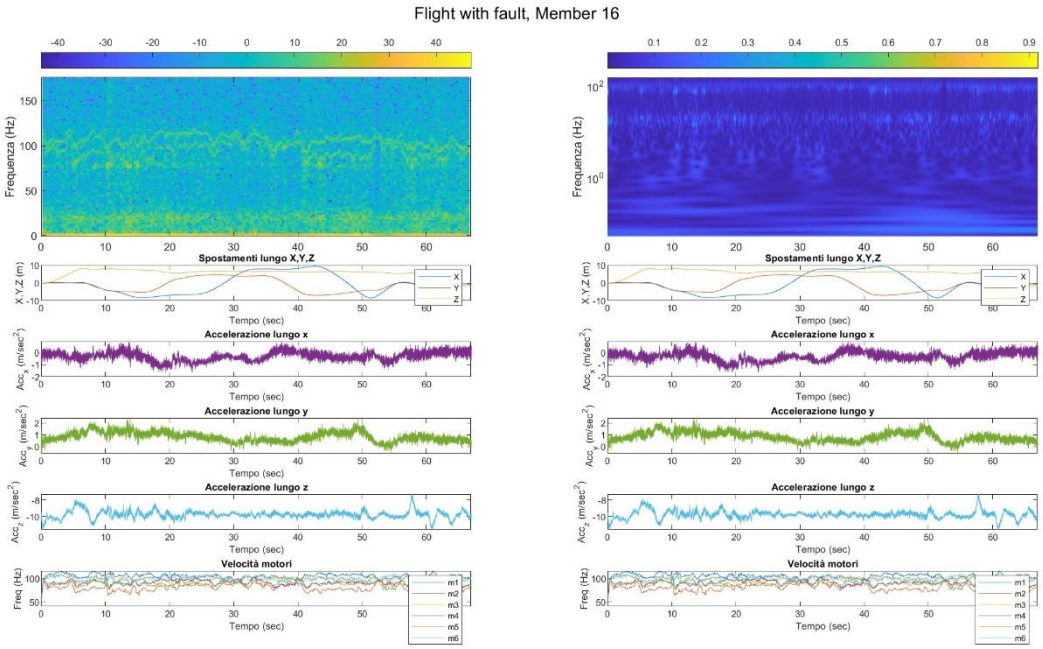


*Figura A.16: confronti spettrogramma – scalogramma, Member 16*

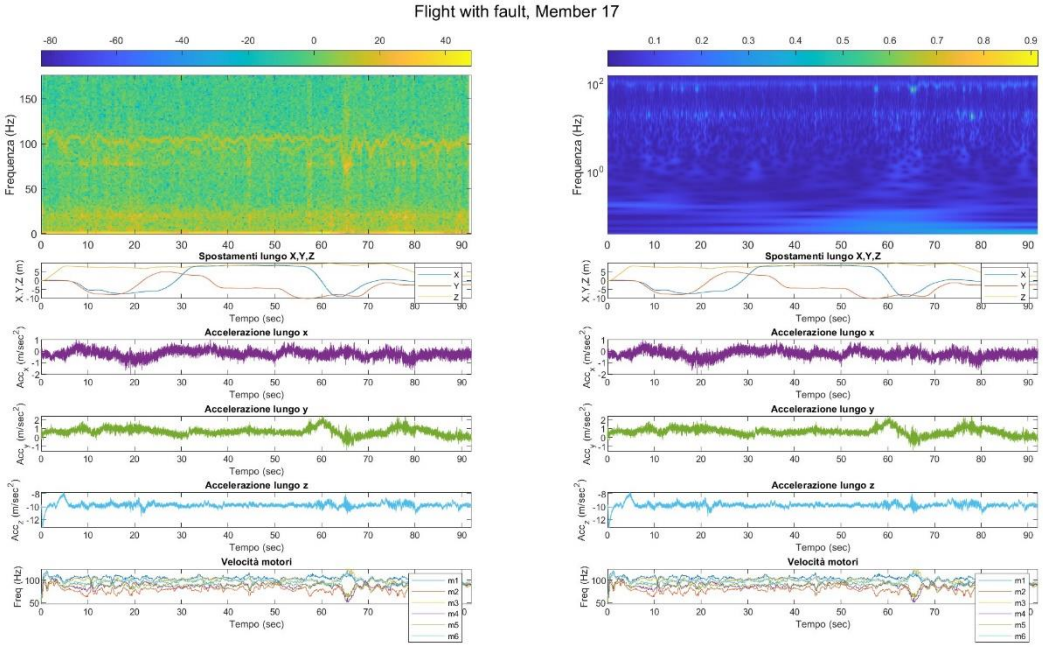


*Figura A.17: confronti spettrogramma – scalogramma, Member 17*

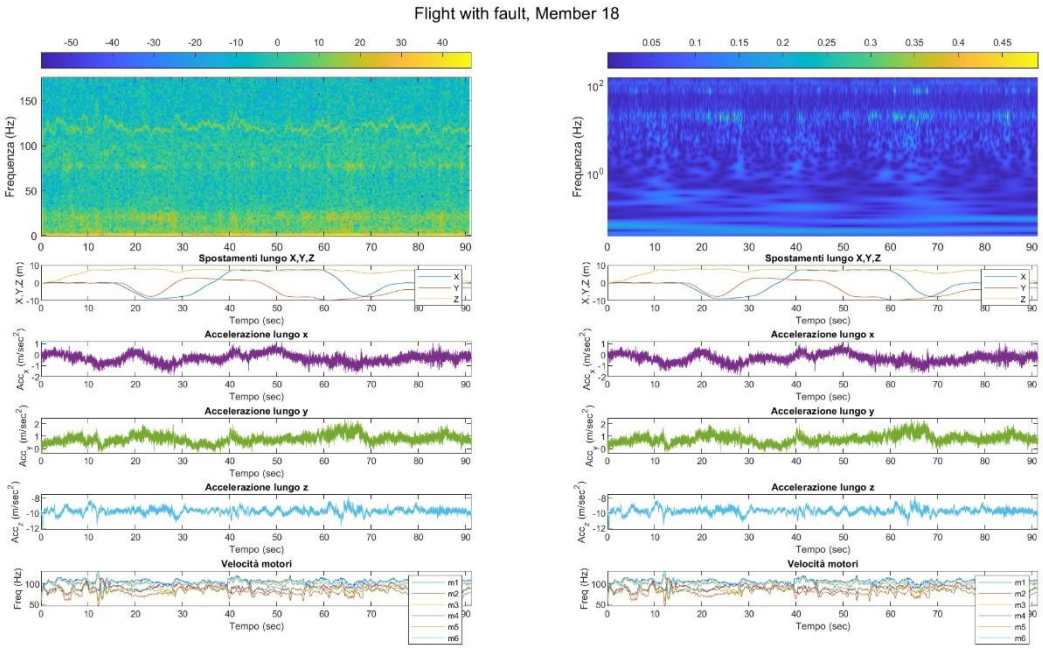


*Figura A.18: confronti spettrogramma – scalogramma, Member 18*

# Bibliografia


[1] Isermann, R. “Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance”, Springer Berlin Heidelberg, 2006

[2] Jeyan, Dr and P S, Ramesh “Comparative analysis of the impact of operating parameters on military and civil applications of mini unmanned aerial vehicle (UAV)”. AIP Conference Proceedings, 2020

[3] Grigore, Laurențiu & Cristescu, Cristian “The Use of Drones in Tactical Military Operations in the Integrated and Cybernetic Battlefield”, Land Forces Academy Review, 2024

[4] European Commission “Drones: The Future of Transportation and Surveillance”, Brussels, 29/11/2022

[5] A. Baldini, L. D’Alleva, R. Felicetti, F. Ferracuti, A. Freddi and A. Monteriù, "UAV-FD: a dataset for actuator fault detection in multirotor drones", International Conference on Unmanned Aircraft Systems (ICUAS), 2023

[6] Puchalski, R.; Giernacki, W. "UAV Fault Detection Methods, State-of-the-Art", Drones, 2022

[7] https://ardupilot.org/copter/docs/common-downloading-and-analyzing-data-logs-in-mission-planner.html - ultima consultazione Novembre 2024

[8] https://it.mathworks.com/help/predmaint/ref/diagnosticfeaturedesigner-app.html - ultima consultazione Novembre 2024

[9] https://it.mathworks.com/help/stats/classification-learner-app.html - ultima consultazione Novembre 2024

[10] Travaglini D. “Trasformazioni tra sistemi di coordinate: software disponibili, limiti e potenzialità”, Forest, 2008

[11] Martinez, Edwin Mauricio, Pedro Ponce, Israel Macias, and Arturo Molina, "Automation Pyramid as Constructor for a Complete Digital Twin, Case Study: A Didactic Manufacturing System", Sensors, 2021

[12] https://it.mathworks.com/help/signal/ref/spectrogram.html - ultima consultazione Novembre 2024

[13] https://it.mathworks.com/help/wavelet/ref/cwt.html - ultima consultazione Novembre 2024

[14] https://it.mathworks.com/help/wavelet/ug/time-frequency-analysis-and-continuous-wavelet-transform.html - ultima consultazione Novembre 2024

[15] Zhang, D. “Fundamentals of Image Data Mining: Analysis, Features, Classification and Retrieval”, Springer, 2019

[16] https://zenodo.org/records/7648996 - ultima consultazione Novembre 2024

[17] B. Wang, D. Liu, Y. Peng and X. Peng "Multivariate Regression-Based Fault Detection and Recovery of UAV Flight Data", IEEE Transactions on Instrumentation and Measurement, 2020

[18] Orhan Yaman, Ferhat Yol, Ayhan Altinors, “A Fault Detection Method Based on Embedded Feature Extraction and SVM Classification for UAV Motors” Microprocessors and Microsystems - Volume 94, 2022