UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in ingegneria informatica e dell'automazione

Sviluppo di classificatori per la diagnosi di guasti (Prog. B3 – PHM 2022 Data Challenge)

Studenti:
Riccardo MANCINI
Arment PELIVANI

Docente:
Prof. Alessandro Freddi

Anno Accademico 2023-2024

Indice

1 Introduzione **1**
1.1 Analisi del dataset 1
1.2 Pre-processing e conversione del dataset in formato MATLAB 2

2 Feature enginereeng **6**
2.1 Caricamento dati 6
2.2 Analisi qualitativa dei segnali nel dominio del tempo 7
2.3 Analisi qualitativa dei segnali nel dominio della frequenza 7
2.4 Estrazione delle Feature Temporali 9
2.5 Estrazione delle Feature Spettrali 11
2.6 Ranking delle Features 12

3 Addestramento e validazione **15**
3.1 Caricamento dati 15
3.2 Modelli di Classificazione 15
3.3 Cross-Validation e Risultati 17
3.3.1 Stratified Sampling 18

4 Test e analisi dei risultati **22**
4.1 Pre-processing e Feature enginereeng 22
4.2 Fase di test 22

5 Conclusione **25**

Elenco delle figure

1.1 Tipologie di fault e posizione dei sensori 2
1.2 Suddivisione dei dati per individui e fault 2
1.3 *main* del programma 3
1.4 Funzione *readDataBySensorName* 4
1.5 Funzione *createTimetable* 5

2.1 Tabelle relative ai 3 sensori nel Workspace di Matlab 6
2.2 Caricamento variabile relativa al sensore PDMP nel DFD 7
2.3 Grafico del segnale PDMP nel dominio del tempo 7
2.4 Grafico del segnale PIN nel dominio del tempo 8
2.5 Grafico del segnale PO nel dominio del tempo 8
2.6 Grafico del segnale spettrale PDMP in scala lineare 9
2.7 Grafico del segnale spettrale PIN in scala lineare 10
2.8 Grafico del segnale spettrale PO in scala lineare 10
2.9 Rank delle features per il segnale PDMP 12
2.10 Rank delle features per il segnale PIN 13
2.11 Rank delle features per il segnale PO 13
2.12 Funzione *replaceNaN* 14

3.1 Pannello di caricamento dati del Classification Learner 16
3.2 Risultati della cross-validation, a 5 fold, per tutti i modelli 17
3.3 Quadratic Discriminant 20
3.4 Ensemble Bagged Tree 20
3.5 Weighted KNN 20
3.6 Kernel Naive Bayes 20
3.7 Wide Neural Network 20
3.8 Cubic SVM 20
3.9 Fine Tree 20
3.10 Matrici di confusione 20
3.11 Esempio di implementazione della stratificazione nella cross-validation per il modello SVM 21

4.1 Risultati della cross-validation, a 5 fold, e del test sui 20 campioni con classe 1 23
4.2 Quadratic Discriminant 24
4.3 Ensemble Bagged Tree 24

4.4 Weighted KNN 24
4.5 Kernel Naive Bayes 24
4.6 Wide Neural Network 24
4.7 Cubic SVM 24
4.8 Fine Tree 24
4.9 Distribuzione delle predizioni di classe sui dati di test 24

Capitolo 1

Introduzione

L'obiettivo principale di questo progetto è sviluppare un classificatore diagnostico in grado di distinguere con precisione tra le diverse tipologie di guasto che possono verificarsi in una rocciatrice durante l'operatività. Questo strumento si rivela cruciale poiché, nella fase di perforazione, i sensori della macchina sono soggetti a intense sollecitazioni ambientali che ne alterano le misurazioni, rendendo difficile discernere tra un effettivo guasto e un semplice disturbo. Per affrontare questa problematica, è stata condotta una simulazione in cui diverse tipologie di rocciatrici sono state sottoposte a varie condizioni operative, raccogliendo dati fondamentali per l'addestramento di modelli di classificazione.

1.1 Analisi del dataset

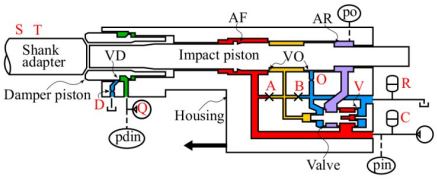
Il dataset impiegato per lo svolgimento del progetto proviene dalla seguente Challenge: "*https://data.phmsociety.org/2022-phm-conference-data-challenge/*". I dati in questione sono stati raccolti nel seguente modo. In primo luogo sono stati montati i seguenti sensori di pressione in diversi punti della perforatrice. Durante le

Sensore	Sample	Descrizione
PIN	50 kHz	Pressione di percussione sul raccordo di ingresso
$PDMP$	50 kHz	Smorzatore di pressione all'interno della camera esterna
PO	50 kHz	Pressione nel volume dietro il pistone

Tabella 1.1: Tabella dei sensori utilizzati

misurazioni, sono stati introdotti in modo controllato diversi tipi di guasti e disturbi, modificando la struttura della perforatrice in varie configurazioni. In particolare, sono state simulate undici diverse condizioni, inclusa una condizione di riferimento "No-fault" (nessun guasto) per confrontare il normale funzionamento della macchina con le condizioni di guasto. La Figura 1.1 mostra la posizione dei sensori di pressione (PIN, PDMP e PO) sulla perforatrice e le diverse tipologie di guasto indotte, che interessano varie componenti della stessa.

La raccolta dei dati è stata suddivisa per individui, un termine utilizzato per indicare diverse configurazioni della perforatrice create variando i parametri di controllo del banco di prova. In totale sono stati simulati otto individui, con



Label	Letter	Description
1	NF	No-fault
2	T	Thicker drill steel.
3	A	A-seal missing. Leakage from high pressure channel to control channel.
4	B	B-seal missing. Leakage from control channel to return channel.
5	R	Return accumulator, damaged.
6	S	Longer drill steel.
7	D	Damper orifice is larger than usual.
8	Q	Low flow to the damper circuit.
9	V	Valve damage. A small wear-flat on one of the valve lands.
10	O	Orifice on control line outlet larger than usual.
11	C	Charge level in high pressure accumulator is low.

Figura 1.1: Tipologie di fault e posizione dei sensori

l'obiettivo di emulare la variabilità presente in una vasta gamma di perforatrici e impianti di perforazione. Cinque di questi individui sono stati utilizzati per il training set, due per il validation set (per la validazione e l'ottimizzazione del modello durante l'addestramento), e uno per il test set finale, per valutare le prestazioni del modello su dati completamente nuovi. La Figura 1.2 mostra la distribuzione del numero di cicli di impatto raccolti per ogni individuo e per ogni classe di guasto (inclusa la classe No-fault).

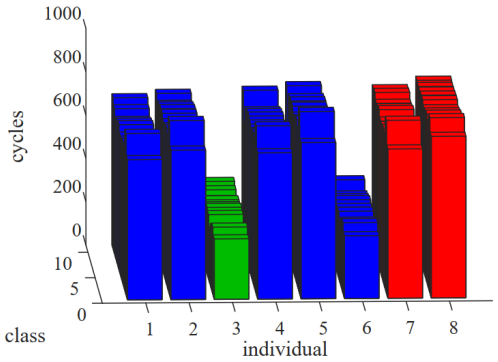


Figura 1.2: Suddivisione dei dati per individui e fault

1.2 Pre-processing e conversione del dataset in formato MATLAB

Come descritto in precedenza, i dataset sono suddivisi in set di riferimento, training set e test set, ciascuno contenente dati provenienti dai tre sensori (PDMP, PIN e PO). Ogni riga dei file di dati rappresenta un ciclo di impatto della perforatrice ed è composta da una serie temporale di misure di pressione, che descrivono le vibrazioni a cui è soggetta la perforatrice durante il ciclo, e dal corrispondente fault code (etichetta), che indica la classe di guasto o la condizione "No-fault".

Un'analisi preliminare dei dati ha evidenziato che le serie temporali non presentano tutte la stessa lunghezza. Per uniformare i dati, è stato necessario troncare le serie più lunghe per allinearle alla lunghezza minima osservata. Inoltre, per facilitare

l'elaborazione e l'estrazione di caratteristiche rilevanti, i dataset relativi a ciascun sensore sono stati uniti e convertiti in un formato *Timetable* di MATLAB. Questo formato è compatibile con il Diagnostic Features Designer Tool di MATLAB, che verrà utilizzato per l'analisi successiva.

Per gestire queste operazioni di pre-elaborazione, è stato sviluppato uno script MATLAB composto da una funzione principale (*main*) e due funzioni di supporto: *readDataBySensorName*, che legge i file CSV e preprocessa i dati raw, e *createTimetable*, che costruisce un oggetto *Timetable* a partire dai dati "puliti". L'esecuzione della funzione *main* richiama le funzioni di supporto per leggere, pre-processare e organizzare i dati in un formato adatto all'analisi successiva.

```
1
2       addpath("./utils/");
3       table_pin= readDataBySensorName("pin");
4       table_pin_TT = createTimetable(table_pin);
5
6       table_po = readDataBySensorName("po");
7       table_po_TT = createTimetable(table_po);
8
9       table_pdmp = readDataBySensorName("pdmp");
10      table_pdmp_TT = createTimetable(table_pdmp);
11
```

Figura 1.3: *main* del programma

La funzione *readDataBySensorName*, il cui funzionamento è illustrato in dettaglio in Figura 1.4, svolge un ruolo cruciale nella costruzione della *Timetable*. Essa si occupa delle seguenti operazioni:

- La funzione esplora la directory data alla ricerca di tutti i file che contengono il nome del sensore specificato come parametro di input.
- Una volta individuati i file pertinenti, la funzione procede alla lettura dei dati contenuti al loro interno.
- Per garantire l'uniformità della lunghezza delle serie temporali e prevenire il fenomeno del zero padding durante la Fast Fourier Transform (FFT), vengono troncate tutte le colonne che superano i 512 elementi. Tale scelta è stata argomentata nel Capitolo 2, dopo aver trattato il ranking delle features calcolate.
- Le colonne contenenti i valori delle misurazioni vengono convertite in numeri reali, escludendo le eventuali etichette o intestazioni. La colonna delle etichette, se presente, viene convertita in numeri interi.
- Infine, la funzione combina la *dataTable* risultante dalla lettura dei file con la *fullDataTable*, ottenendo così la tabella pre-processata che sarà utilizzata per la costruzione della *Timetable* successiva.

```
function fullDataTable = readDataBySensorName(sensore)

    file_list = dir("./data/");
    matching_files = [];
    for i = 1:length(file_list)
        if contains(file_list(i).name, sensore) && ~file_list(i).isdir
            matching_files = [matching_files; {file_list(i).name}];
        end
    end

    matching_files = string(matching_files);
    fullDataTable = [];
    colonnaEtichetta = 1;
    max_columns = 513; % 1 fault code + 512 istanti temporali
    for i = 1:length(matching_files)
        disp(strcat("In elaborazione: ", matching_files(i)));
        dataTable = readtable(strcat("./data/",
            matching_files(i)), 'ReadVariableNames', false);
        dataTable = dataTable(: , 1:max_columns);

        for j = 1:width(dataTable)
            if j ~= colonnaEtichetta && iscell(dataTable.(j))
                dataTable.(j) = str2double(dataTable.(j));
            end
        end

        if iscell(dataTable.(colonnaEtichetta))
                dataTable.(colonnaEtichetta) =
                int32(str2double(dataTable.(colonnaEtichetta)));
        end

        if isempty(fullDataTable)
            fullDataTable = dataTable;
        else
            fullDataTable = [fullDataTable; dataTable];
        end
    end
end

```

Figura 1.4: Funzione *readDataBySensorName*

A questo punto, la tabella risultante verrà passata come argomento della funzione riportata in Figura 1.5, grazie alla quale le serie temporali vengono trasformate in formato *Timetable*, rendendole così leggibili dal Features Designer tool.

```
1
2       function newTable = createTimetable(fullTable)
3           newTable = table();
4
5           for i = 1:size(fullTable, 1)
6               app = table2array(fullTable(i, 2:end));
7               tt = array2timetable(app', TimeStep=seconds(0.00002));
8               newTable = [newTable; {fullTable.(1)(i), tt}];
9           end
10
11          newTable.Properties.VariableNames = {'Fault', 'Serie'};
12
13          clear tt app i;
14      end
15
16
```

Figura 1.5: Funzione *createTimetable*

Capitolo 2

Feature enginereeng

L'estrazione di feature diagnostiche è un processo che richiede competenze specifiche e un'attenta analisi dei dati. Tale attività viene resa più accessibile grazie al Diagnostic Feature Designer di MATLAB, un potente strumento incluso nel Predictive Maintenance Toolbox. Questo tool consente di esplorare e analizzare dati provenienti da sensori, come le misure di vibrazione precedentemente elaborate per estrarre feature diagnostiche significative. Queste feature diagnostiche sono indicatori chiave (proprietà caratteristiche), sia nel dominio del tempo che della frequenza, che racchiudono informazioni fondamentali per il rilevamento di anomalie e guasti. Il Diagnostic Feature Designer semplifica il processo di estrazione di feature, offrendo un'ampia gamma di tecniche statistiche, di elaborazione del segnale e basate su modelli.

2.1 Caricamento dati

Il caricamento dei dati nel Diagnostic Feature Designer (DFD) richiede attenzione poiché ogni tabella, contenente i dati provenienti da un singolo sensore, deve essere caricata individualmente. Questa procedura separata è necessaria perché le analisi diagnostiche da effettuare possono variare in base al tipo di sensore (Figura 2.2).

Name	Value	Size	Class
table_pdmp_TT	*34145x2 table*	34145x2	table
table_pin_TT	*34145x2 table*	34145x2	table
table_po_TT	*34145x2 table*	34145x2	table

Figura 2.1: Tabelle relative ai 3 sensori nel Workspace di Matlab

Durante il caricamento di ciascuna tabella nel DFD, è fondamentale impostare i fault-code come variabili condizionate. In questo modo, la rappresentazione delle serie temporali all'interno del DFD terrà conto di queste informazioni, consentendo di visualizzare e analizzare i dati in relazione alle diverse condizioni di guasto.

La procedura illustrata in Figura 2.2 per il sensore PDMP viene replicata in modo analogo per tutti gli altri sensori presenti nel dataset.

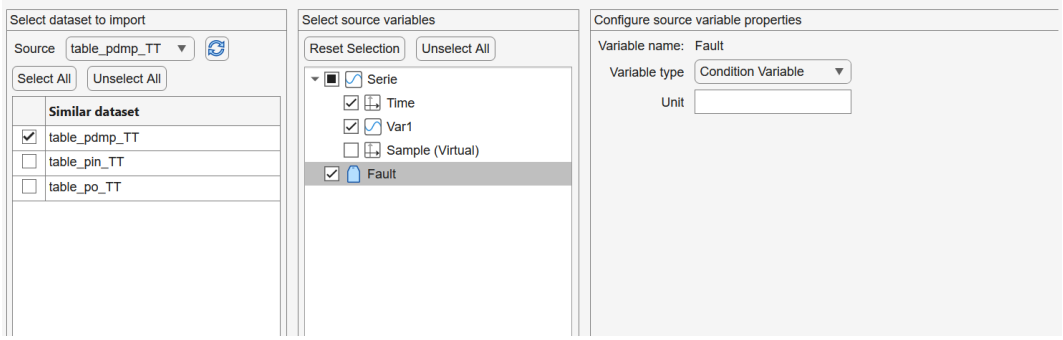


Figura 2.2: Caricamento variabile relativa al sensore PDMP nel DFD

È importante sottolineare che, dopo il caricamento dei dati, si è scelto di mantenere l'impostazione *Full Signal* nella sezione *Frame Policy* del DFD. Questa scelta preserva l'integrità dell'informazione e garantisce risultati più rappresentativi, utilizzando i segnali completi.

2.2 Analisi qualitativa dei segnali nel dominio del tempo

Nelle Figure 2.3, 2.4 e 2.5 sono riportati gli andamenti temporali delle serie relativi ai rispettivi sensori, differenziati per tipologia di guasto.

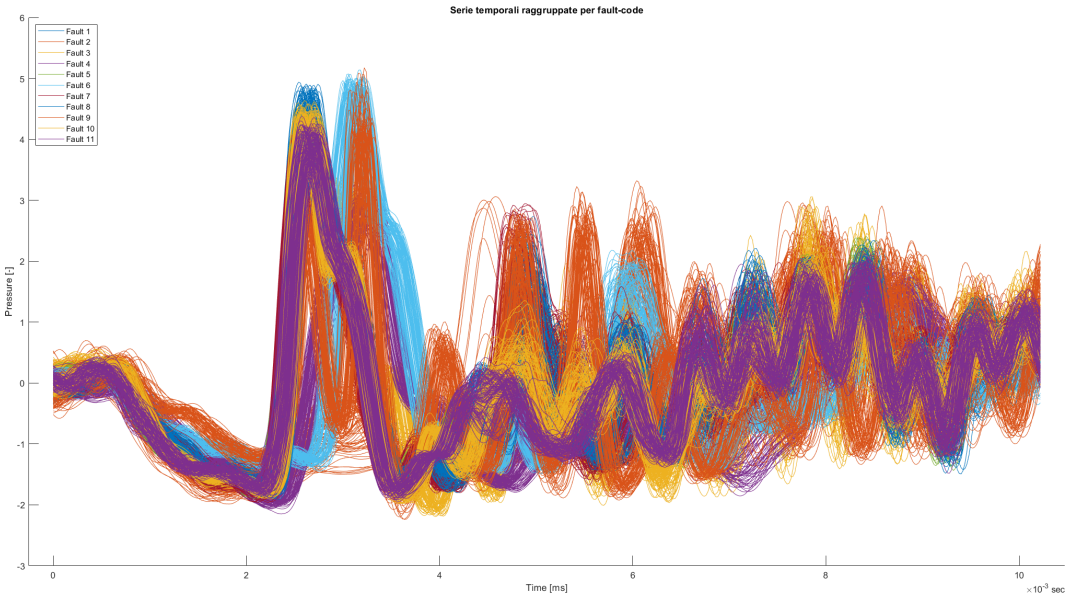


Figura 2.3: Grafico del segnale PDMP nel dominio del tempo

2.3 Analisi qualitativa dei segnali nel dominio della frequenza

Il DFD tool offre diverse opzioni per l'analisi in frequenza, sia parametriche (come il modello autoregressivo, AR) che non parametriche (come lo stimatore di Welch).

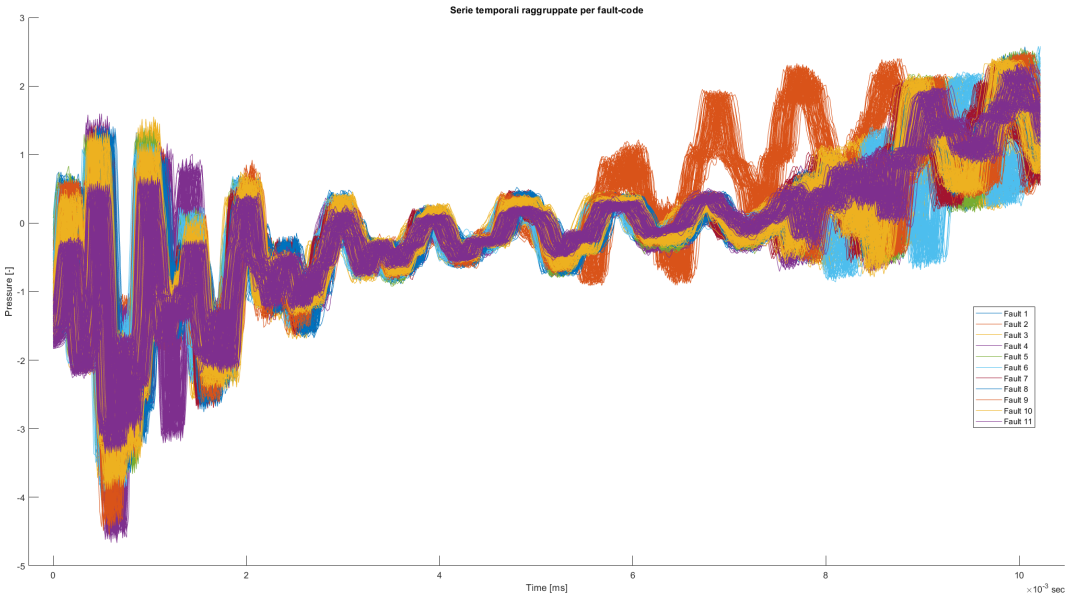


Figura 2.4: Grafico del segnale PIN nel dominio del tempo

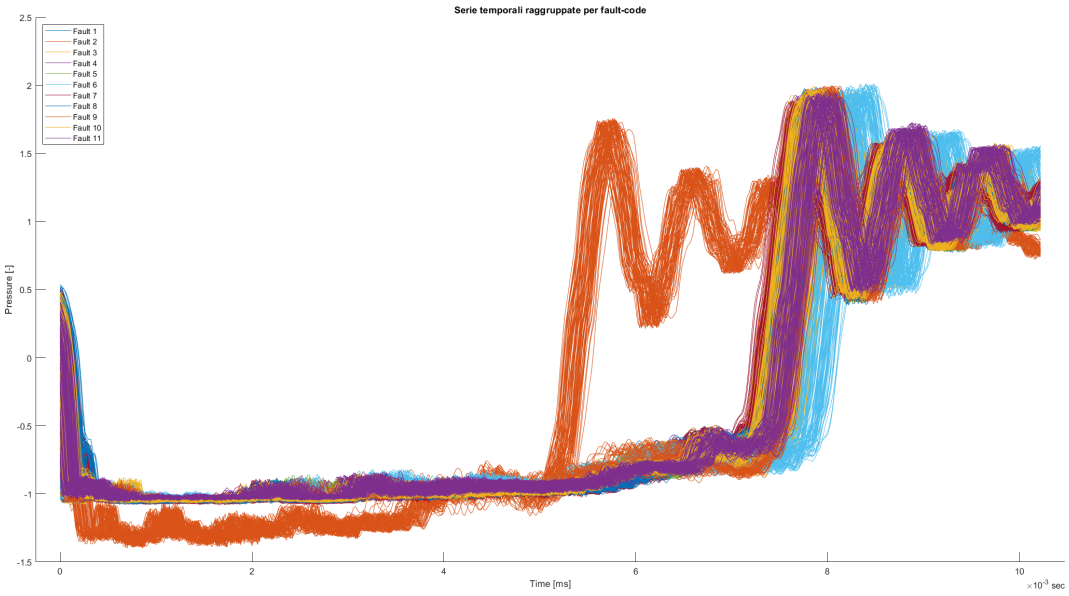


Figura 2.5: Grafico del segnale PO nel dominio del tempo

L'obiettivo di questa analisi è ottenere uno spettro del segnale per generare feature basate sulla frequenza.

Nella sezione "Spectral estimation" del DFD è possibile selezionare il metodo per ottenere la rappresentazione spettrale. Dopo vari tentativi, è emerso che il modello autoregressivo (AR) di ordine 17 permette di estrarre feature che, fornite in ingresso ai classificatori, ne migliorano significativamente le prestazioni.

In genere, il modello AR opera in due fasi. La prima è una fase di "fitting", in cui

i dati disponibili vengono utilizzati per adattare un modello autoregressivo. Nella seconda fase invece, sul modello derivato si calcola la potenza spettrale in forma chiusa. Di seguito sono riportati i risultati ottenuti con il modello AR per tutti e tre i sensori.

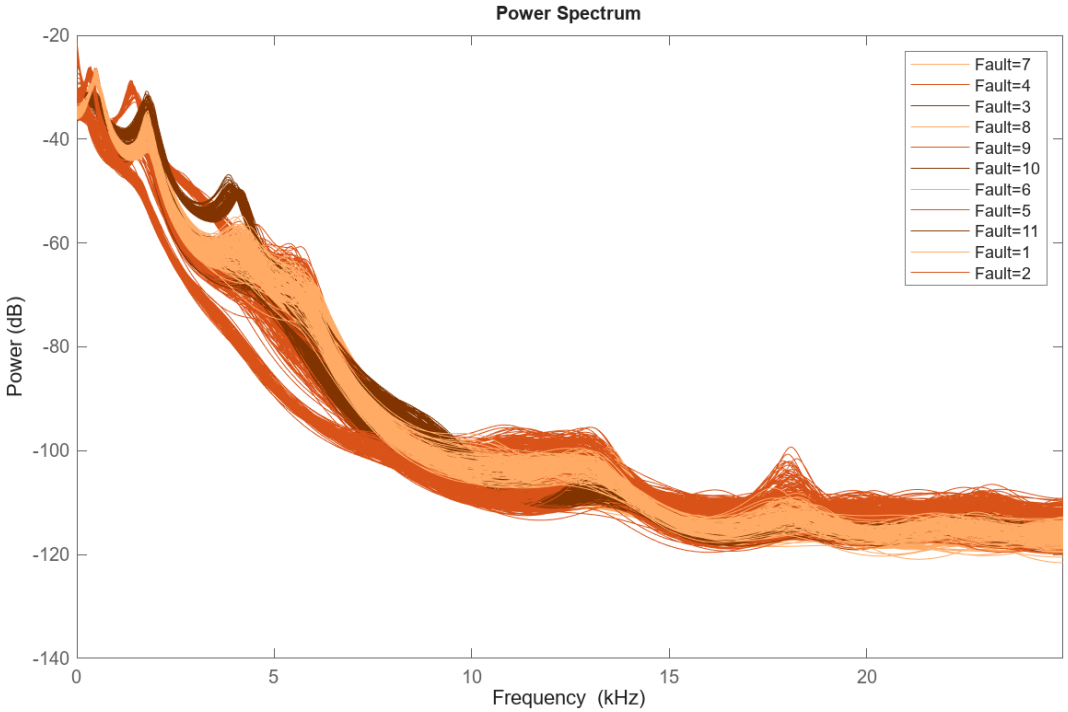


Figura 2.6: Grafico del segnale spettrale PDMP in scala lineare

Ottenuti i segnali temporali e spettrali per tutti e tre i sensori, è stata effettuata l'estrazione delle feature sia nel dominio del tempo che in quello delle frequenze, come descritto nelle sezioni successive.

2.4 Estrazione delle Feature Temporali

Per catturare le caratteristiche statistiche e dinamiche dei segnali di vibrazione, sono state estratte feature temporali utilizzando l'opzione "Time-Domain Features" del DFD. Poiché i dati disponibili consistono esclusivamente in segnali di vibrazione, è stata selezionata l'opzione "Signal Features".

Nel dominio del tempo sono state calcolate le seguenti feature statistiche:

- Media: rappresenta il valore medio del segnale.
- RMS (Root Mean Square): indica l'ampiezza quadratica media del segnale, fornendo una misura dell'energia complessiva.
- Deviazione standard: quantifica la dispersione dei valori del segnale attorno alla media.

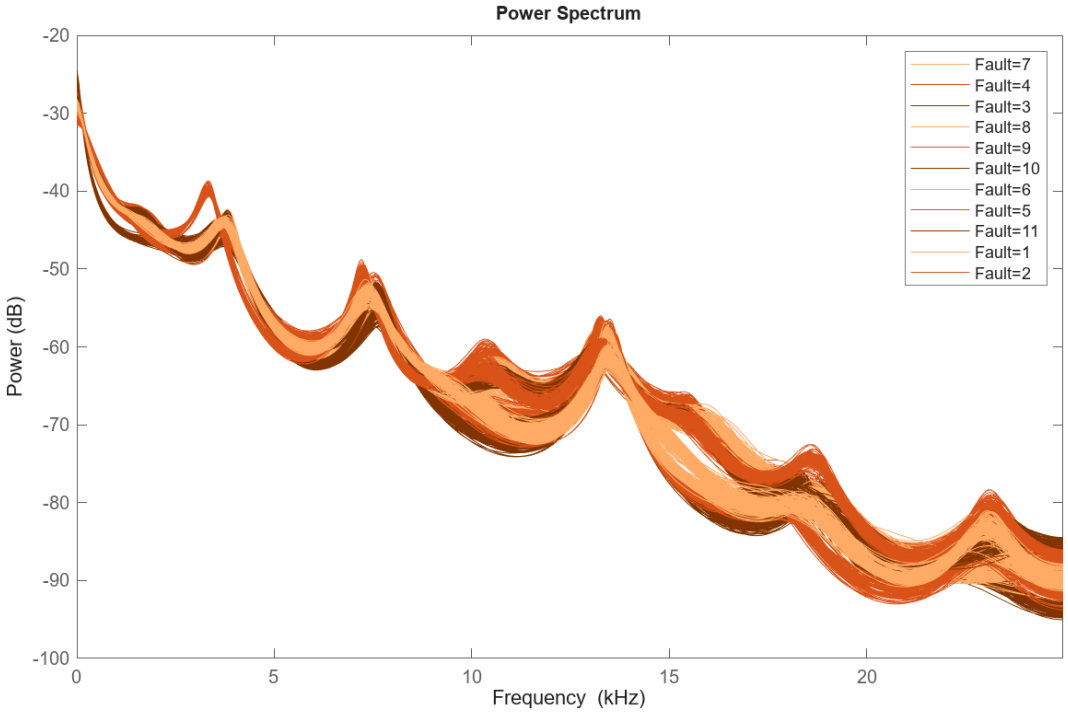


Figura 2.7: Grafico del segnale spettrale PIN in scala lineare

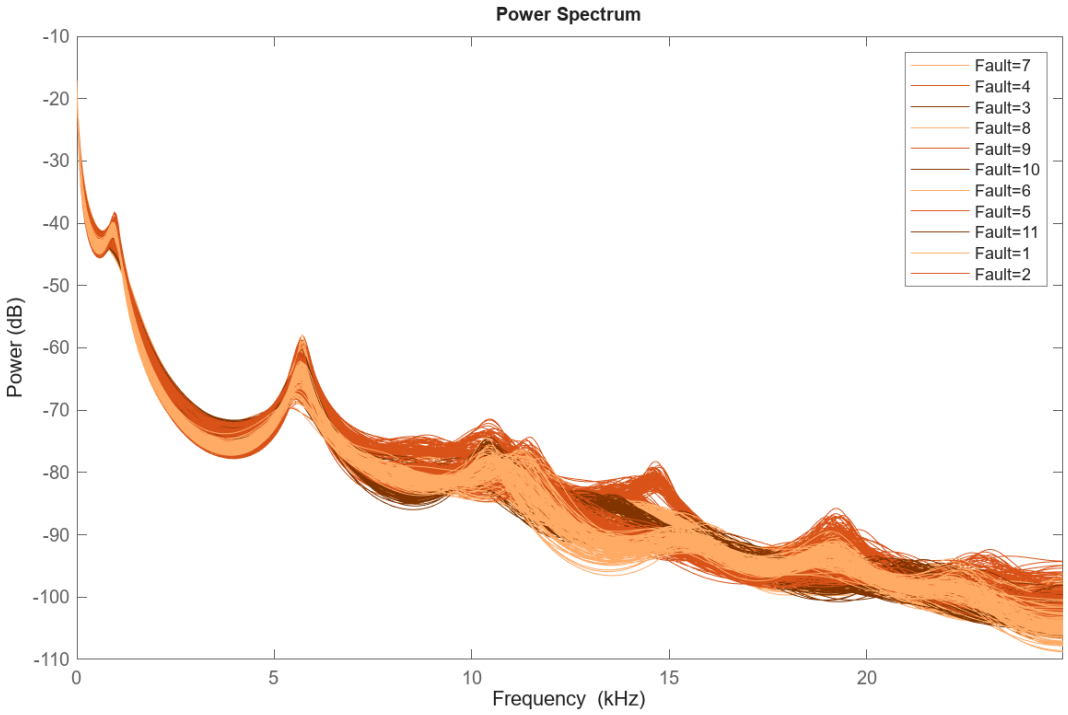


Figura 2.8: Grafico del segnale spettrale PO in scala lineare

- Fattore di forma: rappresenta il rapporto tra il valore RMS e il valore medio del segnale.

- Curtosi: misura l'appiattimento o la "spigolosità" della distribuzione dei valori del segnale rispetto a una distribuzione normale.
- Skewness (asimmetria): indica se la distribuzione dei valori del segnale è simmetrica o asimmetrica rispetto alla media.

Inoltre, sono state estratte le seguenti feature impulsive, utili per rilevare eventi transitori e impulsivi nei segnali:

- Fattore di cresta (Crest Factor): rapporto tra il valore di picco massimo e il valore RMS del segnale.
- Fattore impulsivo (Impulse Factor): rapporto tra il valore di picco massimo e il valore medio del segnale.
- Fattore di "clearance" (Clearance Factor): Rapporto tra il valore di picco massimo del segnale e un livello di riferimento predefinito.
- Valore di picco: rappresenta il valore massimo (in modulo) raggiunto dal segnale.

2.5 Estrazione delle Feature Spettrali

Al fine di analizzare il contenuto in frequenza dei segnali acquisiti dai sensori, sono state estratte feature spettrali utilizzando l'opzione "Spectral features" del DFD. Per ciascun sensore, è stato definito un intervallo di frequenza specifico, compreso tra *0.05005* kHz e *24.97* kHz, basato sull'analisi preliminare degli spettri di potenza dei segnali. Questa scelta ha permesso di focalizzare l'estrazione delle feature sulle bande di frequenza più rilevanti.

Le feature spettrali selezionate per l'analisi sono:

- Ampiezza di picco: rappresenta l'intensità massima raggiunta dal segnale in corrispondenza di specifiche frequenze.
- Frequenza di picco: indica le frequenze alle quali si verificano i picchi di ampiezza.
- Potenza di banda: quantifica l'energia del segnale distribuita in specifiche bande di frequenza.

Considerando la distribuzione dei picchi di frequenza osservata negli spettri dei segnali, è stato impostato un numero massimo di 5 picchi da estrarre per ciascun sensore.

Inoltre, sono stati definiti i seguenti parametri per l'estrazione delle feature:

- Soglia minima di picco: impostata a *-inf* per non escludere picchi di bassa ampiezza.

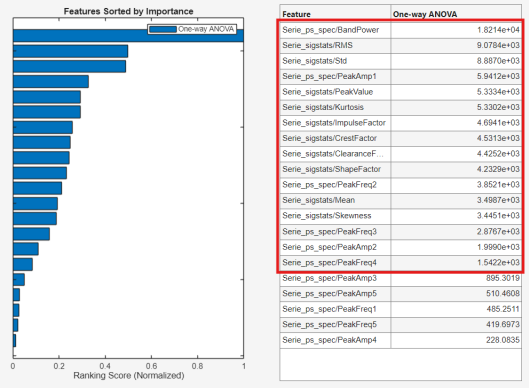
- Gap minimo di frequenza: fissato a *0.001* per evitare l'estrazione di picchi troppo ravvicinati in frequenza.
- Tolleranza all'escursione di picco: impostata a 0 per considerare solo i picchi che superano nettamente la soglia minima.

Questa configurazione ha permesso di ottenere un set di feature spettrali informative e discriminanti per l'analisi successiva.

2.6 Ranking delle Features

Dopo l'estrazione, le feature sono state ordinate per importanza utilizzando l'opzione "Rank features" del DFD. Questa funzione genera una tabella con un diagramma a barre orizzontali che mostra il contributo di ciascuna feature alla classificazione, assegnando un valore basato sul *rapporto F* calcolato tramite l'analisi della varianza (*ANOVA*). Un valore F elevato indica che la feature è più discriminante tra le diverse classi di guasto. Per ciascun sensore, è stata determinata una soglia di selezione basata su un evidente cambio di ordine di grandezza nel ranking delle feature. Sono state selezionate solo le feature con una rilevanza significativamente superiore a questa soglia. Questa metodologia ha permesso di escludere le feature con un contributo marginale alla classificazione, concentrando l'analisi sulle proprietà caratteristiche più discriminanti. In particolare:

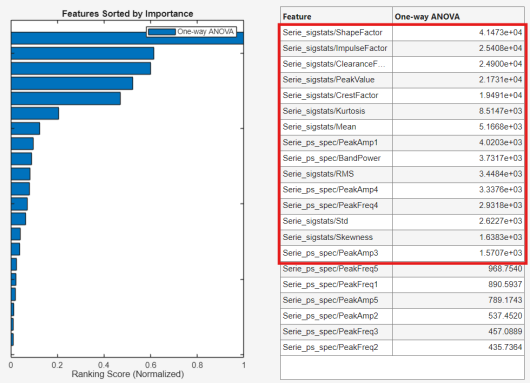
- Per il sensore *PDMP* sono state selezionate le prime 16 features.
- Per il sensore *PIN* sono state selezionate le prime 15 features.
- Per il sensore *PO* sono state selezionate le prime 14 features.



Feature	One-way ANOVA
Serie_ps_spec/BandPower	1.8214e+04
Serie_sigstats/RMS	9.0784e+03
Serie_sigstats/Std	8.8870e+03
Serie_ps_spec/PeakAmp1	5.9412e+03
Serie_sigstats/PeakValue	5.3334e+03
Serie_sigstats/Kurtosis	5.3302e+03
Serie_sigstats/ImpulseFactor	4.6941e+03
Serie_sigstats/CrestFactor	4.5313e+03
Serie_sigstats/ClearanceF...	4.4252e+03
Serie_sigstats/ShapeFactor	4.2329e+03
Serie_ps_spec/PeakFreq2	3.8521e+03
Serie_sigstats/Mean	3.4987e+03
Serie_sigstats/Skewness	3.4451e+03
Serie_ps_spec/PeakFreq3	2.8767e+03
Serie_ps_spec/PeakAmp2	1.9990e+03
Serie_ps_spec/PeakFreq4	1.5422e+03
Serie_ps_spec/PeakAmp3	895.3019
Serie_ps_spec/PeakAmp5	510.4608
Serie_ps_spec/PeakFreq1	485.2511
Serie_ps_spec/PeakFreq5	419.6973
Serie_ps_spec/PeakAmp4	228.0835

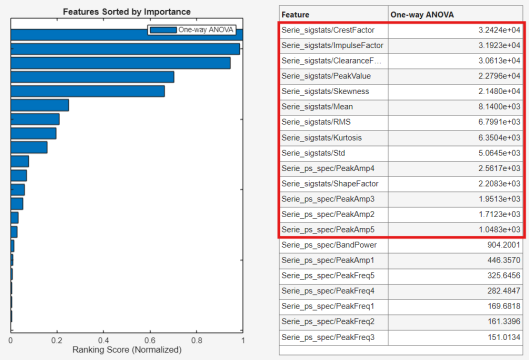
Figura 2.9: Rank delle features per il segnale PDMP

Prima di importare le feature nel Classification Learner tool di MATLAB, sono state esportate nell'ambiente di lavoro per ulteriori analisi ed elaborazioni. Inizialmente, le serie temporali estratte dai file CSV erano state troncate a 567 colonne (corrispondenti



Feature	One-way ANOVA
Serie_sigstats/ShapeFactor	4.1473e+04
Serie_sigstats/ImpulseFactor	2.5408e+04
Serie_sigstats/ClearanceF…	2.4900e+04
Serie_sigstats/PeakValue	2.1731e+04
Serie_sigstats/CrestFactor	1.9491e+04
Serie_sigstats/Kurtosis	8.5147e+03
Serie_sigstats/Mean	5.1668e+03
Serie_ps_spec/PeakAmp1	4.0203e+03
Serie_ps_spec/BandPower	3.7317e+03
Serie_sigstats/RMS	3.4484e+03
Serie_ps_spec/PeakAmp4	3.3376e+03
Serie_ps_spec/PeakFreq4	2.9318e+03
Serie_sigstats/Std	2.6227e+03
Serie_sigstats/Skewness	1.6383e+03
Serie_ps_spec/PeakAmp3	1.5707e+03
Serie_ps_spec/PeakFreq5	968.7540
Serie_ps_spec/PeakFreq1	890.5937
Serie_ps_spec/PeakAmp5	789.1743
Serie_ps_spec/PeakAmp2	537.4520
Serie_ps_spec/PeakFreq3	457.0889
Serie_ps_spec/PeakFreq2	435.7364

Figura 2.10: Rank delle features per il segnale PIN



Feature	One-way ANOVA
Serie_sigstats/CrestFactor	3.2424e+04
Serie_sigstats/ImpulseFactor	3.1923e+04
Serie_sigstats/ClearanceF…	3.0613e+04
Serie_sigstats/PeakValue	2.2796e+04
Serie_sigstats/Skewness	2.1480e+04
Serie_sigstats/Mean	8.1400e+03
Serie_sigstats/RMS	6.7991e+03
Serie_sigstats/Kurtosis	6.3504e+03
Serie_sigstats/Std	5.0645e+03
Serie_ps_spec/PeakAmp4	2.5617e+03
Serie_sigstats/ShapeFactor	2.2083e+03
Serie_ps_spec/PeakAmp3	1.9513e+03
Serie_ps_spec/PeakAmp2	1.7123e+03
Serie_ps_spec/PeakAmp5	1.0483e+03
Serie_ps_spec/BandPower	904.2001
Serie_ps_spec/PeakAmp1	446.3570
Serie_ps_spec/PeakFreq5	325.6456
Serie_ps_spec/PeakFreq4	282.4847
Serie_ps_spec/PeakFreq1	169.6818
Serie_ps_spec/PeakFreq2	161.3396
Serie_ps_spec/PeakFreq3	151.0134

Figura 2.11: Rank delle features per il segnale PO

al numero minimo di istanti temporali tra tutte le serie). Tuttavia, a causa del fenomeno di zero padding introdotto durante il calcolo della trasformata di Fourier, l'estrazione delle feature ha generato valori NaN nella tabella risultante.

Per risolvere questo problema, le feature estratte per ciascun sensore sono state combinate in un'unica tabella e sottoposte alle seguenti operazioni:

1. Per ogni feature, è stata calcolata la media all'interno di ciascuna classe di guasto.

2. I valori NaN sono stati sostituiti con la media della feature corrispondente alla rispettiva classe di guasto.

La funzione illustrata in Figura 2.12, riportata di seguito, è stata sviluppata per automatizzare questo processo.

Dopo aver addestrato i classificatori e analizzato i risultati, l'intero processo è stato ripetuto troncando le serie a 512 istanti temporali, come anticipato all'inizio di questo documento, ottenendo risultati analoghi. Questo conferma che l'approccio adottato è robusto e fornisce risultati soddisfacenti anche con serie temporali di lunghezza diversa. La scelta iniziale di troncare le serie a 567 istanti è stata motivata dalla volontà di preservare la maggior quantità possibile di contenuto informativo, considerando che la lunghezza media delle serie originali era di circa 650 istanti

```
1
2       function finalTable = replaceNaN(T)
3           classiUniche = unique(T.Fault);
4           for i = 1:length(classiUniche)
5               classe = classiUniche(i);
6               indiciClasse = T.Fault == classe;
7               nomiFeature = T.Properties.VariableNames;
8               nomiFeature(strcmp(nomiFeature, 'Fault')) = [];
9
10              for j = 1:length(nomiFeature)
11                  feature = nomiFeature{j};
12                  mediaFeature = mean(T.(feature)(indiciClasse),
13                                      'omitnan');
14                  if j == 25
15                      disp(mediaFeature);
16                  end
17                  T.(feature)(indiciClasse
18                      & isnan(T.(feature))) = mediaFeature;
19              end
20          end
21          finalTable = T;
22      end
23
```

Figura 2.12: Funzione *replaceNaN*

temporali. Tuttavia, la successiva analisi ha dimostrato che una lunghezza di 512 campioni è sufficiente per ottenere risultati altrettanto validi, riducendo al tempo stesso la complessità computazionale e semplificando l'analisi.

Capitolo 3

Addestramento e validazione

La fase successiva consiste nell'addestramento e validazione dei modelli di classificazione selezionati, utilizzando le feature estratte dai vari segnali. A tal fine, è stato utilizzato il Classification Learner di MATLAB. Il Classification Learner è uno strumento che permette di addestrare e valutare una vasta gamma di modelli di classificazione, come alberi decisionali, Support Vector Machine (SVM), reti neurali e molti altri, senza la necessità di scrivere codice. L'utilizzo di questo strumento è motivato dall'obiettivo di sviluppare un modello in grado di analizzare i segnali di vibrazione provenienti dai sensori della rocciatrice e diagnosticare eventuali guasti, come descritto nell'introduzione della relazione.

Nelle sezioni seguenti, verranno illustrate in dettaglio le procedure eseguite nel Classification Learner, a partire dal caricamento dei dati fino all'analisi dei risultati ottenuti dai diversi modelli di classificazione impiegati.

3.1 Caricamento dati

Nel Classification Learner è stata importata la tabella contenente le feature estratte da tutti e tre i sensori, unitamente alla colonna dei fault code, per definire il *Training set*. Questo insieme di dati è costituito da 34145 osservazioni (righe) e 46 variabili (colonne).

La Figura 3.1 mostra il pannello di configurazione iniziale del Classification Learner, con le informazioni relative ai dati importati.

3.2 Modelli di Classificazione

Per la fase di addestramento e validazione, sono stati considerati un'ampia gamma di modelli di classificazione, sfruttando quasi tutte le opzioni offerte dal Classification Learner tool. Nello specifico, i modelli utilizzati sono:

- Alberi decisionali (Tree): Fine Tree, Medium Tree, Coarse Tree.
- Analisi discriminante (Discriminant): Linear Discriminant, Quadratic Discriminant.

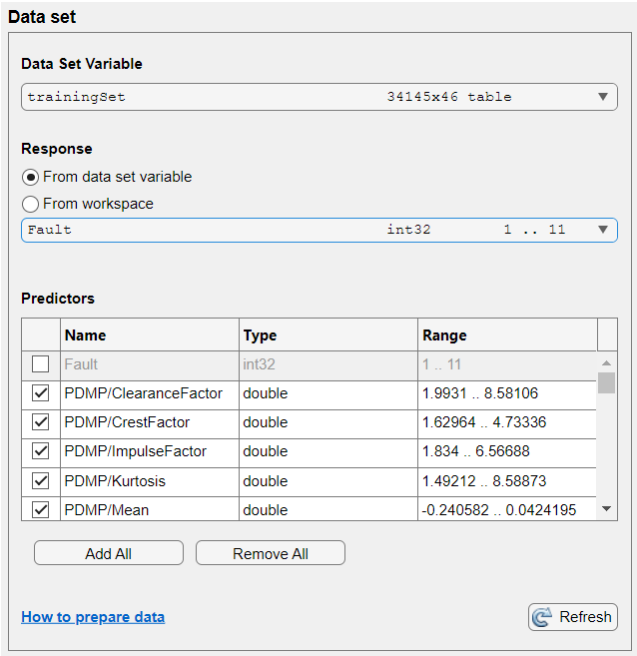


Figura 3.1: Pannello di caricamento dati del Classification Learner

- Naive Bayes: Gaussian Naive Bayes, Kernel Naive Bayes.
- Support Vector Machine (SVM): Linear SVM, Quadratic SVM, Cubic SVM, Fine Gaussian SVM, Medium Gaussian SVM, Coarse Gaussian SVM.
- K-Nearest Neighbors (KNN): Fine KNN, Medium KNN, Coarse KNN, Cosine KNN, Cubic KNN, Weighted KNN.
- Ensemble: Boosted Trees, Bagged Trees, Subspace Discriminant, Subspace KNN, RUSBoosted Trees.
- Reti Neurali (Neural Network): Narrow Neural Network, Medium Neural Network, Wide Neural Network, Bilayered Neural Network, Trilayered Neural Network.

Per ciascun modello di classificazione, sono state impiegate diverse configurazioni, variando sistematicamente i parametri specifici di ciascun algoritmo. Le prestazioni di ciascun modello sono state poi valutate in termini di Accuratezza, True Positive Rate (TPR) e False Negative Rate (FNR), estratti dalle matrici di confusione generate durante la fase di validazione. L'obiettivo è stato quello di individuare la combinazione modello-parametri che ottimizzasse l'accuratezza, minimizzando al tempo stesso gli errori di classificazione.

3.3 Cross-Validation e Risultati

Per garantire la robustezza e la generalizzabilità dei modelli di classificazione, è stata adottata una strategia di cross-validation a 5 fold. Questa tecnica, configurata all'inizio di ogni sessione nel Classification Learner tool, consente di valutare le prestazioni dei modelli su diverse porzioni dei dati, mitigando il rischio di overfitting e fornendo una stima più accurata della loro capacità di generalizzare a dati non visti.

I risultati di addestramento e validazione per tutti i modelli sono riportati in Figura 3.2. Di particolare rilievo sono le matrici di confusione delle versioni più performanti, riportate in Figura 3.10, che offrono una visione dettagliata delle capacità predittive dei modelli.

Model Type	Preset	Accuracy % (Validation)	Error Rate % (Validation)
Discriminant	*Linear Discriminant*	95,7	4,3
Discriminant	***Quadratic Discriminant***	**98,79**	**1,21**
Ensemble	*Boosted Trees*	86,98	13,02
Ensemble	***Bagged Trees***	**97,98**	**2,02**
Ensemble	*Subspace Discriminant*	93,27	6,73
Ensemble	*Subspace KNN*	96,88	3,12
Ensemble	*RUSBoosted Trees*	76,78	23,22
KNN	*Fine KNN*	97,53	2,47
KNN	*Medium KNN*	97,67	2,33
KNN	*Coarse KNN*	95,92	4,08
KNN	*Cosine KNN*	97,53	2,47
KNN	*Cubic KNN*	96,79	3,21
KNN	***Weighted KNN***	**97,77**	**2,23**
Naive Bayes	*Gaussian Naive Bayes*	83,28	16,72
Naive Bayes	***Kernel Naive Bayes***	**92,46**	**7,54**
Neural Network	*Narrow Neural Network*	98,88	1,12
Neural Network	*Medium Neural Network*	99,3	0,7
Neural Network	***Wide Neural Network***	**99,44**	**0,56**
Neural Network	*Bilayered Neural Network*	98,85	1,15
Neural Network	*Trilayered Neural Network*	98,84	1,16
SVM	*Linear SVM*	97,92	2,08
SVM	*Quadratic SVM*	99,46	0,54
SVM	***Cubic SVM***	**99,48**	**0,52**
SVM	*Fine Gaussian SVM*	88,67	11,33
SVM	*Medium Gaussian SVM*	99,21	0,79
SVM	*Coarse Gaussian SVM*	96,81	3,19
Tree	***Fine Tree***	**91,46**	**8,54**
Tree	*Medium Tree*	74,58	25,42
Tree	*Coarse Tree*	37,74	62,26

Figura 3.2: Risultati della cross-validation, a 5 fold, per tutti i modelli

L'analisi di tali matrici ha evidenziato l'ottima capacità dei modelli di classificazione nel distinguere tra le diverse condizioni di guasto e l'assenza di guasto (fault-free). L'elevata accuratezza, superiore al 90% per tutti i modelli in fase di validazione, suggerisce l'efficacia delle fasi di pre-processing e feature engineering nell'estrarre informazioni discriminanti dai dati. In ambito diagnostico, la corretta identificazione dei campioni privi di guasto è cruciale per minimizzare i falsi allarmi, un aspetto

critico per evitare interventi di manutenzione non necessari. L'analisi delle matrici di confusione conferma che tutti i modelli hanno ottenuto buoni risultati in questo senso, con la rete neurale (Figura 3.7) che si distingue per la sua precisione particolarmente elevata in questa classe, commettendo il minor numero di errori di classificazione rispetto agli altri modelli.

L'analisi delle matrici di confusione ha evidenziato una difficoltà comune a tutti i modelli nel predire correttamente la classe 5, oltre a una frequente confusione tra le classi 7 e 8. Questa osservazione è coerente con quanto riportato nelle precedenti illustrazioni, dove si è evidenziata la notevole somiglianza tra i profili temporali e in frequenza dei guasti rappresentati da queste due classi. La difficoltà nel distinguere tra le classi 7 e 8 potrebbe essere attribuita a una sovrapposizione delle loro caratteristiche intrinseche, rendendo più complicato il rilevamento di pattern discriminanti tra i due guasti. Per quanto riguarda la classe 5, la sua scarsa accuratezza predittiva potrebbe essere dovuta non solo alla sua intrinseca complessità, ma anche a una sua sottorappresentazione nel dataset di addestramento.

Questo sbilanciamento delle classi, in cui alcune categorie di guasto sono meno rappresentate di altre, può influenzare negativamente le prestazioni dei modelli di classificazione. Per affrontare questa problematica, si è esplorato nel prossimo capitolo l'applicazione di tecniche di campionamento stratificato, mirate a garantire una distribuzione più equilibrata dei campioni tra le diverse classi durante questa fase di cross-validation.

3.3.1 Stratified Sampling

In pratica, ad ogni iterazione della Cross-Validation, uno dei 5 fold viene riservato per il test, mentre i restanti 4 vengono utilizzati per l'addestramento del modello. Tuttavia, nei problemi di classificazione multiclasse con fold generati casualmente, alcuni fold potrebbero presentare una distribuzione sbilanciata delle classi, influenzando negativamente i risultati della validazione.

Per mitigare questo problema, è stata implementata la tecnica dello Stratified Sampling nella generazione dei fold. La stratificazione assicura che ogni fold mantenga la stessa proporzione di campioni per classe presente nel dataset originale. Ciò previene l'addestramento dei modelli su fold con rappresentazioni distorte delle classi, migliorando l'affidabilità delle stime di performance ottenute durante la cross-validation. La stratificazione è particolarmente cruciale quando le classi sono sbilanciate, ossia quando alcune classi hanno molti meno esempi di altre. In tali casi, questa tecnica previene che i modelli imparino a prevedere sempre la classe maggioritaria, ignorando quelle minoritarie.

Poiché tale funzionalità non era direttamente disponibile nel Classification Learner

tool, è stata importata la funzione di addestramento nel workspace di MATLAB e aggiunto il codice in Figura 3.11.

Si è deciso di applicare la stratificazione alla versione più performante di ciascun modello per valutarne l'impatto sulla validazione, ottenendo i seguenti risultati in termini di accuratezza:

- Quadratic Discriminant: *∼98,75%*
- Ensemble Bagged Trees: *∼98,11%*
- Weighted KNN: *∼97,85%*
- Kernel Naive Bayes: *∼92,56%*
- Wide Neural Network: *∼99,44%*
- Cubic SVM: *∼99,50%*
- Fine Tree: *∼91,52%*

L'applicazione della stratificazione nella cross-validation, sebbene non abbia portato a miglioramenti significativi nell'accuratezza complessiva, specialmente nei modelli già molto performanti, ha contribuito a una maggiore stabilità delle prestazioni, in particolare nei modelli meno accurati della selezione scelta.

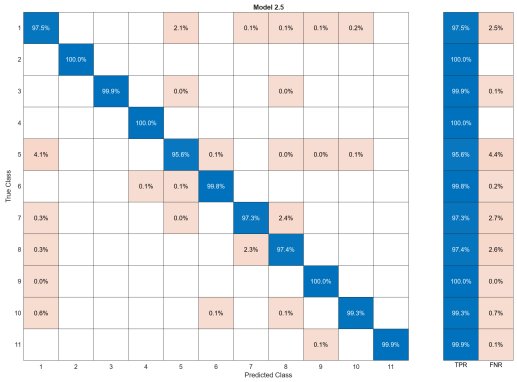


Figura 3.3: Quadratic Discriminant

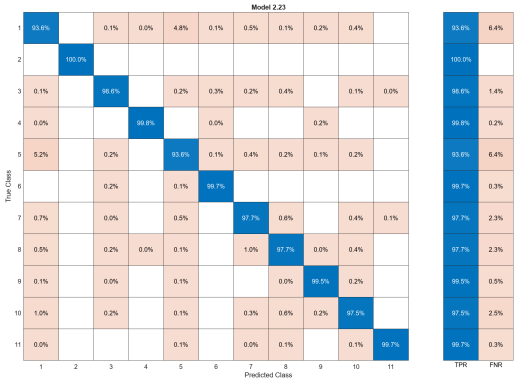


Figura 3.4: Ensemble Bagged Tree

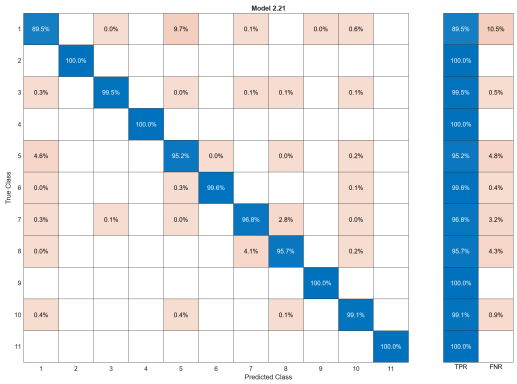


Figura 3.5: Weighted KNN

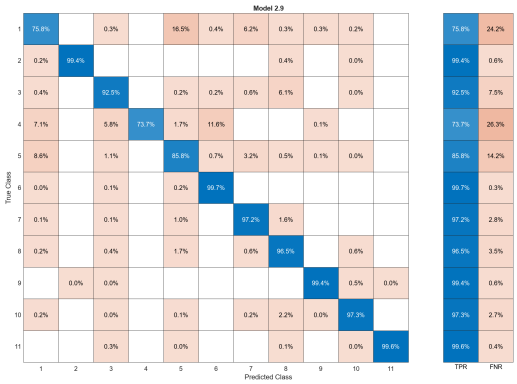


Figura 3.6: Kernel Naive Bayes

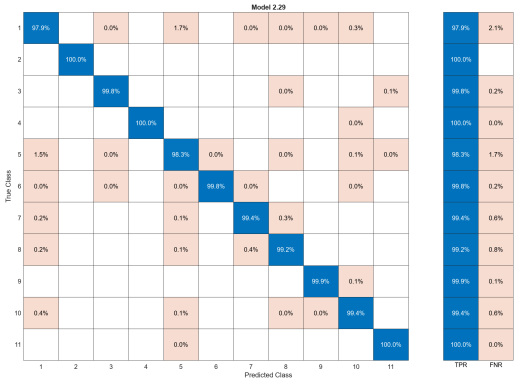


Figura 3.7: Wide Neural Network

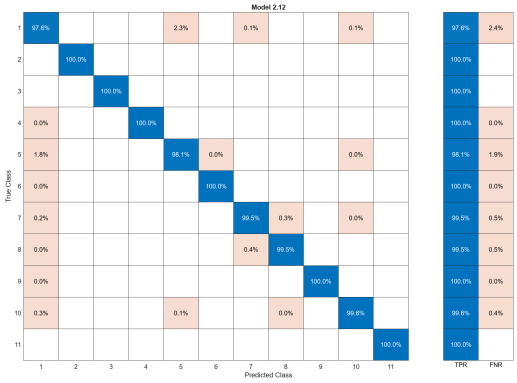


Figura 3.8: Cubic SVM

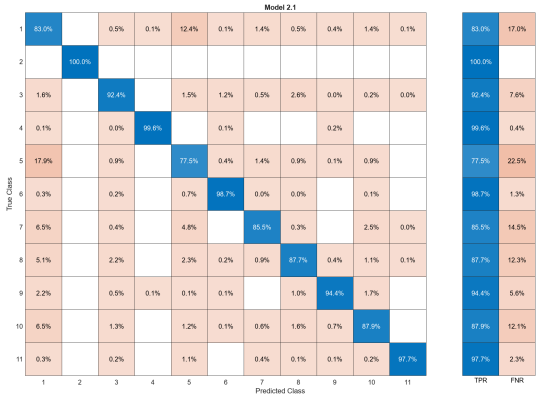


Figura 3.9: Fine Tree

Figura 3.10: Matrici di confusione

```
1
2     % Perform cross-validation
3      cvp = cvpartition(response, 'KFold', 5, 'Stratify', true);
4      partitionedModel = crossval(trainedClassifier.ClassificationSVM,
5                                  'CVPartition', cvp);
6
```

Figura 3.11: Esempio di implementazione della stratificazione nella cross-validation per il modello SVM

Capitolo 4

Test e analisi dei risultati

In questo capitolo, ciascun modello è stato addestrato su tutte le configurazioni utilizzate durante la Cross-Validation e successivamente testato su una configurazione inedita, ossia un'insieme di dati non utilizzati fino ad ora. Il dataset di test presenta una peculiarità: solo una piccola parte (circa 20 acquisizioni) è etichettata come classe 1, mentre la maggioranza (circa 3200 acquisizioni) è etichettata come classe 0, che rappresenta una classe generica tra le 11 disponibili. Pertanto, l'obiettivo del testing non è stato quello di valutare l'accuratezza complessiva dei modelli, bensì di analizzare come le predizioni si distribuissero tra le 11 classi.

4.1 Pre-processing e Feature enginereeng

Prima di poter utilizzare i dati per la classificazione, è stato necessario replicarne il pre-processing, seguendo le stesse procedure applicate al dataset di training. Una volta estratti dai file CSV, i dati sono stati importati in Diagnostic Feature Designer, con l'obiettivo di selezionare le medesime colonne scelte in precedenza per la definizione del dataset di training e per le quali era stato effettuato un ranking delle feature. Successivamente, i dati sono stati riportati nell'ambiente di lavoro di MATLAB, le tabelle relative a ciascun sensore sono state concatenate orizzontalmente e, infine, i dati così elaborati sono stati utilizzati per la fase finale di classificazione.

4.2 Fase di test

Come anticipato nell'introduzione del capitolo, la valutazione delle prestazioni dei modelli attraverso metriche convenzionali, come l'accuratezza, non è stata possibile a causa della mancanza di informazioni sui fault-code reali associati ai campioni di test. Tuttavia, per un sottoinsieme di 20 campioni, forniti già etichettati con la classe di guasto effettiva (classe 1, assenza di guasto), è stata condotta una valutazione parziale dei modelli. La Figura 4.1 presenta una tabella che riporta l'accuratezza ottenuta da ciascuna variante dei modelli su questi 20 campioni. È importante sottolineare che, sebbene questi risultati forniscano un'indicazione preliminare delle prestazioni, non sono sufficienti per trarre conclusioni significative sulla capacità di generalizzazione dei modelli a nuovi dati.

Model Type	Preset	Accuracy % (Validation)	Accuracy % (Test)
Discriminant	*Linear Discriminant*	95,7	100
Discriminant	*Quadratic Discriminant*	98,79	100
Ensemble	*Boosted Trees*	86,98	0
Ensemble	*Bagged Trees*	97,98	85
Ensemble	*Subspace Discriminant*	93,27	100
Ensemble	*Subspace KNN*	96,88	35
Ensemble	*RUSBoosted Trees*	76,78	0
KNN	*Fine KNN*	97,53	65
KNN	*Medium KNN*	97,67	50
KNN	*Coarse KNN*	95,92	25
KNN	*Cosine KNN*	97,53	40
KNN	*Cubic KNN*	96,79	35
KNN	*Weighted KNN*	97,77	50
Naive Bayes	*Gaussian Naive Bayes*	83,28	100
Naive Bayes	*Kernel Naive Bayes*	92,46	85
Neural Network	*Narrow Neural Network*	98,88	80
Neural Network	*Medium Neural Network*	99,3	50
Neural Network	*Wide Neural Network*	99,44	70
Neural Network	*Bilayered Neural Network*	98,85	100
Neural Network	*Trilayered Neural Network*	98,84	90
SVM	*Linear SVM*	97,92	95
SVM	*Quadratic SVM*	99,46	100
SVM	*Cubic SVM*	99,48	85
SVM	*Fine Gaussian SVM*	88,67	5
SVM	*Medium Gaussian SVM*	99,21	100
SVM	*Coarse Gaussian SVM*	96,81	100
Tree	*Fine Tree*	91,46	90
Tree	*Medium Tree*	74,58	0
Tree	*Coarse Tree*	37,74	0

Figura 4.1: Risultati della cross-validation, a 5 fold, e del test sui 20 campioni con classe 1

Nonostante questa limitazione, è stato comunque interessante analizzare la distribuzione delle classi predette, dei dati di test, dai modelli più performanti durante la precedente fase di validazione. Questa analisi qualitativa può offrire spunti utili per comprendere il comportamento dei modelli e identificare eventuali tendenze o criticità.

La distribuzione delle previsioni, illustrata in Figura 4.9, risulta essere pressoché uniforme per la maggior parte dei modelli, suggerendo una buona capacità di discriminare tra le diverse classi di guasto. Tuttavia, il modello Fine Tree si distingue per una tendenza a prediligere la classe 1 (assenza di guasto), indicando una potenziale difficoltà nel rilevare la classe 5. Questo comportamento potrebbe essere legato alla struttura del modello Fine Tree, che potrebbe essere meno adatto a catturare le relazioni complesse che caratterizzano i campioni di classe 5.

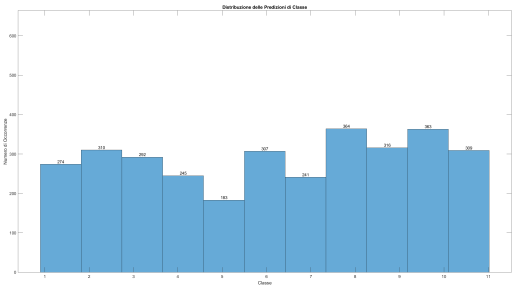


Figura 4.2: Quadratic Discriminant

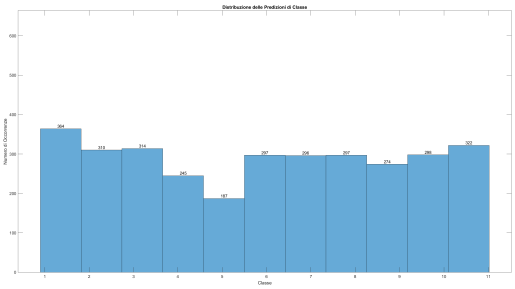


Figura 4.3: Ensemble Bagged Tree

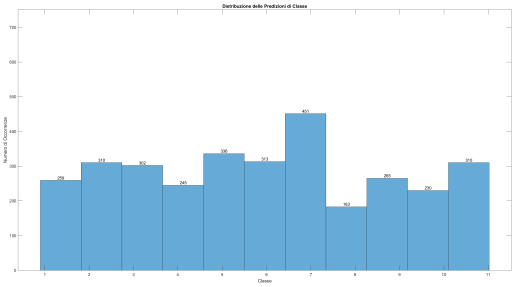


Figura 4.4: Weighted KNN

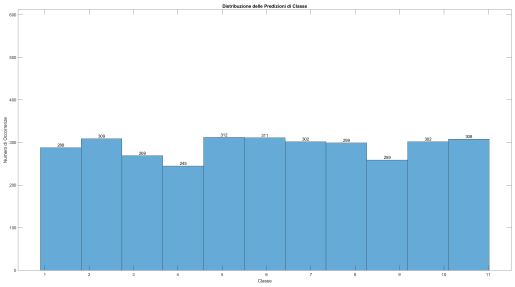


Figura 4.5: Kernel Naive Bayes

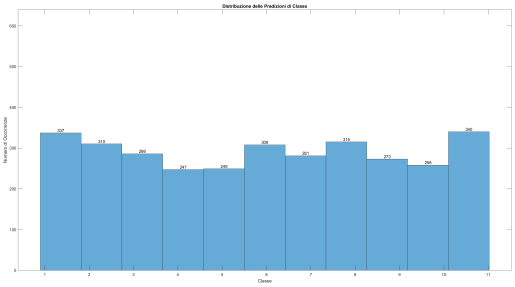


Figura 4.6: Wide Neural Network

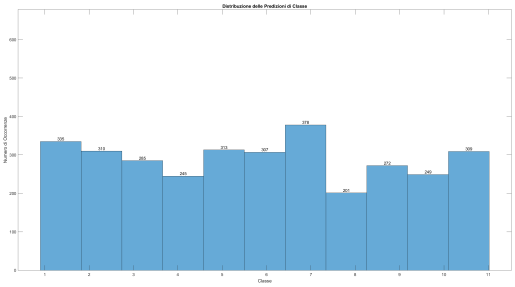


Figura 4.7: Cubic SVM

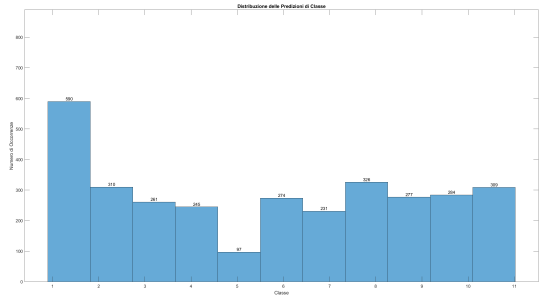


Figura 4.8: Fine Tree

Figura 4.9: Distribuzione delle predizioni di classe sui dati di test

Capitolo 5

Conclusione

Questo lavoro ha evidenziato il potenziale dell'utilizzo di strumenti come Diagnostic Feature Designer, Classification Learner e MATLAB per l'analisi e la classificazione di segnali complessi provenienti da sensori. L'applicazione di queste metodologie ha permesso di sviluppare un modello in grado di identificare e classificare diverse tipologie di guasto in una rocciatrice. L'utilizzo di questi tool ha consentito di affrontare le sfide legate all'analisi di dati complessi e alla creazione di modelli di classificazione. In particolare, l'estrazione di feature significative e la selezione dei modelli più performanti sono state facilitate dall'ambiente integrato offerto da MATLAB. I risultati ottenuti non solo confermano il potenziale di queste metodologie, ma offrono anche un esempio concreto di come esse possano essere utilizzate per migliorare l'efficienza e l'affidabilità dei macchinari industriali.