UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Modulo di diagnosi dei dati generati dal simulatore di un sistema di propulsione

PHM Asia Pacific 2023

Studenti:
Chiara Gobbi
Alice Moretti
Federica Parlapiano
Arianna Ronci

Professore:
Prof. Alessandro Freddi

Anno Accademico 2023-2024

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Modulo di diagnosi dei dati generati dal simulatore di un sistema di propulsione

PHM Asia Pacific 2023

Studenti:
Chiara Gobbi
Alice Moretti
Federica Parlapiano
Arianna Ronci

Professore:
Prof. Alessandro Freddi

Anno Accademico 2023-2024

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione
Via Brecce Bianche – 60131 Ancona (AN), Italy

Indice

Elenco delle figure

Introduzione

In collaborazione con l'Agenzia Spaziale Giapponese (JAXA) è stato sviluppato un simulatore per i sistemi di propulsione delle navicelle spaziali, al fine di migliorare la tecnologia di PHM (Prognostics and Health Management).
Si tratta di simulatore numerico che ha l'obiettivo di prevedere, con elevata precisione, la risposta dinamica di un sistema di propulsione spaziale. In tal modo è stato possibile generare un set di dati che copre sia le condizioni di normale funzionamento che gli scenari di guasto previsti. Questo è importante alla luce del fatto che i dati di telemetria che possono essere acquisiti in orbita sono ridotti a causa della limitazione nell'installazione di sensori e della capacità di downlink.

Lo scenario di un esperimento sul sistema di propulsione è mostrato in figura 1.

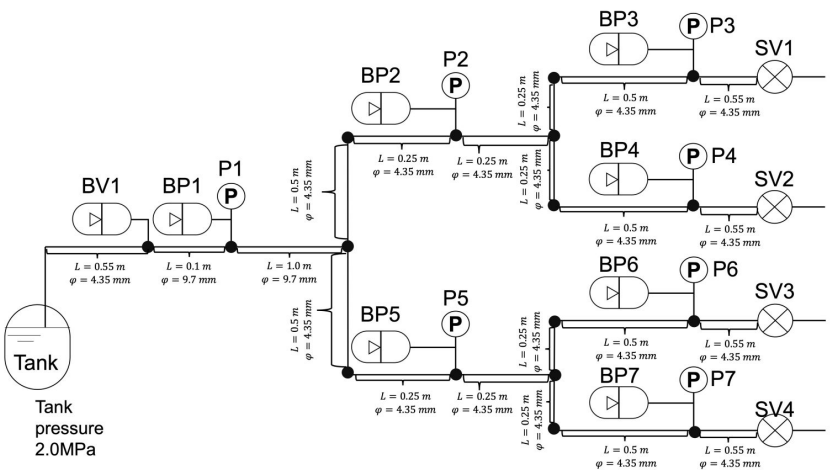


Figura 1: Schema di riferimento per gli esperimenti sul sistema di propulsione

Il seguente progetto ha l'obiettivo di sviluppare un un modulo di diagnosi in grado di classificare anomalie dovute bolle, guasti alle elettrovalvole e casi anomali sconosciuti partendo dai dati generati dal simulatore.
La competizione considerata una serie di esperimenti, alcuni dei quali presentano anomalie dovute alla contaminazione di bolle o un guasto dovuto all'apertura anomala dell'elettrovalvola. Inoltre, nei dati del test è inclusa un'anomalia sconosciuta.

Dataset

Il dataset contiene un totale di 223 esperimenti, ciascuno contenuto in file .cvs, divisi in training set (Case1-177) e test set (Case178-223). In particolare, per i dati del test set non è nota la classe a cui essi appartengono.
In ogni file .csv, la prima colonna rappresenta il tempo della misurazione, mentre le colonne successive rappresentano la pressione nei punti di misurazione P1 - P7, come mostrato nella figura 1.

Prediction goals

Il progetto prevede di svolgere alcuni task:

1. Task 1: determinare quali casi corrispondono al corretto funzionamento (*normal*) e quali ad un funzionamento anormale (*abnormal*).

2. Task 2: per i dati rilevati come *abnormal*, stabilire se si tratta di un'anomalia dovuta a:

   - *bubble contamination* → occasionalmente possono comparire delle bolle d'aria nei tubi durante il funzionamento di un veicolo spaziale; la presenza delle bolle modifica la velocità del suono, provocando lievi variazioni nelle fluttuazioni della pressione; è auspicabile rilevare l'eventuale presenza delle bolle e la loro posizione (task 3).
   - *solenoid valve fault* → un guasto dell'elettrovalvola è una delle principali modalità di guasto nei sistemi di propulsione dei veicoli spaziali; le elettrovalvole si aprono e si chiudono rispettivamente con un rapporto di apertura del 100% e dello 0% e in caso di guasto, le elettrovalvole si aprono con una percentuale compresa in tale range (non massima), il che comporta una riduzione del volume del fluido attraverso l'elettrovalvola; è necessario individuare quali elettrovalvole si sono guastate (task 4) e il loro rapporto di apertura (task 5).
   - *unknown fault* → nel funzionamento possono verificarsi anomalie o guasti del tutto imprevisti e sconosciuti; è necessario distinguere le anomalie sconosciute senza confonderle con anomalie e guasti noti.

3. Task 3: per i dati identificati come *bubble contamination*, determinare la posizione della bolla tra otto posizioni (BV1 e da BP1 a BP7).

4. Task 4: per i dati identificati come *solenoid valve fault*, stabilire quale delle quattro valvole (SV1 - SV4) è guasta.

5. Task 5: per la valvola del solenoide identificata come guasta, prevedere il rapporto di apertura ($0\% \leq OpeningRatio \leq 100\%$).

Strumenti per lo sviluppo

Per lo sviluppo del seguente progetto è stato utilizzato l'ambiente di programmazione MATLAB in particolare sono state utilizzate le app:

- Diagnostic Feature Designer, app che consente di realizzare la generazione e l'estrazione delle feature, parte del flusso di lavoro di manutenzione avanzata; in particolare permette di confrontare le feature generate in modo interattivo al fine di determinare quali sono le migliori e le più efficaci come indicatori di condizione per la diagnosi e la prognostica dei guasti.
- Classification Learner, app che facilita l'addestramento e la valutazione di modelli di classificazione.
- Regression Learner, app che facilita l'addestramento e la valutazione di modelli di regressione.
- Deep Learning Toolbox, utilizzato per valutare l'accuratezza di alcuni modelli.

Metodologia

La metodologia attuata per risolvere la challenge si sviluppa in una sequenza di passi ordinati.

Il punto di partenza è la generazione di feature diagnostiche utili al rilevamento e alla diagnosi di guasti. Tali feature sono necessarie per l'addestramento dei modelli di diagnosi per ciascuno dei task.
Per l'estrazione delle feature da un segnale è possibile specificare una frame policy. Quando si lavora con segnali temporali, come nella challenge affrontata, è utile suddividere il segnale in segmenti più piccoli, chiamati frame, per applicare determinate operazioni a ciascun segmento separatamente.
Se si applica una frame policy a un segnale in un problema di classificazione, il segnale viene diviso in frame più piccoli e ogni frame viene poi classificato individualmente. Tuttavia, per determinare la classe del segnale nella sua interezza, è necessario unire le predizioni di tutti i frame. In questo caso, per risolvere tale problema, si è deciso di implementare un algoritmo di voting che consenta di combinare le informazioni fornite da ciascun frame in una decisione complessiva sulla classe del segnale intero, tenendo conto delle predizioni individuali.
Nel seguente progetto l'algoritmo di votazione considera le etichette predette di ciascun frame e assegna loro un peso equo, per cui ogni frame contribuisce ugualmente alla decisione finale sulla classe del segnale. Calcolata la somma delle predizioni dei singoli frame, l'algoritmo assegna la classe più comune o la classe che ottiene la maggioranza dei voti al segnale intero, dove tale maggioranza è fissata ad una differente soglia in base alla fiducia che si vuole attribuire al risultato della classificazione del

segnale.

Altro passo di cui si compone la metodologia adottata consiste nel suddividere il dataset di training in maniera stratificata (ovvero mantenendo le probabilità a priori delle varie classi) in almeno due parti principali: un insieme di addestramento (training set) e un insieme di validation (validation set). Tale splitting è stato eseguito in maniera differente per i vari task tenendo in conto della numerosità delle classi. Questo è necessario poiché, per come la challenge è impostata, i dati di test non specificano la classe di guasto a cui essi appartengono. Di conseguenza questo è l'unico modo per poter validare il modello e per stimare i parametri degli algoritmi di classificazione.

In particolare:

- Durante lo sviluppo del modello, si è utilizzato il validation set per ottimizzare i parametri del modello. Questo è stato particolarmente utile per evitare l'overfitting e aumentare la generalizzazione con dati nuovi.
- In seconda fase, una volta scelti i parametri opportuni e i modelli di classificazione più accurati, si sono addestrati nuovamente i modelli utilizzando questa volta l'intero training set di dati.

Tale procedura aiuta a garantire che il modello non sia ottimizzato e adattato eccessivamente ai dati specifici dell'insieme di validazione, fornendo una stima più accurata delle prestazioni del modello su nuovi dati.

Infine, essendo che il risultato della challenge è stato pubblicato dagli organizzatori della competizione (file answers.csv), si sono valutati i modelli sull'insieme di test vero e proprio per valutare anche le prestazioni sul test set fornisce e una stima delle reali capacità predittive del modello.

L'implementazione è disponibile al seguente link.

Capitolo 1

Task 1

Il task 1 ha come obiettivo quello di determinare la natura dei dati, distinguendo tra quelli identificati come *normal* e quelli identificati come *abnormal*.
I dati presenti nel dataset sono suddivisi in tre categorie: *normal*, *anomaly* e *fault*. Tuttavia, poiché l'obiettivo della classificazione è binario, al fine di semplificare l'esecuzione del primo task, è stata eseguita una manipolazione delle etichette. In particolare, sono state assegnate le seguenti etichette:

- l'etichetta 0 è stata associata ai dati *normal*;
- l'etichetta 1 è stata assegnata ai dati *anomaly* e ai dati *fault*.

1.1 Generazione delle feature

Dopo aver manipolato i dati, si è proceduto con la generazione delle features nel dominio del tempo e nel dominio delle frequenze, sfruttando le funzionalità messe a disposizione dall'app Diagnostic Feature Designer.

1.1.1 Frame Policy

Per impostazione predefinita, l'app elabora l'intero segnale, ma è anche possibile suddividere il segnale in segmenti uniformi, chiamati *frames*, per condurre un'analisi sequenziale. In particolare, per l'analisi frammentata del segnale è necessario specificare:

- *frame size (FS)*, che specifica l'intervallo di tempo in secondi durante il quale i dati sono raccolti;
- *frame rate (FR)*, che specifica l'intervallo di tempo in secondi tra i tempi di inizio dei frame.

Viene chiamata "frame policy" la politica che determina tali parametri e quindi definisce come il segnale è suddiviso in frame.
Considerando che è bene che la frame rate sia un multiplo di 2 e avendo a disposizione un totale di 1200 campioni per ciascun esperimento, i valori ammissibili per la frame

rate rientrano nell'intervallo compreso tra 2 e 512.
Un altro aspetto da considerare nella scelta della frame policy è il numero di dati. Ad esempio, optando per una frame rate di 256 o 512 secondi, si escluderebbe un numero maggiore di dati in ciascun frame rispetto alla scelta di una frame rate di 64 o 128 secondi.
Per tali ragioni, l'analisi si è soffermata su una frammentazione del segnale scegliendo come frame policy FS=FR=0.064s e FS=FR=0.128s, che rispettivamente determinano una divisione in 19 e 10 finestre. In termini di performance ottenute dal classificatore successivamente addestrato e di bontà delle feature generate, la frame policy FS=FR=0.128s si è rivelata migliore per questo nella successiva trattazione si riportano i risultati ottenuti a partire dal segnale a cui è stata applicata tale policy.

1.1.2 Power Spectrum e Frequency-Domain Features

Tra le funzionalità dell'app Diagnostic Feature Designer è disponibile la generazione dello spettro di potenza di un segnale. Uno spettro di potenza caratterizza il contenuto in frequenza e le risonanze all'interno di un sistema. Poiché il degrado delle prestazioni di un sistema di solito causa cambiamenti nella firma spettrale, il comportamento spettrale costituisce una ricca fonte di informazioni per la generazione di caratteristiche.
Esistono differenti algoritmi parametrici e non parametrici per ottenere lo spettro di potenza di un segnale. L'opzione scelta per la generazione è il modello parametrico denominato AutoRegressive (AR) Model. Tale opzione in particolare presuppone la specifica dell'ordine del modello, parametro che indica il numero di campioni per effettuare la stima. In breve, il modello AR si basa sull'assunzione che il valore corrente di un segnale sia una combinazione lineare dei suoi valori passati, con l'aggiunta di un errore casuale. Per stimare lo spettro di potenza con il modello AR, si calcola innanzitutto il modello AR dai dati del segnale, poi successivamente, si calcola la funzione di trasferimento del modello AR e infine, si calcola lo spettro di potenza utilizzando la funzione di trasferimento.

Nel caso analizzato per ogni esperimento sono disponibili 7 segnali di pressione raccolti ciascuno da un diverso sensore nei punti di misurazione P1 - P7. Gli spettri ottenuti sono raffigurati di seguito.

In particolare per completezza, in questa sezione relativa al task 1 sono stati riportati tutti e 7 gli spettri di potenza ottenuti. Per brevità, nei task successivi invece gli spettri saranno omessi in quanto di fatto si tratta di spettri di potenza generati a partire dai medesimi segnali di pressione per un sottoinsieme di casi (esperimenti) d'interesse per lo specifico task.

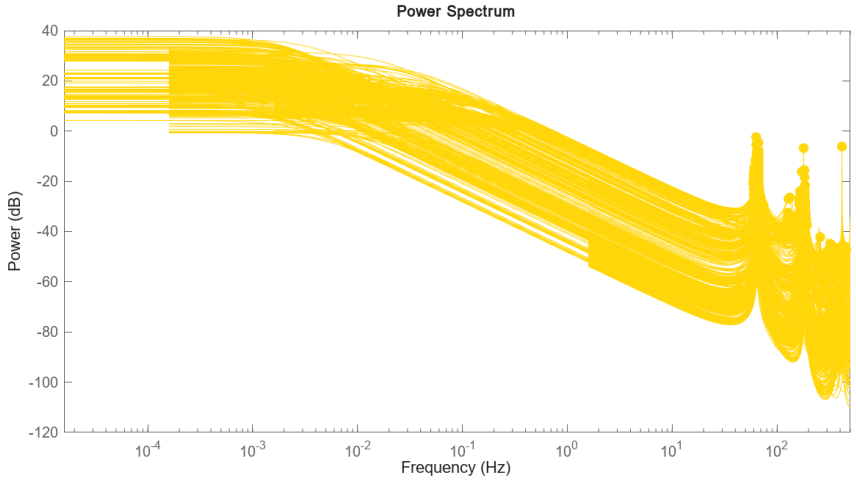


Figura 1.1: Power Spectrum per il segnale P1

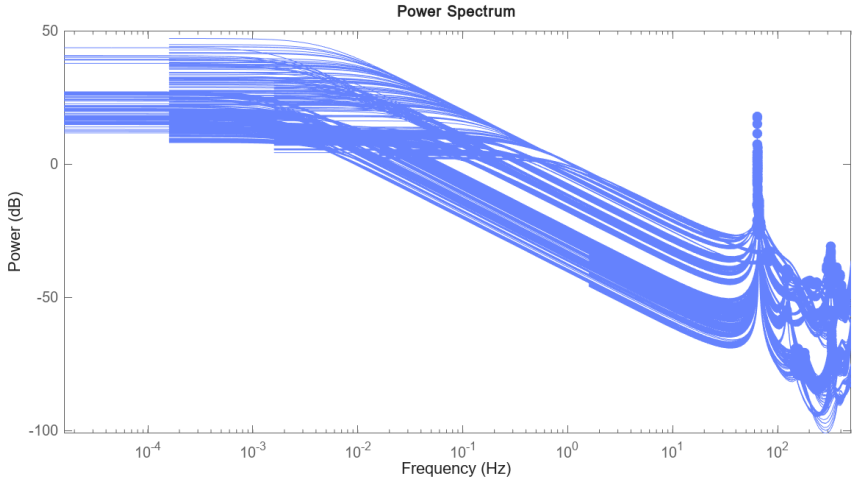


Figura 1.2: Power Spectrum per il segnale P2

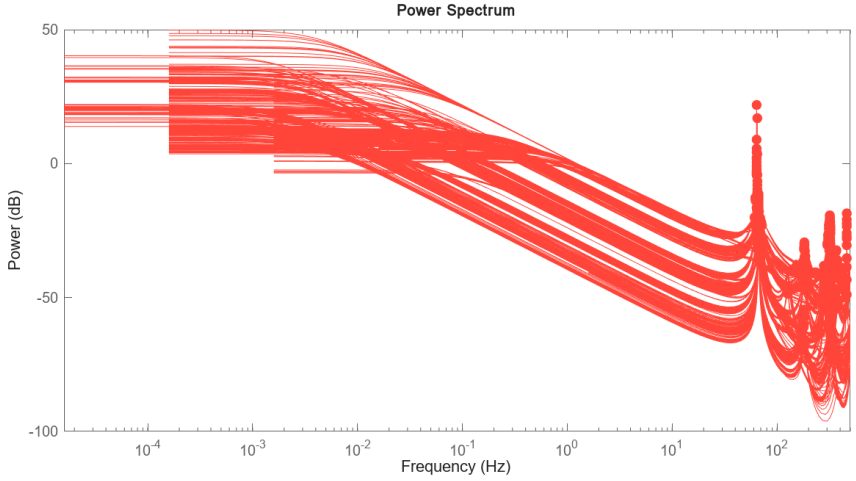


Figura 1.3: Power Spectrum per il segnale P3

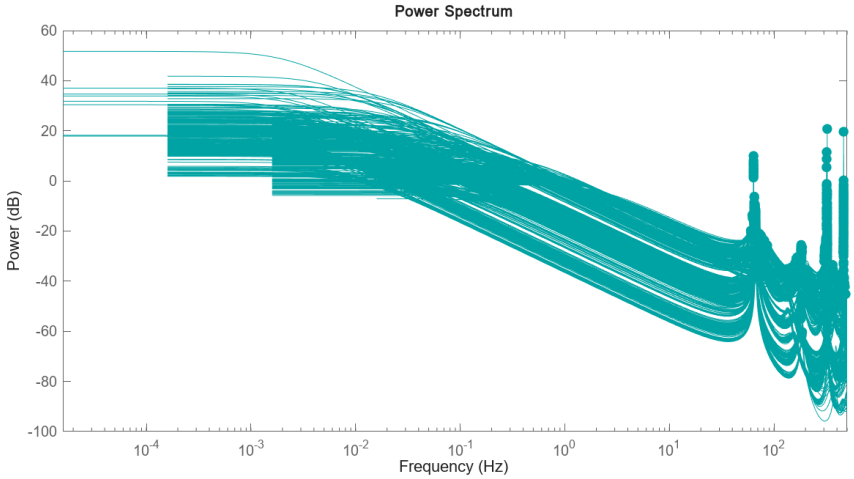


Figura 1.4: Power Spectrum per il segnale P4

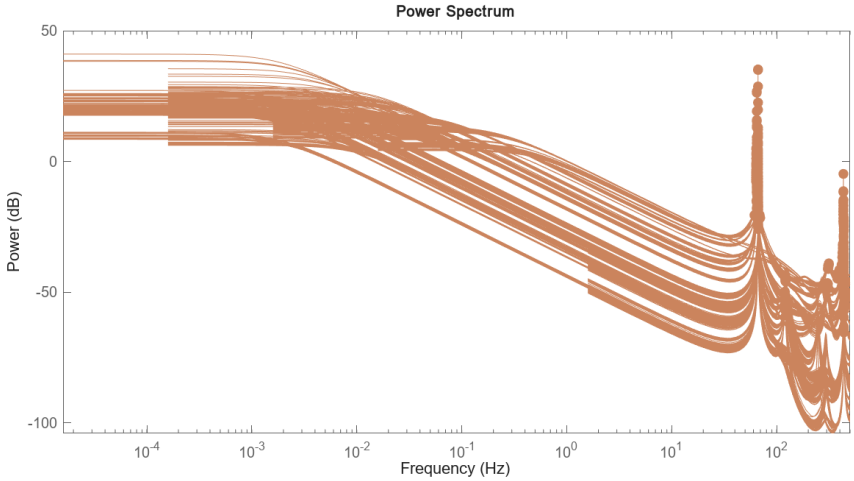


Figura 1.5: Power Spectrum per il segnale P5

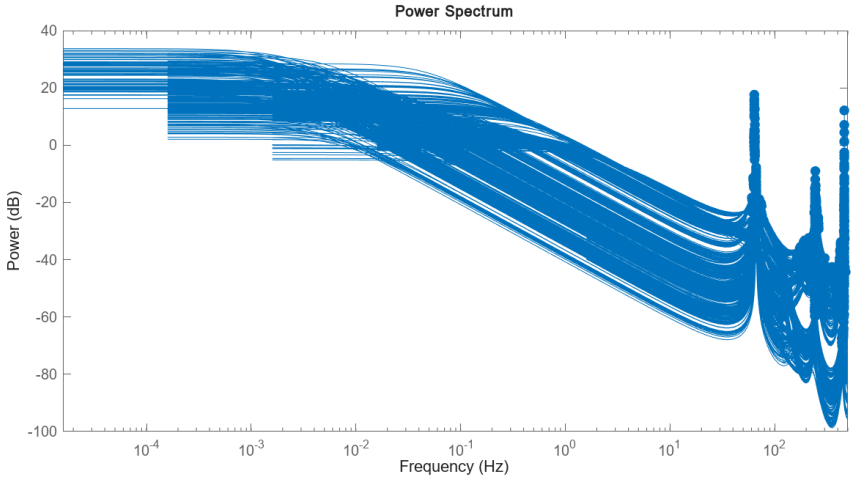


Figura 1.6: Power Spectrum per il segnale P6

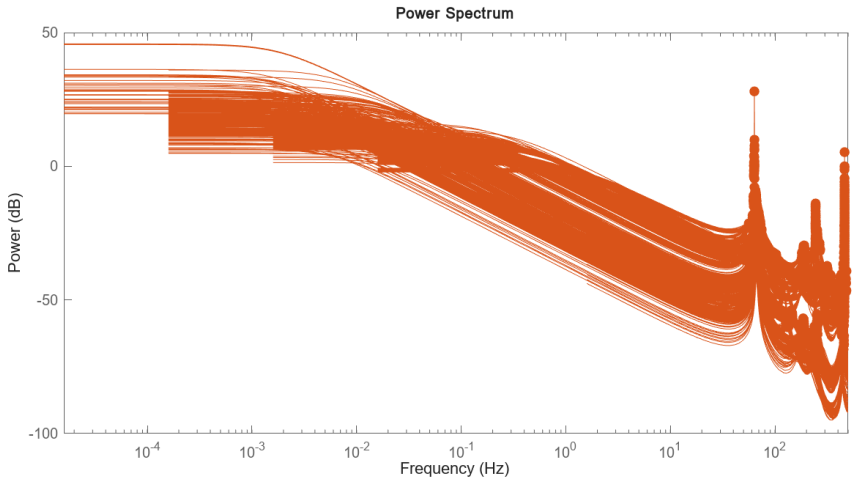


Figura 1.7: Power Spectrum per il segnale P7

Dall'analisi degli spettro di potenza dei segnali si nota praticamente in tutti i casi la distribuzione di energia che, nella gamma di frequenze tra 10 Hz e circa 300 Hz, è caratterizzata da un aumento significativo dell'energia, con picchi distintivi. Per evitare di inglobare una eccessiva componente di rumore non desiderato si sono tagliate fuori dalla banda in frequenza selezionata le frequenze più alte.

1.1.3 Time-Domain Features

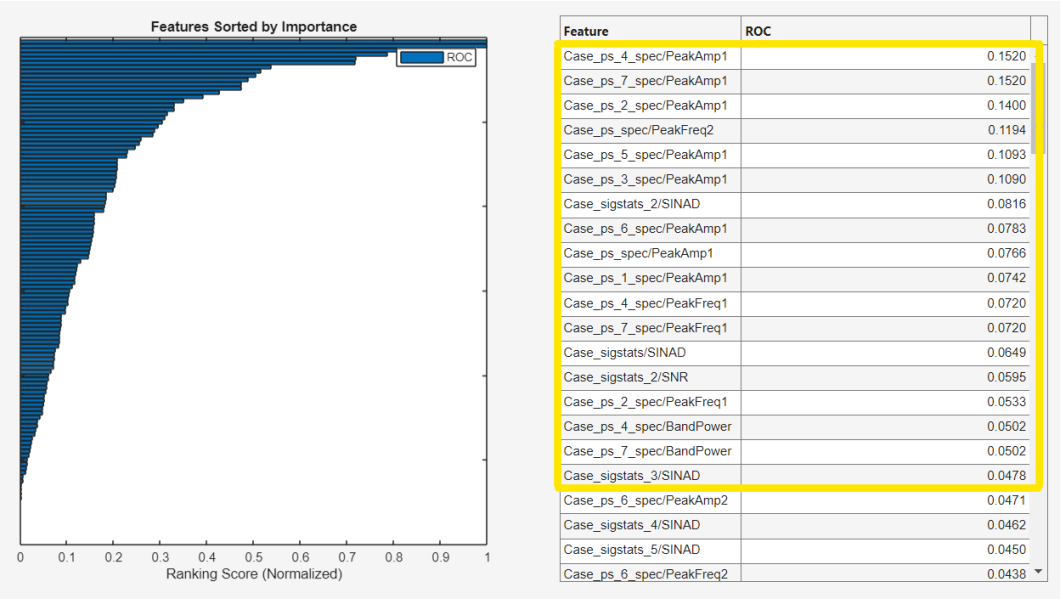
Diagnostic Feature Designer fornisce un insieme di opzioni per la generazione di caratteristiche diagnostiche nel dominio del tempo.
Per un segnale stocastico si pone l'interesse sul suo andamento medio su quelli che sono i parametri tipici delle variabili aleatorie, come valore medio, varianza, ... In generale, le feature di un segnale nel tempo forniscono indicatori statistici generali basate sul segnale che possono essere applicati a qualsiasi tipo di segnale.
Le variazioni in queste caratteristiche possono indicare cambiamenti nello stato di salute del sistema per questo sono utili al fine della diagnosi.

1.1.4 Ranking delle feature

La selezione delle feature è un'attività importante nel processo di classificazione dei dati, soprattutto quando si ha a che fare con un gran numero di variabili, in quanto può avere un impatto significativo sulle prestazioni complessive del modello. Un modo comune per selezionare le feature è attraverso l'uso di tecniche di ranking.
A tal proposito, generate le feature in frequenza e nel tempo, al fine di estrarre il sottoinsieme di feature maggiormente efficaci per l'addestramento del modello di classificazione, si è effettuato il ranking delle feature. Sono disponibili differenti metodi, ognuno dei quali utilizza un approccio statistico diverso.
Poiché il task di interesse è la classificazione, è opportuno usare come metrica una

tra quelle suggerite tra le voci supervised ranking, in quanto appropriate per valutare quanto efficacemente ciascuna caratteristica separa i dati con diverse etichette di condizione.
Il metodo predefinito per variabili di condizione (condition variable) a due valori (classificazione binaria) è il T-Test, metodo semplice che considera solo se le medie dei due gruppi etichettati sono uguali o meno. Il T-Test è principalmente utile per identificare caratteristiche inefficaci da scartare.
In realtà come metodo specifico per il caso d'esame si è impegato il ROC (Receiver Operating Characteristic), un metodo utilizzato principalmente per valutare le prestazioni di un classificatore binario.

Calcolato il valore secondo la metrica scelta del rank di ciascuna feature e ordinate per valore decrescente, l'obiettivo è selezionare il sottoinsieme ottimo di feature, tenendo conto del fatto che potrebbe esserci un punto in cui l'aggiunta di ulteriori feature non determina un miglioramento significativo delle prestazioni del modello addestrato su di esse.
Sulla sinistra è rappresentato un grafico a barre ordinato per valori decrescenti, in cui ogni barra corrisponde ad una feature e la dimensione di tale barra è pari al valore normalizzato tra 0 e 1 del ROC. Sulla destra sono riportate le medesime informazioni in forma tabellare; si sono evidenziate attraverso la bordatura in giallo le feature scelte ed estratte, in numero pari a 20, a seguito di differenti prove e valutazioni.
Il ranking risultante è quello riportato in figura 1.8.



Feature	ROC
Case_ps_4_spec/PeakAmp1	0.1520
Case_ps_7_spec/PeakAmp1	0.1520
Case_ps_2_spec/PeakAmp1	0.1400
Case_ps_spec/PeakFreq2	0.1194
Case_ps_5_spec/PeakAmp1	0.1093
Case_ps_3_spec/PeakAmp1	0.1090
Case_sigstats_2/SINAD	0.0816
Case_ps_6_spec/PeakAmp1	0.0783
Case_ps_spec/PeakAmp1	0.0766
Case_ps_1_spec/PeakAmp1	0.0742
Case_ps_4_spec/PeakFreq1	0.0720
Case_ps_7_spec/PeakFreq1	0.0720
Case_sigstats/SINAD	0.0649
Case_sigstats_2/SNR	0.0595
Case_ps_2_spec/PeakFreq1	0.0533
Case_ps_4_spec/BandPower	0.0502
Case_ps_7_spec/BandPower	0.0502
Case_sigstats_3/SINAD	0.0478
Case_ps_6_spec/PeakAmp2	0.0471
Case_sigstats_4/SINAD	0.0462
Case_sigstats_5/SINAD	0.0450
Case_ps_6_spec/PeakFreq2	0.0438

Figura 1.8: Rank Features task 1

Le feature scelte possono essere esportate sia sul workspace che sull'app Classification Learner. Inoltre, Diagnostic Feature Designer offre la possibilità di generare un codice che implementi tutto il processo di estrazione delle top n feature secondo il ranking specificato in modo automatico.
Di fatti per ogni task si è creata la relativa funzione; queste sono posizionate nella sottocartella del progetto $feature/$.

1.2 Addestramento del modello di classificazione

Per come la challenge è impostata, i dati di test non specificano la classe di guasto a cui essi appartengono. Di conseguenza in questa prima fase, per poter validare il modello si è deciso di dividere il training set specificato dalla challenge (Case 1-177) in due set distinti uno per l'addestramento del modello di classificazione uno per poter successivamente validare il modello. In altro modo non sarebbe stato possibile testare il modello su dati non visti in fase di addestramento.
A tal proposito per ogni task si è effettuato uno split come implementato nel file *train_test_split.m* della repository.

In particolare per il task 1, lo split si è effettuato in modo tale che per ogni classe (normal, valve fault, bubble anomaly) sia rispettato il rapporto 80/20 per la divisione. Ricavate le feature, ordinate secondo i criteri spiegati, queste sono state utilizzate per fare il training dei modelli di classificazione, valutando le prestazioni risultanti da differenti scelte in termini di numero di feature da considerare.
Inoltre, sono state effettuate anche differenti prove in termini di possibili algoritmi di classificazione, considerando tutti quelli proposti nel catalolgo dell'app Classification Learner.
In definitiva per il task 1, il modello di classificazione dalle migliori performance è stato addestrato su un numero di feature pari a 20 e una frame policy caratterizzata dai parametri $FS = FR = 0.128s$, da cui si determina una suddivisione del segNale in 10 frame. Il modello addestrato si basa sull'algoritmo *Optimizable Ensemble Tree*, algoritmo di classificazione che adotta la strategia di addestrare più alberi decisionali su sottoinsiemi diversi del set di dati di addestramento e quindi combinare le loro previsioni per ottenere una previsione più accurata e robusta.

Il classificatore assegna una etichetta ad ogni singolo frame. Per ricavare l'etichetta da associare all'esperimento nella sua interezza, si utilizza un algoritmo di voting, che in particolare in questo caso si basa su una maggioranza qualificata ($\frac{2}{3}n_{frame} + 1$) per la decisione della classe abnormal, alla luce del fatto che il costo di missclassificazione per le due classi è differente: è più costoso classificare erroneamente un caso normale come abnormal perché questo innesca tutto il loop di gestione di un guasto che non sarebbe invece necessario; al contrario invece è meno costoso non riconoscere un caso di funzionamento anormale poiché generalmente, anche se con un tempo di

detection più ampio, l'errore di classificazione commesso nell'istante attuale verrebbe riconosciuto all'aumentare degli effetti derivanti dal fault che agisce sul sistema, essendo che la supervisione è un processo ciclico. In sintesi almeno 8 frame su 10 devono essere predetti come dati di classe abnormal affinchè il caso sia etichettato nella sua interezza come tale.
In figura 1.9 è riportata la matrice di confusione ottenuta testando le performance del miglior classificatore, l'Optimizable Ensemble, ottenuto sui dati di validazione.

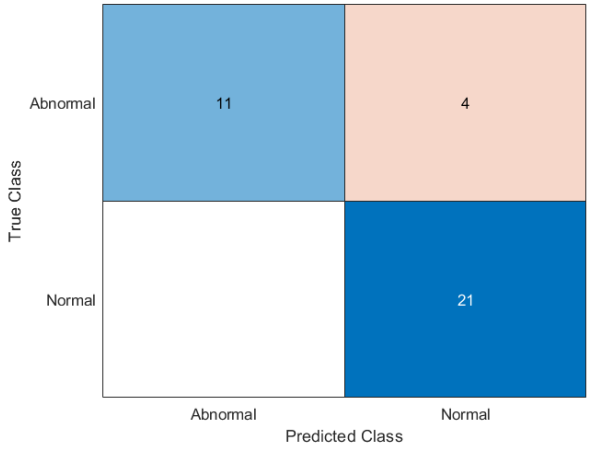


Figura 1.9: Matrice di confusione per il task 1 sul validation set

Il classificatore riconosce correttamente tutti i casi di corretto funzionamento, mentre non riconosce alcuni casi di comportamento anormale.
Per quanto l'ottimo è chiaramente l'assenza di casi missclassificati, è preferibile che il modello classifichi come caso di normale funzionamento un caso di guasto piuttosto che il viceversa.

Capitolo 2

Task 2

Il task 2 ha come obiettivo quello di discriminare la tipologia di guasto per tutti quei casi che nel task precedente sono stati etichettati come anormali. In particolare le classi di guasto sono le seguenti:

- *bubble contamination*, a cui è associata l'etichetta numerica 2;
- *solenoid valve fault*, a cui è associata l'etichetta numerica 3;
- *unknown fault*, a cui è associata l'etichetta numerica 1;

Tale task a differenza del precedente ha una difficoltà in più dovuta al fatto che la classe *unknown fault* non è rappresenta sui dati di training.
La classificazione senza dati di addestramento per una classe rappresenta una sfida significativa nell'ambito dell'apprendimento automatico, in quanto i modelli di classificazione solitamente imparano pattern a partire dai dati di addestramento per poi fare predizioni su nuovi dati.
Alla luce di ciò, l'approccio adottato per questo task è stato di impiegare una tipologia di modello idonea a gestire questa assenza di dati, ossia One-Class Support Vector Machine.
One-Class SVM è una tecnica di apprendimento automatico utilizzata per affrontare problemi di rilevamento di anomalie, che a differenza delle Support Vector Machine (SVM) tradizionali, sono addestrate solo con esempi di una singola classe. L'obiettivo principale è creare un modello che possa distinguere dati della classe nota dalla presenza di eventuali anomalie o outlier, senza la necessità di avere dati di tale classe.

2.1 Modellazione

Per poter gestire la classe di dati *unknown* non presente nel training set, si è deciso di scomporre un problema di classificazione ternaria in due problemi di classificazione binaria.
Lo schema che ne risulta è uno stacking di modelli di classificazione come mostrato in figura 2.1.

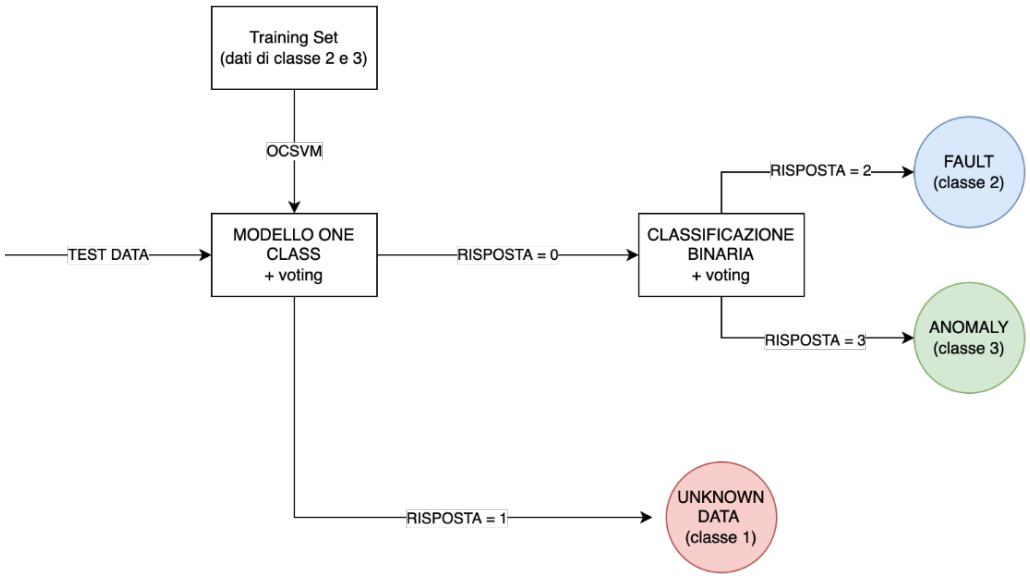


Figura 2.1: Stacking di Modelli

In particolare, l'idea è quella di considerare un training set costituito dai dati sui guasti sulle bolle e sulle valvole ed etichettarli con la label "0". Tale etichetta corrisponde alla classe di guasto nota (*known*). I dati sono poi utilizzati per fare il training di un classificatore "one-class".

Un classificatore "one-class" è un tipo di modello di machine learning progettato per identificare anomalie o outlier in un insieme di dati, senza la necessità di esempi negativi durante la fase di addestramento. Questo tipo di classificatore è particolarmente utile in questo caso in quanto si dispone solo di dati normali e si desidera identificare pattern anomali o comportamenti fuori dalla norma.

I dati riconosciuti come *unknown* in fase di tesing, sono etichettati con la label "1".
I dati etichettati come classe *known* (classe "0", cioè guasti noti) sono invece dati in pasto ad un ulteriore modello di classificazione in grado di discriminare quale sia il tipo di guasto.
Tale classificatore, in fase di testing, assegnerà la classe 2 nel caso di fault (guasto sulle valvole) e la classe 3 nel caso di anomaly (anomalia sulle bolle).

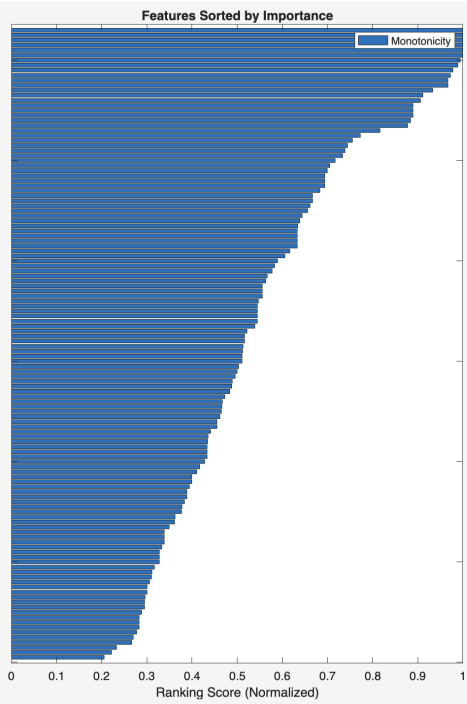
2.2 Generazione delle feature

La generazione di features è una pratica essenziale nell'analisi dei segnali per la diagnosi in quanto consente di estrarre informazioni da segnali complessi che possano variare nel tempo e nella frequenza.
In questo caso, è stato necessario estrarre due set di features diversi per la classificazione degli *unknown* e per la classificazione tra fault e anomalie. Per entrambe le

estrazioni, è stato settato l'ordine del modello a 10.

2.2.1 Unknown Data

Per quanto riguarda la classificazione dei dati *unknown*, sono state estratte le features diagnostiche relative al tempo e alla frequenza così come specificato in sezione 1.1. In seguito, è stato eseguito un ranking delle features secondo il criterio *monotonicity*. La monotonicità e analisi delle features indica la direzione costante o l'andamento uniforme di una relazione tra una feature e la variabile target. In altre parole, quando c'è monotonicità, all'aumentare (o al diminuire) dei valori di una feature, la variabile di risposta segue una direzione costante.

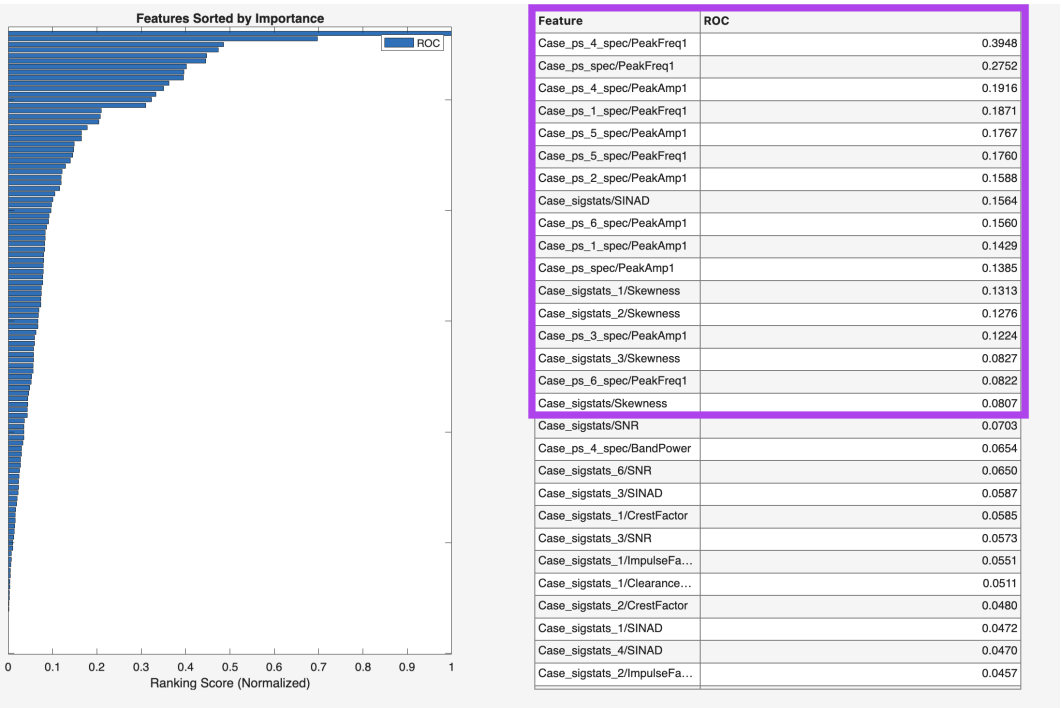


Feature	Monotonicity
Case_sigstats/Kurtosis	0.5556
Case_sigstats/PeakValue	0.5556
Case_sigstats_1/Kurtosis	0.5556
Case_sigstats_1/PeakValue	0.5556
Case_sigstats_4/Kurtosis	0.5556
Case_sigstats_4/PeakValue	0.5556
Case_sigstats_2/Kurtosis	0.5525
Case_sigstats_3/Kurtosis	0.5494
Case_sigstats_6/Kurtosis	0.5432
Case_sigstats_4/SINAD	0.5401
Case_sigstats_5/Kurtosis	0.5370
Case_sigstats_2/PeakValue	0.5370
Case_sigstats_5/PeakValue	0.5185
Case_sigstats_1/SINAD	0.5062
Case_sigstats_1/SNR	0.5031
Case_sigstats/SNR	0.4938
Case_sigstats_6/PeakValue	0.4938
Case_sigstats/SINAD	0.4938
Case_sigstats_4/SNR	0.4907
Case_sigstats_3/PeakValue	0.4877
Case_sigstats_3/CrestFactor	0.4537
Case_sigstats_6/CrestFactor	0.4290
Case_sigstats_5/CrestFactor	0.4198
Case_sigstats_1/CrestFactor	0.4136
Case_sigstats_3/ImpulseFa…	0.4105
Case_sigstats_4/CrestFactor	0.4074
Case_sigstats_2/CrestFactor	0.3981
Case_sigstats_4/ImpulseFa…	0.3920
Case_sigstats_4/Clearance…	0.3889

Figura 2.2: Rank Features task2, caso unknown

2.2.2 Bubble Anomaly e Solenoid Valve Fault

Per l'estrazione di features che consentano la discriminazione tra la classe fault (classe 2) e anomaly (classe 3), si è invece utilizzato un raking diverso. In breve, il criterio di ranking ROC, utilizzato per la classificazione binaria, si basa sull'analisi della curva ROC e dell'area sotto di essa per valutare quanto bene il modello riesce a distinguere tra le due classi.



Feature	ROC
Case_ps_4_spec/PeakFreq1	0.3948
Case_ps_spec/PeakFreq1	0.2752
Case_ps_4_spec/PeakAmp1	0.1916
Case_ps_1_spec/PeakFreq1	0.1871
Case_ps_5_spec/PeakAmp1	0.1767
Case_ps_5_spec/PeakFreq1	0.1760
Case_ps_2_spec/PeakAmp1	0.1588
Case_sigstats/SINAD	0.1564
Case_ps_6_spec/PeakAmp1	0.1560
Case_ps_1_spec/PeakAmp1	0.1429
Case_ps_spec/PeakAmp1	0.1385
Case_sigstats_1/Skewness	0.1313
Case_sigstats_2/Skewness	0.1276
Case_ps_3_spec/PeakAmp1	0.1224
Case_sigstats_3/Skewness	0.0827
Case_ps_6_spec/PeakFreq1	0.0822
Case_sigstats/Skewness	0.0807
Case_sigstats/SNR	0.0703
Case_ps_4_spec/BandPower	0.0654
Case_sigstats_6/SNR	0.0650
Case_sigstats_3/SINAD	0.0587
Case_sigstats_1/CrestFactor	0.0585
Case_sigstats_3/SNR	0.0573
Case_sigstats_1/ImpulseFa…	0.0551
Case_sigstats_1/Clearance…	0.0511
Case_sigstats_2/CrestFactor	0.0480
Case_sigstats_1/SINAD	0.0472
Case_sigstats_4/SINAD	0.0470
Case_sigstats_2/ImpulseFa…	0.0457

Figura 2.3: Rank Features task2, caso anomalie

2.3 Addestramento del modello di classificazione

Ricavate le features, ordinate secondo i criteri spiegati, queste sono state utilizzate per fare il training dei modelli di classificazione.
In prima battuta, sia per la classificazione dei dati *unknown* (primo modello) che per quella dei dati *known* (secondo modello), si sono scelti un numero diverso di features da considerare.
In particolare, l'obiettivo è stato quello di trovare un trade off in modo da avere un numero di features sufficienti:

- troppe poche features non permettono di definire una regione di separazione tra le classi opportuna,
- troppe features potrebbero portare a definire una regione di separazione troppo specifica che si adatta troppo ai dati di addestramento.

In particolare, a seguito di diverse prove, è stato deciso di utilizzare 22 features per il primo modello e 17 per il secondo.
Tale valutazione è stata eseguita suddividendo i dati di training di classe 2 e 3 in ulteriori due set:

- training set costituito dall'80% dei dati,
- validation set costituito dal restante 20%.

Tale suddivisione è stata eseguita mantenendo la probabilità a priori del training originario, ovvero, si è garantita, sia in training che in validation, la stessa percentuale di numerosità originaria delle due classi.

2.3.1 Classificazione dei dati unknown

Per la costruzione e l'ottimizzazione del modello che permetta di individuare i casi di guasto non noti, oltre ai dati scelti per il validation set secondo i criteri precedentemente illustrati, sono stati generati dati nuovi.
Tali dati sono stati costruiti sovrapponendo componenti armoniche o rumore di misura al segnale originario. In definitiva:

- i dati del validation set dovrebbero essere etichettati con la label "0" ad indicare che la classe di guasto è conosciuta,
- i dati artificiali dovrebbero essere etichettati con la label "1" ad indicare che la classe di guasto è non nota.

In particolare, i dati artificiali non sono utilizzati nella fase di train ma solo in quella di validation per valutare la bontà del modello.
Infatti, considerare tali dati per addestrare un modello di classificazione binaria, non avrebbe presumibilmente permesso di individuare qualsiasi tipologia di dato di classe *unknown* non previsto nel training. L'individuazione di un qualsiasi guasto *unknown* è invece un requisito necessario dal momento in cui non si conoscono i dati di test.

Questo è proprio il principio del classificatore "one-class" che impara dai dati di un'unica classe (classe 0) a riconoscere dati di classi differenti (classe 1).
In questo caso, si è utilizzato il classificatore "one-class" di tipo *One-Class SVM*, che utilizza l'algoritmo *Support Vector Machine* per identificare la regione che contiene la maggior parte dei dati normali durante l'addestramento e successivamente valuta la distanza di nuovi dati da questa regione per riconoscere le anomalie. Infatti, tale modello si basa sul calcolo di un threshold che corrisponde ad una soglia di decisione che separa i dati normali dalle anomalie. Questa soglia è ottimizzata in base ai criteri specifici dell'algoritmo *SVM*.

Per quanto riguarda la generazione dei dati artificiali per valutare le performance del classificatore, sono stati considerati due tipi di guasto *unknown* generati addizionando al segnale di pressione non affetto da guasto un rumore di misura o la somma di alcune armoniche fondamentali.

Nel primo caso, si è generato un vettore pseudo-casuale di numeri compresi tra 0 e 1 e della stessa lunghezza del segnale originale. Si è poi scalato tale vettore per un scalare che rappresenta la potenza di rumore. Il vettore così ottenuto è stato infine sommato al segnale originale.

In particolare, si è prestata attenzione a generare un segnale rumoroso che non sia nè troppo diverso dal segnale originale nè troppo simile ad esso. A tale scopo, sono state eseguiti diversi esperimenti variando il parametro di potenza.
Ad esempio, figura 2.4 mostra l'andamento del rumore generato con $potenza = 0.2$ e $potenza = 0.4$. Nel primo caso il segnale rumoroso è troppo simile al segnale originale, mentre, nel secondo caso, il modello di classificazione riesce a determinare la presenza di dati con guasto non noto.

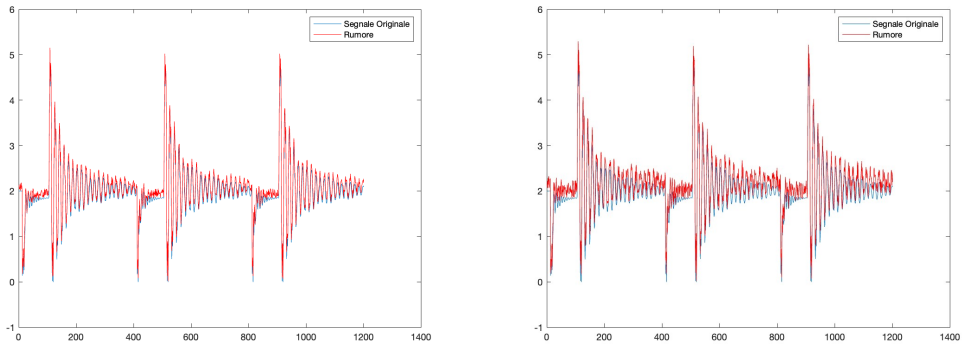


Figura 2.4: Creazione del segnale rumoroso con $potenza = 0.2$ e $potenza = 0.4$

La seconda tipologia di dati sintetici è stata generata sommando al segnale di pressione originario una o più armoniche fondamentali.
Un esempio è mostrato in figura 2.5.

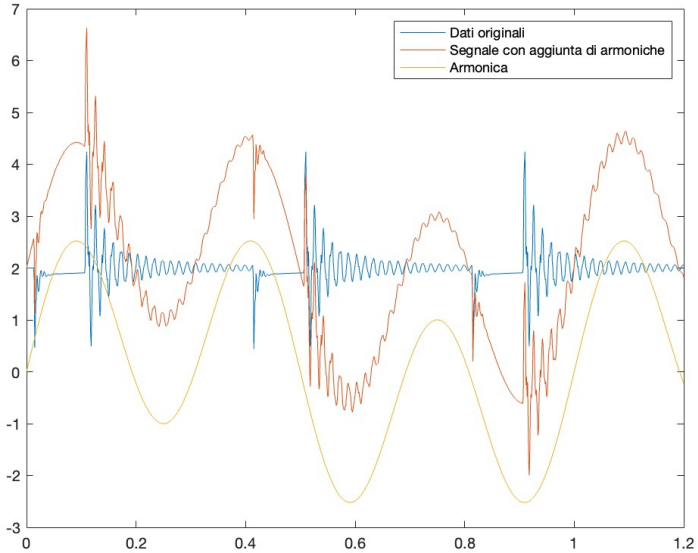


Figura 2.5: Aggiunta di armoniche al segnale originario

In particolare, sono stati eseguiti diversi esperimenti in modo tale da individuare la

frame policy opportuna e la soglia di voting per etichettare un esperimento con 1 piuttosto che 0. Come risultato, sono stati scelti:

- $frame\ \ rate = 0.128s$,
- $frame\ \ size = 0.128s$ in modo da non scartare parti del segnale.

Tale scelta della frame policy consente di non scartare parti di segnale utile ed è più veloce rispetto alla scelta $frame\ \ rate = 0.064s$ in quanto calcola le features per meno finestre. In seguito, dagli esperimenti eseguiti accentuando o meno il rumore, risulta che, soprattutto nei casi di aggiunta di rumore di misura al segnale, non sempre i dati *unknown* generati sono individuati con una confidenza elevata. In altre parole, raramente le finestre di uno stesso esperimento sono etichettate con 1 a maggioranza.
Al contrario, per alcuni esperimenti, viene associata l'etichetta 1 ad un'unica finestra del segnale. D'atra parte, i casi di guasto noto sono sempre riconosciuti come tali (classe 0).
Per tale ragione, a fronte del fatto che per un segnale di $1.2s$ la scelta di una $frame\ \ policy = 0.128s$ e di un $frame\ \ rate = 0.128s$ comporta la creazione di 10 finestre, si è deciso di settare il parametro di voting a 1.
Ovvero, basta che una sola di queste 10 finestre sia riconosicuta come guasto *unknown* per dire che tutto il segnale rappresenti un guasto non noto. Questo garantisce l'individuazione massima della classe guasto non noto anche nel caso in cui il segnale corrispondente sia molto simile ad un segnale di guasto noto (guasto sulla bolla o guasto sulla valvola).

2.3.2 Classificazione dei dati di guasto noti

I dati di guasto noto, anomalie sulle bolle e guasti sulle valvole, sono anche utilizzati per fare il training di un modello di classificazione che permettesse di distinguere tra le due classi.
In particolare, dopo aver eseguito lo splitting dei dati di training forniti dalla challenge in training set fittizio e validation set come spiegato precedentemente, si è ottimizzato il modello di classificazione tramite il tuning dei parametri.
In particolare, i risultati migliori sono stati ottenuti tramite la selezione delle prime 17 features selezionate in ordine decrescente secondo il ranking di figura 2.3 e con frame policy pari a $0.128s$.
Infatti, le prestazioni dei modelli ottenuti con frame policy pari a $0.064s$ sono analoghe al caso con $0.128s$. Dunque, a parità di accuracy, si è scelto il modello più veloce.
Inoltre, per quanto riguarda il parametro di voting scelto per ottenere una target unico per il segnale è 6 su 10. Questo significa che, serve una decisione a maggioranza dei voti delle singole finestre per scegliere per una determinata classe di guasto.
Poichè il numero di finestre è di ordine pari, nel momento in cui la maggioranza non venga raggiunta, il classificatore etichetterà il dato come classe 3 per default.

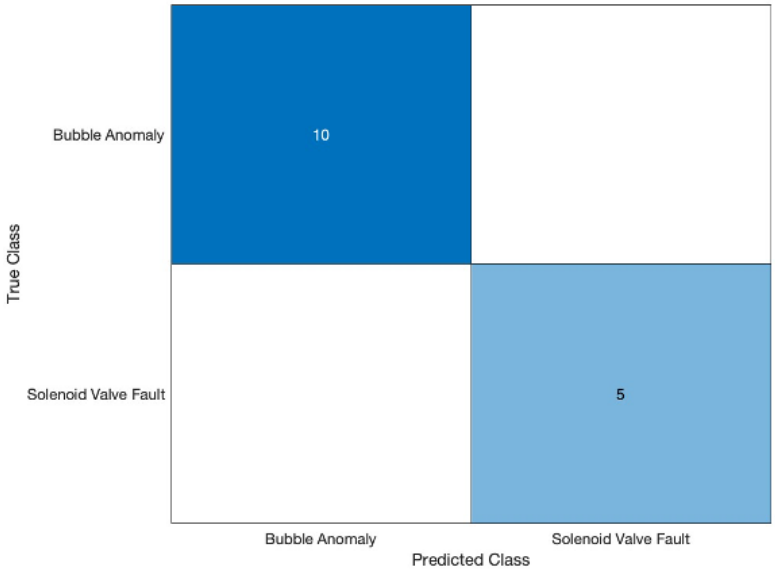


Figura 2.6: Matrice di confusione per la classificazione di guasti noti

Osservando le decisioni prese dal classificatore a runtime, la scelta di questo parametro come default non impatta negativamente l'accuratezza complessiva. Infatti, le classi risultano ben separabili in quanto la maggioranza per i dati di training viene sempre raggiunta.
L'accuratezza ottenuta per il validation set utilizzando il modello migliore, ovvero l'Optimizable Ensemble, è riportata in figura 2.6.

Capitolo 3

Task 3

Il task 3 ha come obiettivo quello di determinare il posizionamento della bolla per tutti i dati che nel precedente task sono stati etichettati come contaminazione da bolla. In particolare, sono 8 le possibili posizioni in cui la contaminazione può essere avvenuta:

- *BP1*, a cui è associata l'etichetta numerica 1;
- *BP2*, a cui è associata l'etichetta numerica 2;
- *BP3*, a cui è associata l'etichetta numerica 3;
- *BP4*, a cui è associata l'etichetta numerica 4;
- *BP5*, a cui è associata l'etichetta numerica 5;
- *BP6*, a cui è associata l'etichetta numerica 6;
- *BP7*, a cui è associata l'etichetta numerica 7;
- *BV1*, a cui è associata l'etichetta numerica 8.

3.1 Generazione delle feature

Una volta pre-processati i dati, si sono generate le feature nel dominio del tempo e nel dominio delle frequenze, affidandosi ancora una volta a Diagnostic Feature Designer.
Anche per il presente task, come per i due precedenti, si sono inizialmente testate le frame policy con frame rate $0.128s$ con frame size $0.128s$ e frame rate $0.064s$ con frame size $0.064s$. Confrontando i risultati dei classificatori, la frame policy $FS = FR = 0.128s$ si è dimostrata essere migliore, portando a valori di accuratezza maggiori.

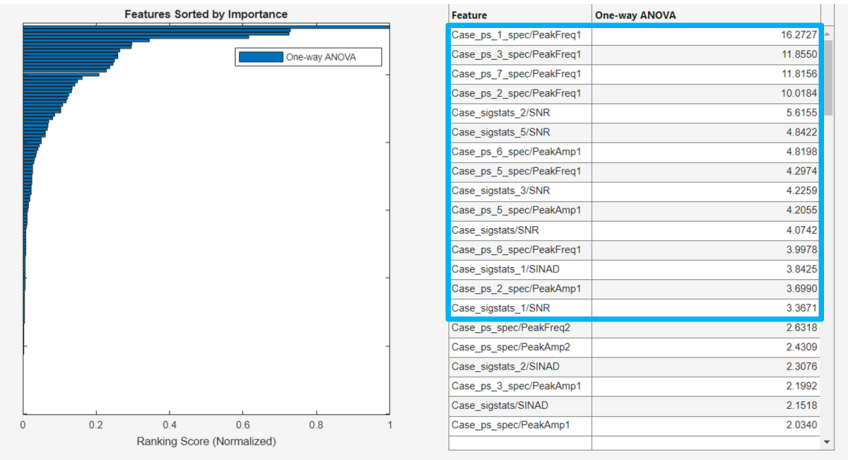
Anche in questo caso si sono generati gli spettri dei segnali facendo affidamento al modello parametrico AutoRegressive (AR).
L'ordine scelto per il modello è 10, che equivale a dire che esso si basa su 10 campioni

per effettuare la stima.

Le feature nel dominio del tempo sono state calcolate attraverso la funzionalità *Signal Features* dell'applicativo, che calcola feature statistiche di base dai segnali.

Le feature nella frequenza sono state ricavate attraverso la funzionalità *Spectral Features*.

Poiché il task in questione è di classificazione multiclasse, il ROC non può essere utilizzato per ordinare le feature.
Il metodo di ranking usato in questo caso è l'ANOVA.



Feature	One-way ANOVA
Case_ps_1_spec/PeakFreq1	16.2727
Case_ps_3_spec/PeakFreq1	11.8550
Case_ps_7_spec/PeakFreq1	11.8156
Case_ps_2_spec/PeakFreq1	10.0184
Case_sigstats_2/SNR	5.6155
Case_sigstats_5/SNR	4.8422
Case_ps_6_spec/PeakAmp1	4.8198
Case_ps_5_spec/PeakFreq1	4.2974
Case_sigstats_3/SNR	4.2259
Case_ps_5_spec/PeakAmp1	4.2055
Case_sigstats/SNR	4.0742
Case_ps_6_spec/PeakFreq1	3.9978
Case_sigstats_1/SINAD	3.8425
Case_ps_2_spec/PeakAmp1	3.6990
Case_sigstats_1/SNR	3.3671
Case_ps_spec/PeakFreq2	2.6318
Case_ps_spec/PeakAmp2	2.4309
Case_sigstats_2/SINAD	2.3076
Case_ps_3_spec/PeakAmp1	2.1992
Case_sigstats/SINAD	2.1518
Case_ps_spec/PeakAmp1	2.0340

Figura 3.1: Rank Features task 3

In figura 3.1 è riportato il feature ranking secondo l'ANOVA.
Le feature incluse nella bordatura di colore azzurro sono quelle scelte per l'addestramento del classificatore.
Tali feature sono state esportate come tabella nel workspace di MATLAB, salvate per eventuali usi futuri e per poi esser importate in Classification Learner.

3.2 Addestramento del modello di classificazione

3.2.1 Training del modello

Si sono addestrati a questo punto i modelli sulle migliori feature ottenute dal ranking.

I dati sono stati divisi i dati in modo da avere $\frac{2}{3}$ di dati di train e $\frac{1}{3}$ di dati per il testing del modello.

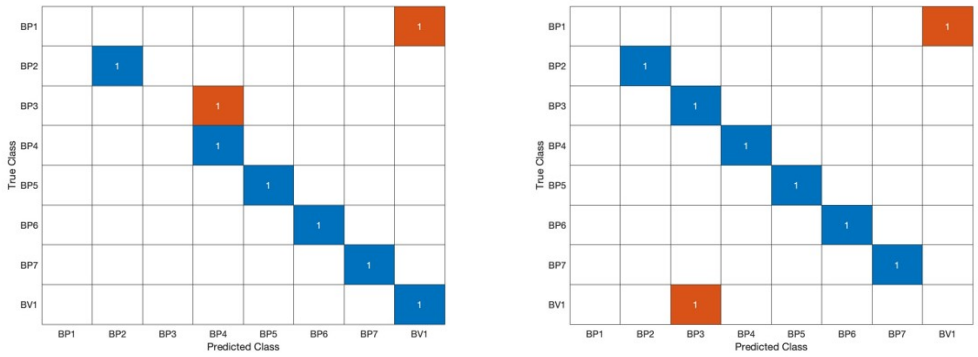


Figura 3.2: Matrici di confusione per il task 3 sul validation set

In figura 3.2 sono riportate le matrici di confusione ottenute testando il classificatore sui dati di test (a sinistra $FR = 0.064s$ e $fs = 0.064s$ e a destra $FR = 0128s$ e $FR = 0.128s$).
I due classificatori sono caratterizzati dalla stessa accuracy di test, pari al 75%.
Entrambi confondono un dato di classe $BP1$ classificandolo come appartenente alla classe $BV1$ (ricordiamo che le due bolle sono in serie).
Il classificatore con $FR = 0.064s$ classifica un campione di classe $BP3$ come uno di classe $BP4$ (configurazione in parallelo). Il classificatore con $FR = 0.128s$ classifica un campione di classe $BV1$ assegnandogli, poco comprensibilmente vista la distanza delle due bolle nello schema, la classe $BP3$.

Considerato che i classificatori sono caratterizzati da una stessa accuracy, poiché la frame policy con $FR = 0.128s$ determina un numero minore di finestre, si è scelto come classificatore migliore quello con $FR = 0.128s$.

3.2.2 Split dei segnali in sotto-finestre

La numerosità dei dati a disposizione per questo task e per i successivi è piuttosto ridotta, pari a 24 per il task 3 e 48 per i task 4 e 5, determinando un numero di campioni insufficiente per una valutazione esaustiva dei modelli addestrati e della generalizzazione di questi. Ad esempio, nel caso del task 3, per poter avere almeno un dato di test per ciascuna classe, si avrebbero solo 2 dati di training per ogni classe, un numero insufficiente sia di dati di training che di test.

Per sopperire a questa problematica in modo da avere più dati a disposizione, soprattutto per testare il modello su dati di cui le etichette sono note, si è adottata una soluzione che prevede di dividere ciascuna acquisizione in n sotto-segnali. In particolare, poiché l'acquisizione di ogni segnale di pressione fa riferimento ad un intervallo temporale entro il quale si realizza il processo di apertura ($300ms$) e di

chiusura ($100ms$) per tre volte, si è scelto di porre $n = 3$, ossia dividere ogni segnale nella sua interezza in tre sotto-segnali.
Così facendo si sono triplicati i dati a disposizione, arrivando ad un numero totale di 72 campioni per il task 3 e 144 per il 4 e il 5.
Questo permette anche di valutare se il modello ha una buona generalizzazione.

Inoltre, per evitare che i dati di training fossero sbilanciati sia in termini di classi rappresentate che di spacecraft da cui i dati provengono si è implementato un algoritmo di splitting con l'obiettivo di avere almeno un campione di ogni classe e relativi a differenti spacecraft nel testset.
Tale algoritmo prevede una suddivisione randomica e stratificata di ciascuna classe, in modo che nessuna classe sia sotto-rappresentata rispetto alle altre. La suddivisione randomica è stata adottata anche e soprattutto per far sì che al modello venissero dati in input campioni ottenuti da diversi spacecraft perché, come specificato dalla challenge, ad ogni spacecraft è associata una differente difficoltà di classificazione. Inoltre, effettuando differenti prove si è osservato che le performance ottenibili variano a seconda della suddivisione in termini di spacecraft che si realizza.
In sintesi, attraverso lo splitting randomico, si è potuto addestrare il modello su dati più eterogenei e rappresentanti ciascuno degli spacecraft, anziché adattarsi ai dati generati da uno specifico spacecraft.

Essendo i dati di partenza 24 ed essendoci 3 campioni per ciascuna classe, dopo lo split si sono ottenuti 72 campioni totali di cui 9 per ogni classe.
Si è deciso di dividere il dataset in modo che $\frac{1}{3}$ dei campioni venisse usato per il testing e i $\frac{2}{3}$ per il training. In questo modo, seguendo l'algoritmo di splitting sopra citato, 3 campioni di ciascuna classe, randomicamente, sono stati inseriti nel dataset di testing e i restanti dati sono stati aggiunti al dataset di training.

A questo punto, si sono calcolate le feature dei dati ottenuti dallo split (sia con frame policy $FR = 0.064s$ e $FS = 0.064s$ che con $FR = 0.128s$ e $FS = 0.128s$), per poi importarle in Classification Learner. Si sono così addestrati tutti i classificatori disponibili nell'applicativo.
I risultati ottenuti con l'Optimizable Ensemble sono riassunti dalla matrice di confusione sui dati di test è riportata in figura 3.3.

Emerge come il classificatore addestrato sulle feature ottenute con frame policy $FS = 0.064s$ e $FR = 0.064$ abbia performance leggermente migliori (un misclassificato in meno) rispetto all'altra frame policy.
In particolare, la classe $BV1$ viene confusa con la classe $BP1$. Questo è probabilmente dovuto al fatto che le due bolle sono in serie, quindi l'effetto di variazione di una si ripercuote sull'altra e viceversa.
In un caso un campione di classe $BP4$ viene classificato come di classe $BP3$. Queste due bolle sono in parallelo e quindi è comprensibile che il modello si confonda.

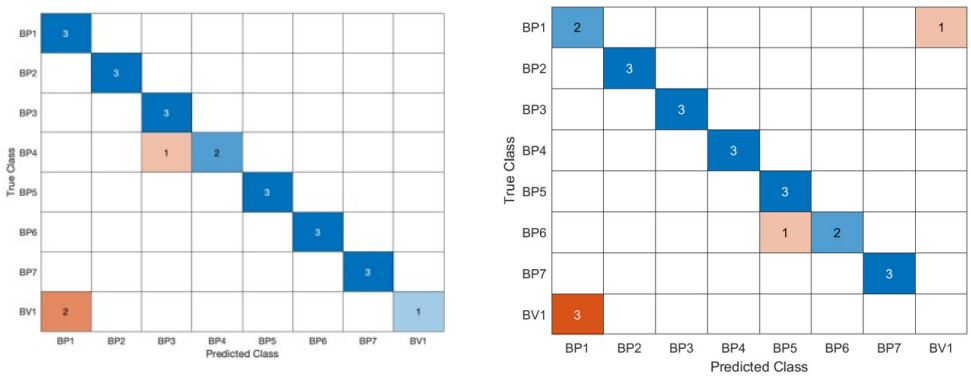


Figura 3.3: Matrici di confusione sui dati splittati

Si tenga presente che le matrici ottenute derivano da una suddivisione randomica dei dati, pertanto i dati usati per il training in un caso (e di conseguenza quelli usati per il testing) sono diversi da quelli usati nell'altro caso.

Capitolo 4

Task 4

Il task 4 ha come obiettivo quello di determinare quale valvola si è guastata, a partire dai dati che nel task 2 sono stati identificati come guasto sulla valvola. In particolare, sono presenti quattro valvole:

- *SV1*, a cui è associata l'etichetta numerica 1;
- *SV2*, a cui è associata l'etichetta numerica 2;
- *SV3*, a cui è associata l'etichetta numerica 3;
- *SV4*, a cui è associata l'etichetta numerica 4.

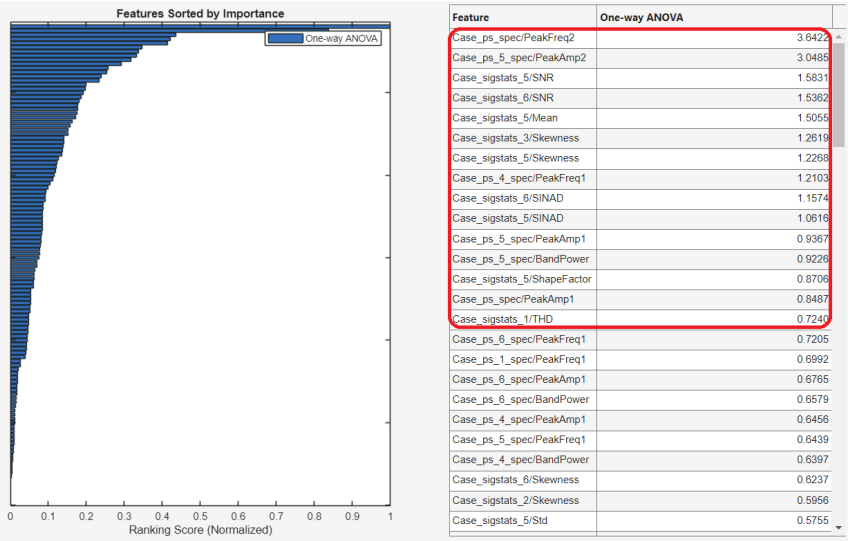
4.1 Generazione delle feature

Dopo aver effettuato una prima fase di pre-processing dei dati, sono state generate le feature, sia nel dominio del tempo che nel dominio della frequenza sfruttando le funzionalità messe a disposizione da Diagnostic Feature Designer.
Come nel caso dei task precedenti, inizialmente, si è provato ad applicare una frame policy con frame rate $0.128s$ e frame size $0.128s$ e con frame rate $0.064s$ e frame size $0.064s$. Non del tutto soddisfatti dei risultati ottenuti, si è deciso di fare un'ulteriore prova, utilizzando una frame policy con frame rate $0.256s$ e frame size $0.256s$.

Successivamente, sono state estratte le feature nel dominio del tempo tramite la funzionalità *Signal Features*. Dopodiché, sono stati generati gli spettri, ancora una volta utilizzando il modello parametrico AutoRegressive (AR) con ordine pari a 10 e sono state estratte le feature nel dominio della frequenza tramite la funzionalità *Spectral Features*.

Nella figura 4.1 è riportato il ranking delle feature, ordinato utilizzando l'analisi della varianza (ANOVA), poichè si tratta di un problema di classificazione multiclasse.

Le feature comprese nel riquadro con bordo rosso rappresentano le 15 feature che si è deciso di utilizzare per effettuare l'addestramento del modello.



Feature	One-way ANOVA
Case_ps_spec/PeakFreq2	3.6422
Case_ps_5_spec/PeakAmp2	3.0485
Case_sigstats_5/SNR	1.5831
Case_sigstats_6/SNR	1.5362
Case_sigstats_5/Mean	1.5055
Case_sigstats_3/Skewness	1.2619
Case_sigstats_5/Skewness	1.2268
Case_ps_4_spec/PeakFreq1	1.2103
Case_sigstats_6/SINAD	1.1574
Case_sigstats_5/SINAD	1.0616
Case_ps_5_spec/PeakAmp1	0.9367
Case_ps_5_spec/BandPower	0.9226
Case_sigstats_5/ShapeFactor	0.8706
Case_ps_spec/PeakAmp1	0.8487
Case_sigstats_1/THD	0.7240
Case_ps_6_spec/PeakFreq1	0.7205
Case_ps_1_spec/PeakFreq1	0.6992
Case_ps_6_spec/PeakAmp1	0.6765
Case_ps_6_spec/BandPower	0.6579
Case_ps_4_spec/PeakAmp1	0.6456
Case_ps_5_spec/PeakFreq1	0.6439
Case_ps_4_spec/BandPower	0.6397
Case_sigstats_6/Skewness	0.6237
Case_sigstats_2/Skewness	0.5956
Case_sigstats_5/Std	0.5755

Figura 4.1: Rank Feature task 4

4.2 Addestramento del modello

4.2.1 Training del modello

Dopodiché, è stato addestrato il classificatore. Sono state eseguite tre prove, una per ciascuna delle frame policy precedentemente introdotte, e i dati sono stati suddivisi in modo tale che i due terzi fossero assegnati al training set, mentre il rimanente terzo al test set.

Le matrici di confusione ottenute sono riportate in figura 4.2, rispettivamente nell'ordine $FR = FS = 0.064s$, $FR = FS = 0.128s$ e $FR = FS = 0.256s$. È evidente che in entrambi i casi sono presenti due errori, pertanto l'accuratezza è la stessa per ciascuna delle matrici. Tuttavia, considerando che la frame policy con $FR = 0.256s$ determina un numero minore di finestre, si è deciso di adottare quel classificatore.

Va notato che, dato che in tutti i casi sono state applicate delle frame policy, per non assegnare una classe ad ogni singola finestra ma ad ogni campione, è stata presa una decisione basata sulla maggioranza dei voti ottenute dalle singole finestre.

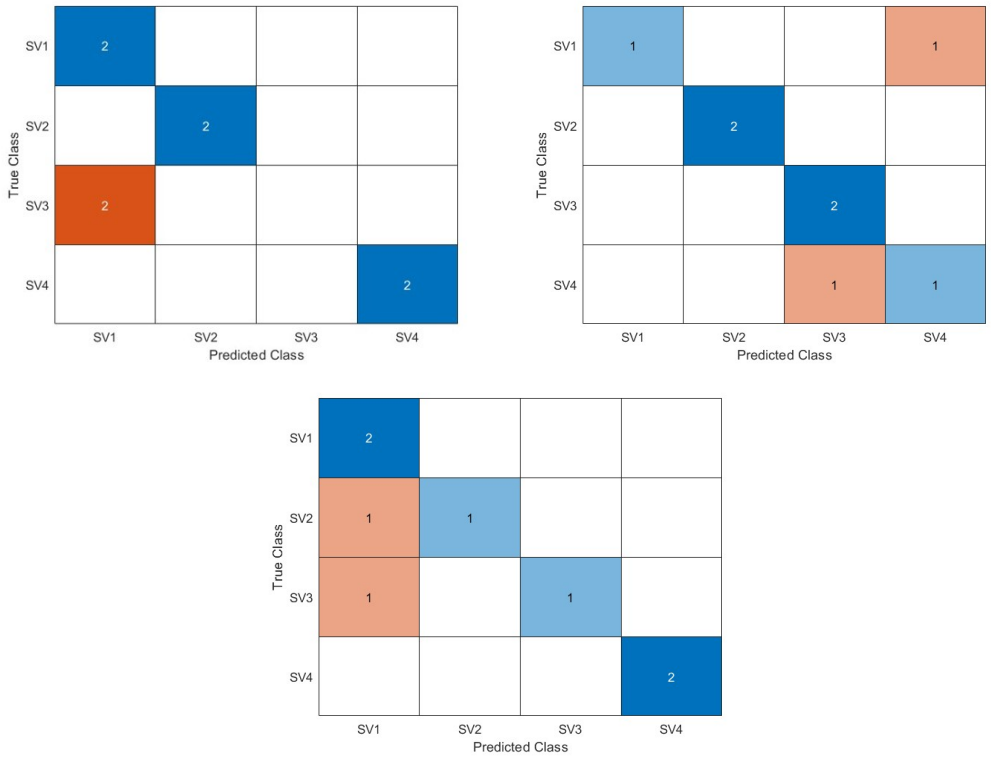


Figura 4.2: Matrici di confusione per il task 4 sul validation set

4.2.2 Split dei segnali in sotto-finestre

Come nel caso del task 3, vista la scarsa quantità di dati a disposizione, si è deciso di suddividere il segnale in 3 sotto-segnali. Così facendo si sono ottenuti 144 campioni, invece dei 48 iniziali. Inoltre, per suddividere in maniera randomica i dati tra train set e test set, è stato applicato il medesimo algoritmo di splitting del caso precedente, ma in questo caso, essendo i dati in numero maggiore, vengono inseriti in maniera random 6 campioni nel dataset di testing.

Dopodiché, sono state generate le feature utilizzando i dati ottenuti dopo lo split con le diverse frame policy:

- $FS = 0.064s$ e $FR = 0.064$
- $FS = 0.128s$ e $FR = 0.128$

In questo caso, si è deciso di non effettuare una prova con $FS = 0.256$ e $FR = 0.256$ poiché, avendo suddiviso i dati in finestre di $400ms$, avere una frame rate e una frame size così ampi potrebbe non permettere di catturare caratteristiche rilevanti del segnale.

Le feature ottenute sono state importate su Classification Learner per l'addestramento dei modelli. In entrambi i casi, il classificatore che ha riportato dei risultati migliori

è l'Optimizable Ensemble. Le rispettive matrici di confusione sono riportate nella figura 4.3.

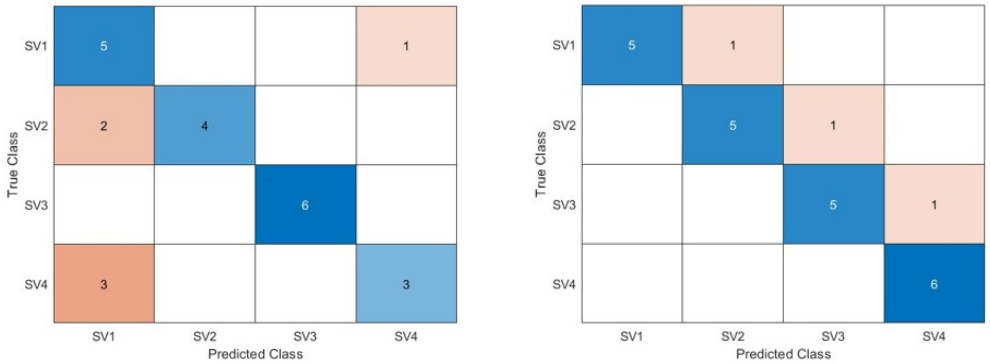


Figura 4.3: Matrici di confusione sui dati splittati

Come si può evincere, la matrice di confusione ottenuta con $FS = 0.128s$ e $FR = 0.128$ (a destra) ha fornito dei risultati migliori, infatti, quasi tutti gli elementi al di fuori della diagonale sono nulli. Al contrario, nella matrice di confusione ottenuta nella prova svolta con $FS = 0.064s$ e $FR = 0.064$ (a sinistra) vengono commessi più errori e molti dati errati sono stati assegnati alla classe SV1. In particolare, si nota che sia 2 campioni di SV2 che 3 campioni di SV3 vengono confusi con la classe SV1. Tuttavia, va notato che le matrici riportate sono state ottenute a fronte di differenti suddivisioni tra training e test poiché è stata applicata una suddivisione randomica dei dati.

Capitolo 5

Task 5

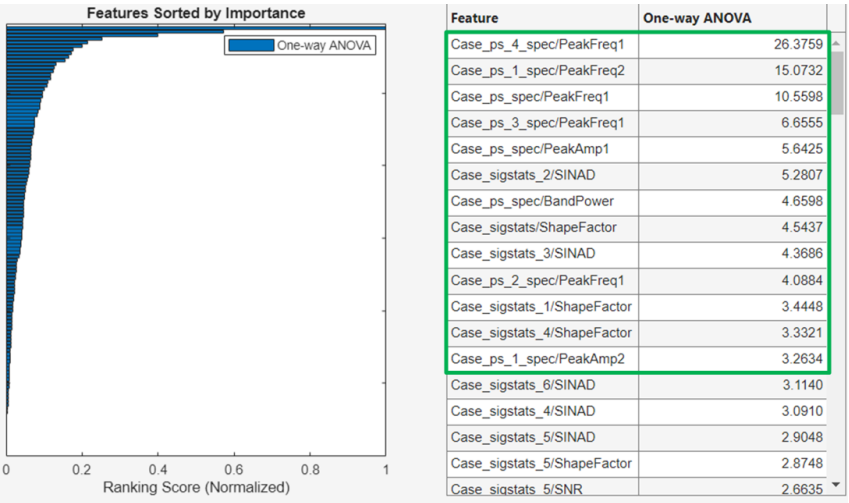
Il task 5 ha come obiettivo quello di determinare la percentuale di apertura della valvola per i dati che sono stati identificati come appartenenti alla classe 3 nel task 2. I valori possibili sono compresi tra 0% e 100%.

5.1 Generazione delle feature

Una volta pre-processati i dati, si sono generate le feature nel dominio del tempo e nel dominio delle frequenze, affidandosi ancora una volta a Diagnostic Feature Designer.
Anche per il presente task, come per i due precedenti, si sono inizialmente testate le frame policy con frame rate $0.064s$ e con frame size $0.064s$, con frame rate $0.128s$ e con frame size $0.128s$ e infine con frame rate $0.256s$ e con frame size $0.256s$.
Confrontando i risultati dei classificatori, la frame policy $FS = FR = 0.128s$ si è dimostrata essere migliore, portando a valori di accuratezza maggiori.

Anche in questo caso si sono generati gli spettri dei segnali facendo affidamento al modello parametrico AutoRegressive scegliendo come ordine del modello 10.

In questo caso, lo sviluppo del task è stato fatto seguendo sia la strada della classificazione che quella della regressione.
In entrambi i casi si è adottato come metodo di ranking delle feature l'ANOVA.
Per la regressione esistono metodi di ranking delle feature specifici, come il Kruskal-Wallis. Tuttavia, anche usando un altro metodo di ranking, le performance di classificazione sono risultate essere le stesse.
La decisione finale è stata quella di usare come metodo di ranking l'ANOVA e di usare per classificazione e regressione le stesse feature.



Feature	One-way ANOVA
Case_ps_4_spec/PeakFreq1	26.3759
Case_ps_1_spec/PeakFreq2	15.0732
Case_ps_spec/PeakFreq1	10.5598
Case_ps_3_spec/PeakFreq1	6.6555
Case_ps_spec/PeakAmp1	5.6425
Case_sigstats_2/SINAD	5.2807
Case_ps_spec/BandPower	4.6598
Case_sigstats/ShapeFactor	4.5437
Case_sigstats_3/SINAD	4.3686
Case_ps_2_spec/PeakFreq1	4.0884
Case_sigstats_1/ShapeFactor	3.4448
Case_sigstats_4/ShapeFactor	3.3321
Case_ps_1_spec/PeakAmp2	3.2634
Case_sigstats_6/SINAD	3.1140
Case_sigstats_4/SINAD	3.0910
Case_sigstats_5/SINAD	2.9048
Case_sigstats_5/ShapeFactor	2.8748
Case_sigstats_5/SNR	2.6635

Figura 5.1: Rank Feature task 5

5.2 Addestramento del modello di classificazione

5.2.1 Training del modello di classificazione

Ricavate e ordinate le feature, queste sono state usate per addestrare i modelli di classificazione, valutando le prestazioni risultanti da differenti scelte in termini di frame policy.
Nel paragrafo precedente è stato indicato 13 come numero di feature selezionato per l'addestramento, tuttavia, sia per la classificazione che per la regressione, prima di arrivare a quello ottimale, si sono fatte delle prove variando il numero di feature usate per l'addestramento. 13 feature hanno determinato l'accuratezza maggiore per la classificazione e l'RMSE minore per la regressione, pertanto verranno discussi solo i modelli addestrati su 13 feature.

Anche in questo caso si è diviso il dataset secondo il rapporto 80/20.

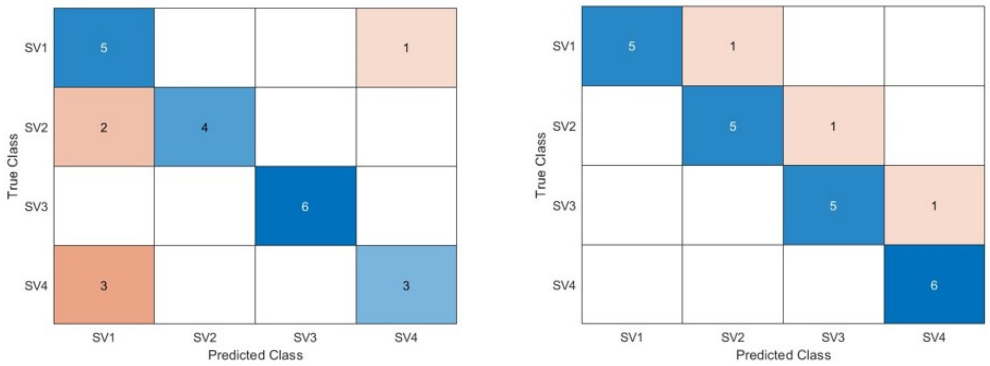


Figura 5.2: Matrici di confusione per frame policy $FR = 0.064s$ e $FS = 0.064s$ (a sinistra) e frame policy $FR = 0.128s$ e $FS = 0.128s$ (a destra).

Dalle matrici di confusione emerge come il modello derivante dall'applicazione della frame policy con $FR = 0.128s$ e $FS = 0.128s$ comporta un minor numero di misclassificati.
Pertanto, questo è il modello che è stato utilizzato per classificare i dati effettivi di test.

5.2.2 Split dei segnali in sotto-finestre

Come per i precedenti due task, vista la scarsa quantità di dati a disposizione, si è deciso di suddividere il segnale in 3 parti e per ricavare i dati di test è stato applicato l'algoritmo di splitting già descritto.

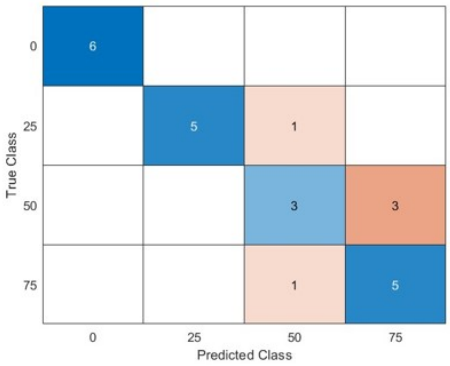


Figura 5.3: Matrice di confusione sui dati splittati

In questo caso, si è deciso di effettuare una prova solo con $FR = 0.064$ e $FR = 0.128$.
Le feature calcolate sui segnali splittati sono state importate su Classification Learner e si sono addestrati i modelli.
Ancora una volta il modello migliore è risultato essere l'Optimizable Ensamble.
I risultati sui dati di test etichettati sono riportati in figura 5.3.

5.2.3 Training del modello di regressione

Oltre alla strada della classificazione è stata percorsa anche la strada della regressione, che è quella effettivamente richiesta dalla challenge.

I dati di training messi a disposizione dalla competizione sono stati divisi anche in questo caso in un sottoinsieme per l'addestramento e in uno per il testing secondo il rapporto 80/20.

In figura 5.4 è riportato uno scatterplot in cui sull'asse delle x è riportato il numero ad identificare il campione. Sull'asse delle y è indicato il valore di apertura della valvola.

I punti sono colorati in rosso e in verde ad indicare, rispettivamente, il valore predetto e il valore reale di apertura della valvola.

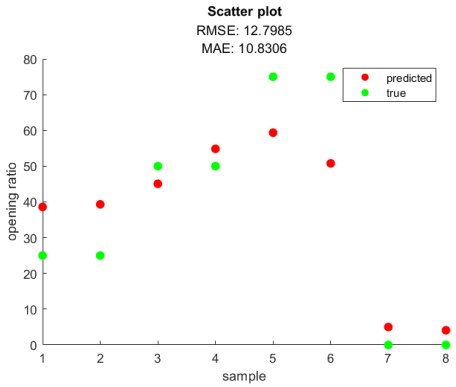


Figura 5.4: Scatter plot sulla regressione.

L'RMSE e il MAE ottenuti sono pari a 12.7985 e 10.8306, rispettivamente.
Si può osservare che in alcuni casi il regressore riesce ad avvicinarsi molto al valore reale dell'apertura, come per esempio nel caso dei sample 7 e 8, in altri, invece, commette un errore non trascurabile, come per il caso del sample 6.

Capitolo 6

Testing

In tale capitolo si discutono i risultati ottenuti in ciascuno dei task a partire da nuovi modelli addestrati in questo caso sulla totalità dei dati che la challenge propone come training set e testati sui dati parte del test set, finora mai utilizzato.
La configurazione dei modelli è scelta sulla riga di quando emerso per la fase precedente, così come il sottoinsieme di feature generate e estratte sono le stesse individuate in fase di validazione. In particolare, per poter estrarre esattamente le stesse feature, piuttosto che farlo manualmente, si sono impiegate le rispettive funzioni di generazione delle feature, esportate da Diagnostic Feature Designer, che sono raccolte nella sotto cartella del progetto $testing/feature$.

Per valutare le performance dei modelli, oltre alla matrice di confusione, viene proposta una visualizzazione tridimensionale dei dati di test, in cui gli assi sono stati scelti in base alle feature più significative nel singolo task. Questa valutazione era stata pensata in origine quando non erano disponibili le etichette dei dati di test come modo euristico per valutare le performance.

Il testing e i risultati di seguito riportati sono l'output dell'esecuzione del file $tasks.m$ che si occupa di:

- generare le opportune feature a partire dai dati di testing, garantendo che siano le stesse estratte a partire dal training set e le stesse su cui quindi è addestrato il modello di classificazione;
- caricare il modello di classificazione addestrato;
- predire le classi per ogni caso;
- calcolare l'accuratezza e la matrice di confusione;
- generare il plot tridimensionale dei dati.

In realtà, per tutti i task di classificazione, ad eccezione del primo, il risultato dell'addestramento del modello verrà discusso riportando due matrici di confusione.
La prima matrice di confusione è calcolata sulla base del formato di consegna dei risultati richiesto dalla challenge, mentre la seconda matrice di confusione proposta

prende in considerazione solo i dati effettivamente classificati dai nostri modelli.

Infatti, la suddivisione in task imposta dalla competizione è, di fatto, una classificazione in cascata, in cui i risultati dei task precedenti influenzano quelli dei task successivi.
Ad esempio, nel caso del task 2 la competizione richiede di eseguire la classificazione in *unknown*, *bubble anomaly* e *solenoid fault* solo sui dati classificati come *abnormal* nel task 1. Alle tre classi devono essere assegnate, rispettivamente, le etichette 1, 2 e 3. I casi etichettati come *normal* nel task 1 devono essere indicati con l'etichetta 0. Quindi, per il task 2 la prima matrice di confusione riporta la classe *other*, ad indicare tutti quei casi che nel task 1 sono stati classificati come *normal*.
Nel caso del task 3, invece, solo quei dati classificati come *bubble anomaly* nel task 2 devono essere considerati per la classificazione nelle diverse classi di anomalia. Tutti gli altri casi, sia quelli classificati come normali nel task 1 che quelli classificati come *solenoid fault*, devono esser etichettati con la label 0, ovvero *other* nella matrice di confusione.
Analogamente, nel task 4 e 5 sono stati classificati solo quei dati che nel task 2 erano stati etichettati come *valve fault*. Al resto dei casi viene attribuita la classe 0, ovvero la classe *other* nella matrice.

La seconda matrice di confusione, invece, esclude la classe *other* e considera solo i dati che il classificatore ha effettivamente visto e classificato, quindi riporta solo le classi che il classificatore addestrato per quel task effettivamente conosce.

Per ogni matrice di confusione, è anche riportato lo score di accuracy ottenuto sommando la diagonale principale e dividendo tale valore per il numero totale di elementi.
Infine, è anche stata calcolata un'ulteriore accuratezza denominata *total accuracy*. Tale metrica è il risultato della somma della diagonale principale della seconda matrice di confusione diviso per il numero totale di elementi della prima matrice una volta eliminata la colonna di dati classificati come *other* o *normal* nel task precedente. Questa metrica è utile per individuare quei campioni miss-classificati nel task precedente e dunque per considerare come si distribuiscono gli errori commessi in cascata.
Nel caso in questione, la *total accuracy* e l'accuratezza relativa alla seconda matrice di confusione coincidono sempre in quanto i dati in input ad ogni task non sono mai confusi con quelli di altre classi. Ad esempio, per quanto riguarda il task 1, nessuno dei dati *normal* è erroneamente classificato come *abnormal*.
Tutti i valori di accuratezza calcolati sono necessari per valutare le performance. Ad esempio, considerare solo l'accuratezza della matrice di sinistra, potrebbe falsare lo score in quanto tiene conto anche della classificazione eseguita per i task precedenti. Ad esempio, nel task 4 o 5 che richiedono pochi dati in input, il calcolo risulta troppo

ottimistico considerando anche la predizione fatta per il task 1 che utilizza molti più dati.

6.1 Task 1

Il modello utilizzato per il task 1 è basato sull'algoritmo di classificazione Optimizable Ensamble, con i parametri configurati in base a quanto emerso nella fase di validation, addestrato su un totale di 20 feature estratte a partire dall'intero training set disponibile, applicando una frame policy caratterizzata da $FR = FS = 0.128s$.
Si è quindi dato in input al modello il test set e si sono predette le etichette di ciascun caso.

In figura 6.1 è riportata la matrice di confusione ottenuta. Come accadeva nella fase preliminare del progetto il modello è in grado di classificare correttamente tutti i casi di normale funzionamento, mentre sbaglia la predizione della classe di alcuni casi di funzionamento anormale.
L'accuratezza ottenuta, calcolata a partire dalla matrice è pari a 91.3%.

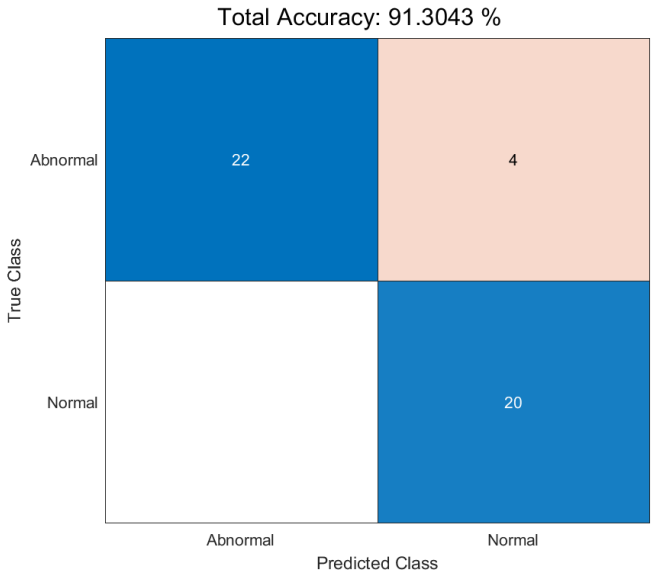


Figura 6.1: Matrice di confusione del task 1 sui dati di test

In figura 6.2 è riportato il plot dei dati in uno spazio tridimensionale: a sinistra è associato un colore a ciascuno dato in base alla classe predetta (arancione per normal, giallo per abnormal), mentre a destra è associato un colore a ciascuno dato in base alla corretta o errata classificazione del dato.
Si evidenza il fatto che i dati proiettati nelle tre dimensioni corrispondenti alle feature con ranking più alto non sono così facilmente separabili, come invece sarebbe

desiderabile: non è possibile in alcun modo tracciare un piano che separi una classe dall'altra.

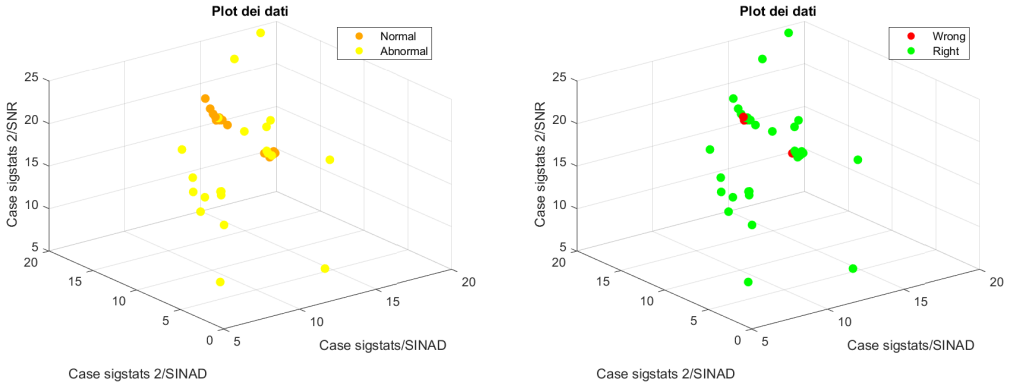


Figura 6.2: Plot tridimensionale dei dati secondo le tre feature con ranking maggiore nel task 1

6.2 Task 2

Una volta riaddestrato il modello Optimizable Ensamble con i parametri scelti nella fase di validation e considerando in input tutti i dati di training, sono state calcolate le matrici di confusione riportate in figura 6.3.

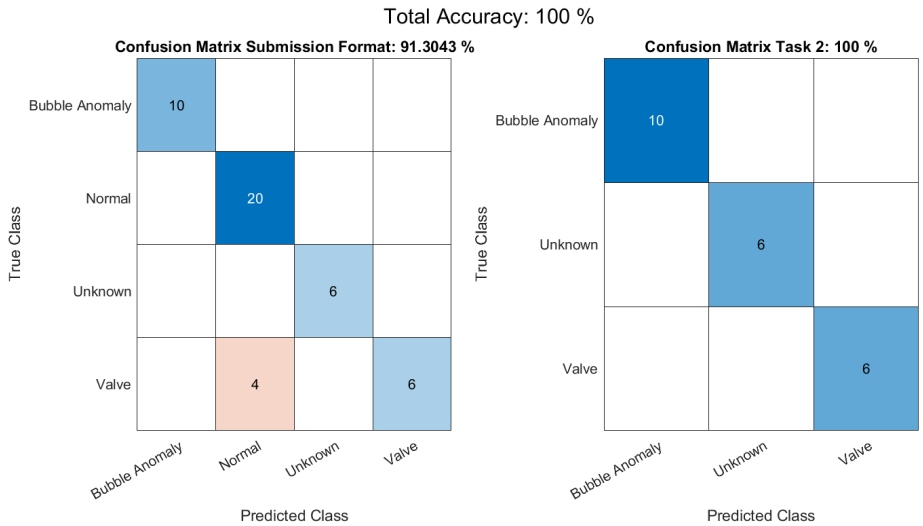


Figura 6.3: Matrice di confusione del task 2 sui dati di test

In questo caso gli score ottenuti indicano accuratezza massima, ovvero, sia i dati classificati come guasto non noto sia i dati classificati come guasto su bolla e guasto su valvola sono indovinati correttamente.

In figura 6.4 e 6.5 sono mostrati degli scatter plot relativi alla visualizzazione tridimensionale delle tre features più importati per ogni modello secondo il relativo ranking. Nonostante tali grafici non siano esaustivi in quanto rappresentano solo una parte dello spazio delle features, è possibile notare come i punti si distribuiscano nello spazio definendo una bordo di decisione piuttosto netto.

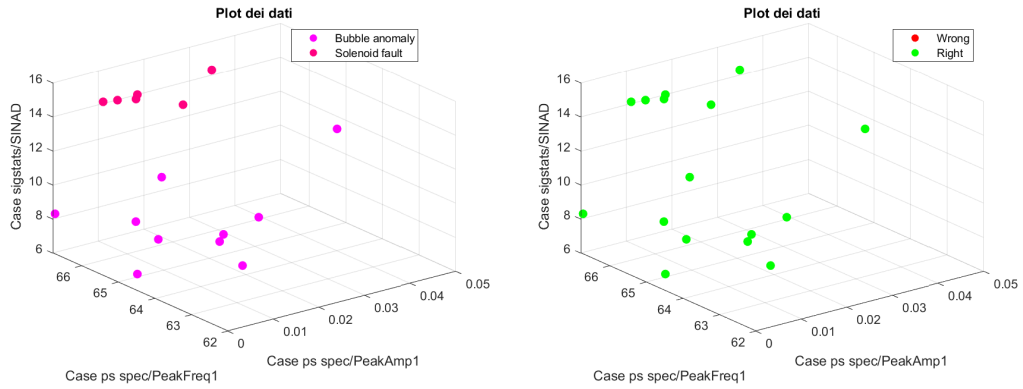


Figura 6.4: Plot tridimensionale dei dati secondo le tre feature con ranking maggiore nel task 2, dati di guasto noto

Questo appare sopratutto evidente nella visualizzazione dei dati di classe di guasto non nota in quanto questi risultano piuttosto differenti rispetto agli altri dati dislocandosi molto più distanti nello spazio.

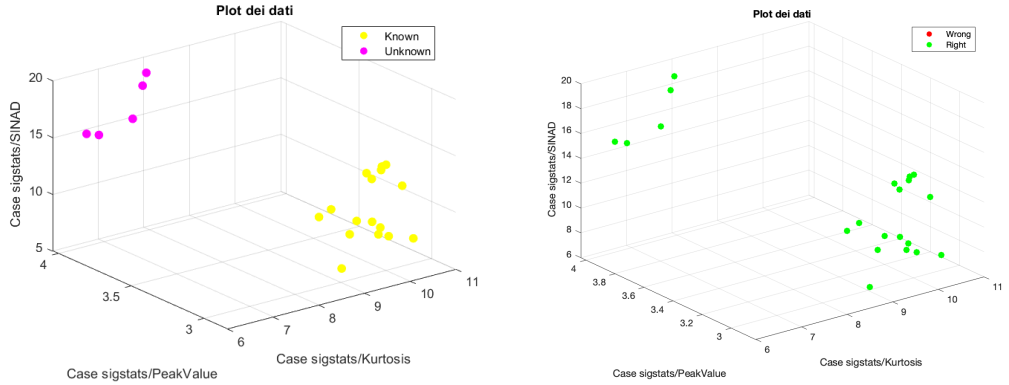


Figura 6.5: Plot tridimensionale dei dati secondo le tre feature con ranking maggiore nel task 2, dati di guasto non noto

6.3 Task 3

Una volta importate le feature dell'intero dataset di training in Classification Learner, si sono selezionati tutti gli algoritmi di classificazione messi a disposizione dall'applicativo.
Anche questo caso il modello migliore è l'Optimizable Ensamble. A seguito dell'addestramento, il modello è stato esportato per poter essere testato sui dati di test.

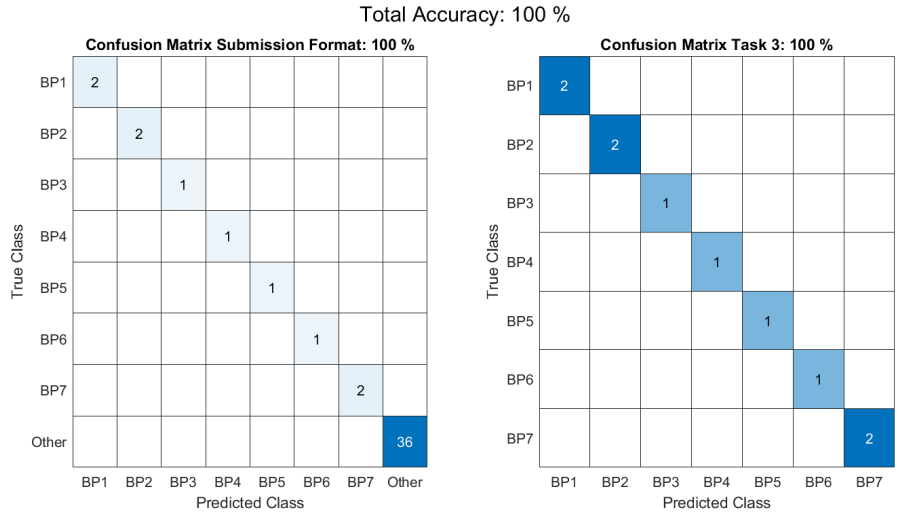


Figura 6.6: Matrice di confusione del task 3 sui dati di test

Il risultato dell'addestramento del modello è sintetizzato con le matrici di confusione in figura 6.6.
In entrambi i casi, sia che si consideri l'accuratezza del classificatore secondo il formato della challenge sia che la si consideri solo sui dati effettivamente dati in pasto al classificatore, l'accuratezza è massima.
Infatti, in questo task, anche considerando i dati misclassificati nel task 1, siccome questi dati erroneamente predetti sono appartenenti alla classe valvola e sono stati classificati come appartenenti alla classe normale, rispettando il formato di sottomissione, sono stati etichettati con la label 0, ovvero l'etichetta corretta.
Inoltre, tutti i dati anomali vengono classificati correttamente.

Nella figura 6.7 si riportano due grafici in cui ciascun punto rappresenta un campione del dataset, mentre sugli assi sono riportate le prime tre feature ottenute dal feature ranking.
Nel grafico a sinistra ogni elemento di test è colorato in base alla classe di appartenenza; nel grafico sulla destra i punti sono colorati in rosso o in verde a seconda che, rispettivamente, siano stati classificati scorrettamente o correttamente dal classificatore.

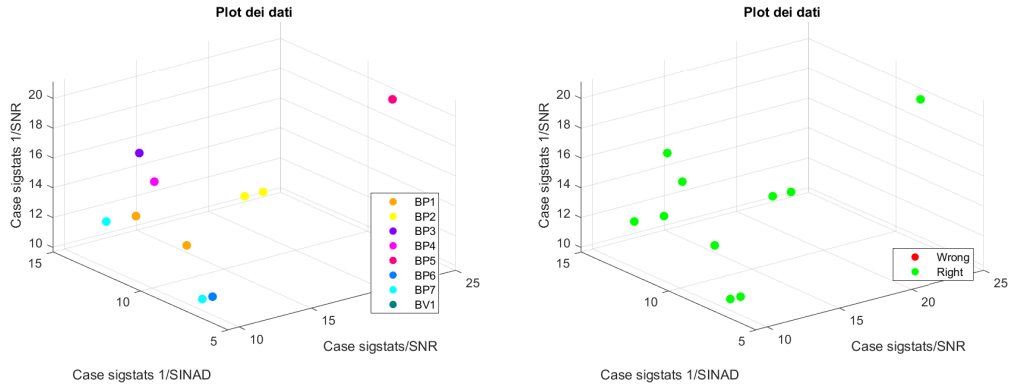


Figura 6.7: Plot tridimensionale dei dati secondo le tre feature con ranking maggiore nel task 3

6.4 Task 4

Per effettuare la classificazione, si è nuovamente fatto affidamento al tool Classification Learner.
Dopo aver importato le feature precedentemente scelte per questo task e applicando la medesima frame policy ($FS = FR = 0.256s$), è stato svolto l'addestramento tramite l'algoritmo Optimizable Ensemble, ossia l'algoritmo che aveva prodotto i risultati migliori nella precedente fase di training.

Per valutare le performance della classificazione, sono state utilizzate, ancora una volta, due matrici di confusione che sono riportate nella figura 6.8.

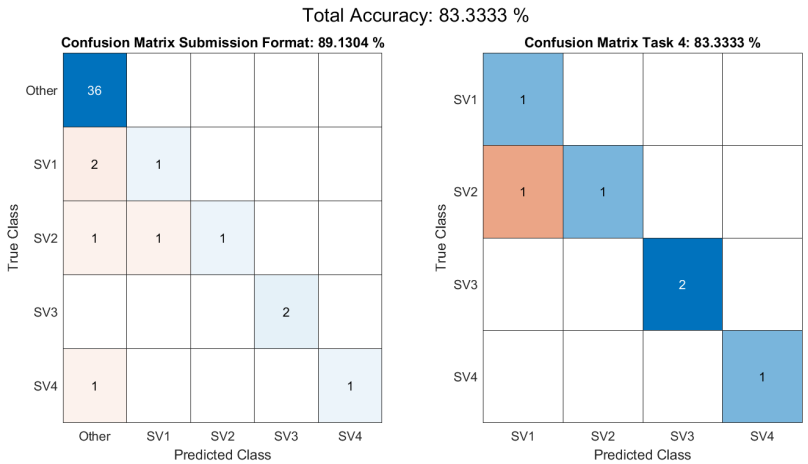


Figura 6.8: Matrice di confusione del task 4 sui dati di test

Le due matrici non hanno la stessa accuratezza in quanto, per effettuare la classificazione in questo task, la challenge richiede di considerare solo i dati che nel task 2 sono stati etichettati come *valve fault* e dalla matrice di confusione di tale task emerge che i casi erroneamente classificati come *normal* appartengono tutti alla classe delle valvole. Pertanto, l'accuratezza della matrice di confusione relativa ai dati stati utilizzati dal classificatore nel task 4 è del 83.333% poiché tutti i dati, tranne uno, vengono correttamente assegnati alla loro classe di appartenenza. Tuttavia, tale accuratezza è diversa da quella complessiva, poiché in precedenza quattro dati che avrebbero dovuto essere etichettati come *valve fault* sono stati erroneamente etichettati come *normal*.

Nella figura 6.9 sono riportati due grafici in uno spazio tridimensionale: a sinistra ogni punto è associato ad un colore differente sulla base della classe di appartenenza predetta, mentre a sinistra ogni punto è associato al colore verde se la classe predetta è quella corretta o al colore rosso se è la classe predetta non è quella corretta.
Si può notare che i dati sono abbastanza separabili tra di loro sulla base della valvola in cui è presente il guasto.

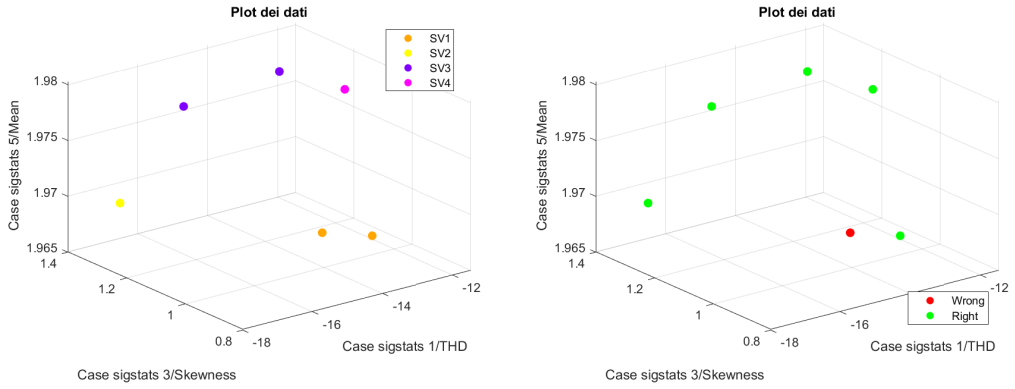


Figura 6.9: Plot tridimensionale dei dati secondo le tre feature con ranking maggiore nel task 4

6.5 Task 5

Come spiegato nel capitolo precedente, per questo task si è sviluppata sia la strada della classificazione che quella della regressione.
Si discuteranno i risultati ottenuti con entrambe le metodologie, ma si considera come definitivo quello ottenuto con la regressione.

6.5.1 Classificazione

Una volta importate le feature calcolate sull'intero dataset di training in Classification Learner, si è riaddestrato il modello Optimizable Ensamble con i parametri emersi

come migliori dalla fase di validation.

A seguito dell'addestramento, il modello è stato esportato per poter essere testato sui dati di test effettivi.
Per la valutazione della classificazione sono state utilizzate delle matrici di confusione riportate in figura 6.10.

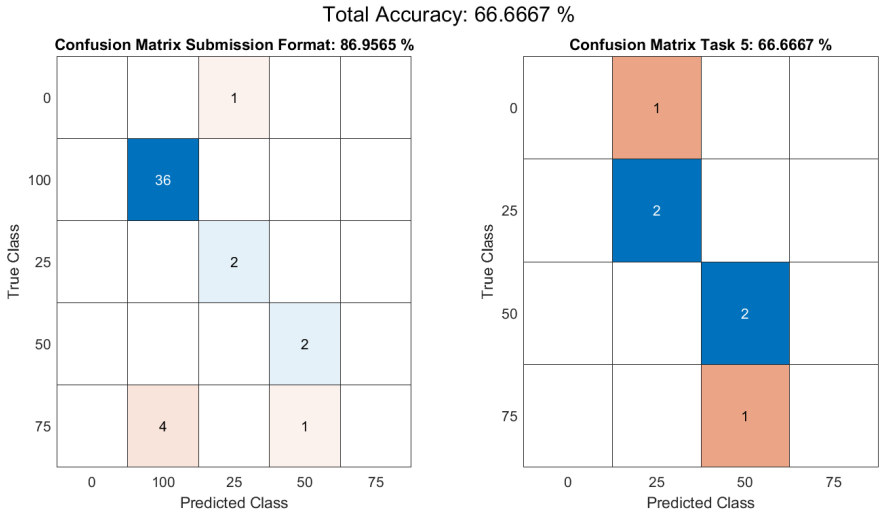


Figura 6.10: Matrice di confusione del task 5 sui dati di test

L'accuratezza calcolata considerando il formato di consegna è pari all'86.96%.
Se invece si considerano solo i dati effettivamente classificati dal modello, si ottiene un'accuratezza del 66.67%.
Va considerato che per confrontare i risultati della classificazione con i valori veri, siccome questi ultimi sono valori continui da 0 a 100, mentre i risultati della classificazione sono discreti, 0, 25, 50 e 100, i valori reali sono stati sostituiti con i valori discreti delle 4 classi.
In particolare, si è preso il valore continuo, si è calcolato modulo 25; se il risultato è maggiore di 12, si è sottratto a 25 il valore del modulo; questo rappresenta il valore da sommare al valore originale. Se il risultato è minore di 12, si prende il negativo del risultato del modulo e si somma questo numero al valore di partenza. Infine, si prende il minimo tra il risultato ottenuto e 75 in modo tale che il risultato non sia mai superiore a 75.

Anche in questo caso sono stati realizzati due grafici tridimensionali (6.12), in cui ciascun punto rappresenta un campione del dataset, mentre sugli assi sono riportate le prime tre feature ottenute dal feature ranking.

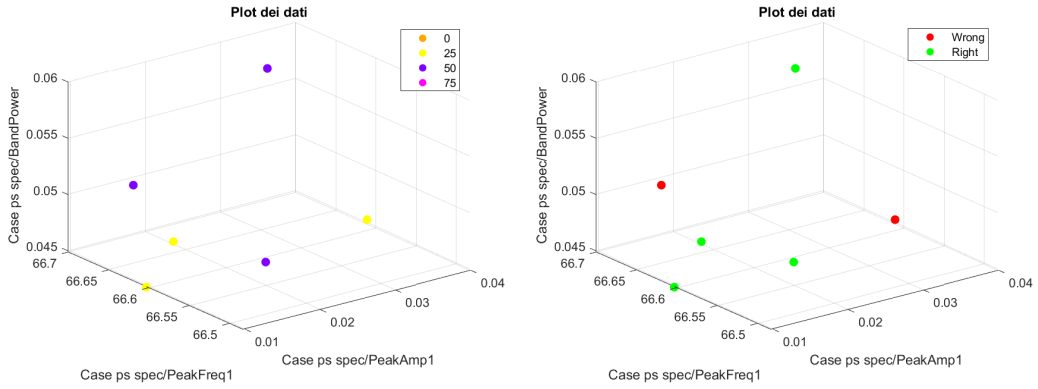


Figura 6.11: Plot tridimensionale dei dati secondo le tre feature con ranking maggiore nel task 5

6.5.2 Regressione

La combinazione di parametri rivelatasi migliore in fase di validazione è stata usata per addestrare il modello di regressione sulle feature estratte dai dati di training completi.

A questo punto si è usato il modello per predire il valore di apertura dei dati di test.

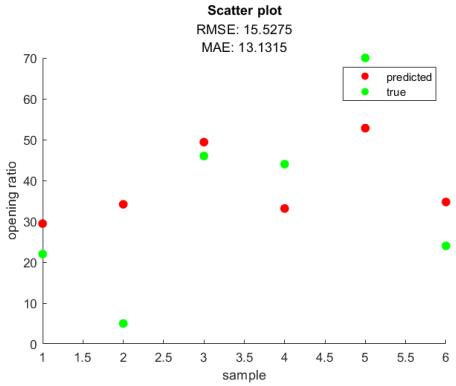


Figura 6.12: Scatter plot sulla regressione.

Con la regressione il valore di RMSE ottenuto è 15.5275, mentre il valore di MAE è 13.1315.
È interessante osservare che il sample 2, che nella classificazione era stato predetto come appartenente alla classe 25 ma la cui classe vera è 0 (considerato quanto spiegato nel capitolo precedente), è stato predetto con il valore di apertura 34.17, quando il valore vero era 5 (nella regressione).
Lo stesso vale per il sample 5, il cui valore vero è 70 ed è stato predetto con 52.8. Nella classificazione è stato predetto con 50.
La confusione del modello di regressione, quindi, è sovrapponibile alla confusione del modello di classificazione.

Capitolo 7

Valutazione

In conclusione, per poter misurare le prestazioni ottenute nei termini proposti dalla challenge si è implementato operativamente il calcolo della metrica proposta, secondo la quale:

1. per la corretta classificazione della condizione normale/anormale (task 1) $\rightarrow$ 10 punti;

2. limitatamente ai dati correttamente rilevati come anormali nel task 1, per la corretta classificazione della condizione bubble anomaly/valve fault/unknown $\rightarrow$ 10 punti;

3. limitatamente ai dati correttamente rilevati come contaminazione da bolla nel task 2, per la corretta identificazione della posizione della bolla $\rightarrow$ 10 punti;

4. limitatamente ai dati correttamente rilevati come fault su valvola nel task 2, per la corretta identificazione della posizione del fault (valvola guasta) $\rightarrow$ 10 punti;

5. limitatamente ai dati correttamente rilevati come fault su valvola nel task 2, per la corretta previsione del grado di apertura della valvola guasta $\rightarrow$ $max(-|verità - previsione| + 20, 0)$.

Per lo spacecraft-4, i punteggi sono raddoppiati, data la maggiore difficoltà.

Come tale punteggio sia stato implementativamente calcolato è specificato nel file *calculate_score* nella sottocartella di progetto *testing/evaluation*.

Il punteggio totale raggiungibile è pari a 1380, di cui:

- 690 punti sono ottenibili dalla corretta classificazione di tutti i dati del task 1;

- 390 punti sono ottenibili dalla corretta classificazione di tutti i dati del task 2;

- 150 punti sono ottenibili dalla corretta classificazione di tutti i dati del task 3;

- 150 punti sono ottenibili dalla corretta classificazione di tutti i dati del task 4;

- 300 punti sono ottenibili dalla corretta classificazione di tutti i dati del task 5.

Lo score raggiunto dal sistema di diagnosi implementato è pari a 1327.6, di cui:

- 630 su 690 punti sono stati ottenuti dalla corretta classificazione di 21/23 dati dello spacecaft-4 e 21/23 dati degli altri spacecraf nel task 1;
- 330 su 390 punti sono stati ottenuti dalla corretta classificazione di 22/26 dati nel task 2, considerando che i 4 errati in questo task sono di fatto un errore di propagazione dal task precedente;
- 150 su 150 punti sono stati ottenuti dalla corretta classificazione di tutti i dati nel task 3;
- 70 su 150 punti sono stati ottenuti dalla corretta classificazione di 5/10 dati nel task 4, considerando che 4 errori sui 5 compiuti in questo task sono di fatto un errore di propagazione dal primo task;
- 147.6 su 300 punti nel task 5.

Come si nota un peso notevole nella valutazione finale è stato dato dalle performance del modello di regressione del task 5, dove sono stati ottenuti meno della metà dei punti totali previsti.

Dal rapporto tra lo score ottenuto e lo score massimo si ottiene il punteggio finale di 79%.