Università Politecnica delle Marche
Facoltà di Ingegneria
Dipartimento di Ingegneria dell'Informazione

PHM Data Challange Asia Pacific 2023

Spacecraft Propulsion System Anomaly Detection

Elisa Pace
Micol Zazzarini
Andrea Fiorani

Indice

1 **6**
1.1 PHM Data Challange Asia Pacific 2023 . . . 6
1.1.1 Sistema di Propulsione Sperimentale . . . 6
1.1.2 Tipologie di anomalie . . . 7
1.1.3 Obbiettivo del progetto . . . 7
1.1.4 Dataset . . . 8
1.1.5 Metriche di Valutazione . . . 8
1.1.6 Formato della Consegna . . . 9
1.2 Strumenti utilizzati . . . 9
1.2.1 Matlab . . . 9
1.2.2 Diagnostic Feature Designer . . . 9
1.2.3 Classification Learner . . . 10
1.2.4 Regression Learner . . . 10
1.2.5 Experiment Manager . . . 10

2 Task 1: Determinazione dello Stato Normale o Anomalo dei Dati di Test **11**
2.1 Implementazione . . . 11
2.1.1 Preprocessing dei dati . . . 12
2.1.2 Generazione delle features . . . 13
2.1.3 Preparazione dei dati di training e Addestramento dei classificatori . . . 15
2.1.4 Hyperparameter Tuning . . . 18
2.1.5 Testing e Valutazione della performance . . . 20

3 Task 2: Classificazione dello Stato Anomalo dei Dati di Test **22**
3.1 Implementazione . . . 23
3.1.1 Preprocessing dei dati . . . 23
3.1.2 Generazione delle Features . . . 25
3.1.3 Addestramento dei modelli . . . 27
One Class Support Vector Machine (OC-SVM) . . . 27
Classificazione Binaria . . . 28
3.1.4 Hyperparameter Tuning . . . 29
3.1.5 Testing e Valutazione della performance . . . 31

4 Task 3: Determinazione della Posizione delle Bolle nel Sistema di Propulsione **34**
4.1 Implementazione . . . 35
4.1.1 Data Preprocessing . . . 35
4.1.2 Features Extraction . . . 36
4.1.3 Training e Hyperparameter Tuning . . . 36
4.1.4 Testing e Valutazione della performance . . . 38

5 Task 4: Determinazione delle valvole soggette a guasto nel Sistema di Propulsione **39**
5.1 Implementazione . . . 40
5.1.1 Data Preprocessing e Features Extraction . . . 40
5.1.2 Training e Hyperparameter Tuning . . . 40
5.1.3 Testing e Valutazione della performance . . . 42

6 Task 5: Predizione del Rapporto di Apertura delle Valvole Solenoidi **43**
6.1 Implementazione . . . 43
6.1.1 Data Preprocessing . . . 43
6.1.2 Estrazione delle Features . . . 44
6.1.3 Training . . . 44
6.1.4 Hyperparameter Tuning . . . 45
6.1.5 Testing e Valutazione della Performance . . . 46

7 Risultati Finali **48**
7.1 Pipeline e Punteggio Complessivo . . . 48
7.2 Esecuzione del Progetto . . . 49

A **50**

Elenco degli Snippet di Codice

2.1 Caricamento dei dati numerici dai file `.csv` nella directory specificata e salvataggio in una cell array. . . . 12
2.2 Creazione della tabella `trainingSet` con due colonne: `Case`, che contiene i dati numerici, e `Task1`, che rappresenta l'etichetta per il Task 1. . . . 12
2.3 Assegnazione delle etichette per il Task 1 . . . 12
2.4 Splitting . . . 17
3.1 Preprocessing of training data: Step 1 . . . 23
3.2 Preprocessing of training data: Step 2 . . . 24
3.3 Addestramento della One Class Support Vector Machine . . . 28
3.4 OC-SVM: Features Extraction and Prediction . . . 31
3.5 Sistema di voting utilizzando un treshold di una predizione su dieci . . . 31
3.6 RUS Boosted Trees: Features Extraction and Prediction . . . 31
4.1 Task 3: Training Data Preprocessing . . . 35
7.1 Task 1: Esecuzione della pipeline di test . . . 48
7.2 All predictions table . . . 49
7.3 Calcolo del punteggio totale . . . 49

Elenco delle tabelle

2.1 Parameter tuning ranges for the model Ensemble Boosted Trees . . . 18

3.1 Parameter tuning ranges for the model RUS Boosted Trees . . . 30
3.2 Metriche di valutazione per ciascuna classe. . . . 32

4.1 Parameter tuning ranges for the model Ensamble Boosted Trees . . . 38

5.1 Parameter tuning ranges for the model Subspace KNN . . . 41

6.1 Parameter tuning ranges for Gaussian Process Regression (GPR) model. . . . 46

A.1 task 5: configurazioni testate I . . . 50
A.2 task 5: configurazioni testate II . . . 50
A.3 task 5: configurazioni testate III . . . 50
A.4 task 4: configurazioni testate I . . . 51
A.5 task 4: configurazioni testate II . . . 51
A.6 task 1: configurazioni testate I . . . 52
A.7 task 1: configurazioni testate II . . . 52

Elenco delle figure

1.1 Sistema di Propulsione Sperimentale . . . 6
1.2 Profilo di pressione estratto dal modello . . . 7
1.3 Struttura del dataset . . . 8

2.2 Case P1: Signal Trace . . . 14
2.3 Case P1: Power Spectrum . . . 15
2.4 Feature Ranking . . . 16
2.5 Feature Table . . . 16
2.6 Risultati del testing del modello Ensemble Bagged Trees . . . 17
2.7 Hyperparameters tuning results . . . 19
2.8 Final Model: Confusion Matrix . . . 19
2.9 Final Test: Confusion Matrix . . . 20

3.1 Approccio a cascata . . . 23
3.3 Impostazione frame policy . . . 25
3.4 Range di frequenze e picchi significativi selezionato . . . 26
3.5 Ranking delle features: Estrazione delle prime 30 features sulla base della Monotonicity 26
3.6 Monitoraggio delle features per le labels 1 e 2 . . . 27
3.7 Confronto della performance dei modelli trainati . . . 29
3.9 Best RUS Boosted Trees: Confusion Matrix . . . 30
3.10 Testing Task 2: Confusion Matrix . . . 32

4.1 Punti in corrispondenza dei quali possono verificarsi anomalie da bolle . . . 34
4.2 Task 3: example of histogram from DFD . . . 36
4.4 Tuned Ensamble Boosted Trees: Confusion Matrix . . . 37
4.5 Testing Task 3: Confusion Matrix . . . 38

5.1 Valvole Solenoidi . . . 39
5.3 Testing Task 4: Confusion Matrix . . . 42

6.1 Response Plot for Predictions: Gaussian Process Regression (GPR) . . . 45
6.2 Actual vs Predicted (Validation) . . . 46
6.3 Testing Results: task 5 . . . 47

Capitolo 1

1.1 PHM Data Challange Asia Pacific 2023

Il seguente documento verge sullo svolgimento della *PHM (Prognostics and Health) Data Challange Asia Pacific 2023* ([(JA23]), proposta dalla *Japan Aerospace Exploration Agency (JAXA)*. La sfida nasce dalla scarsità dei dati di telemetria che possono essere acquisiti in orbita a causa delle limitazioni delle installazioni di sensori e dalla capacità di downlink. Per questo motivo, la JAXA ha sviluppato un simulatore numerico per prevedere la risposta dinamica di un sistema di propulsione di un veicolo spaziale con elevata precisione per generare un set di dati che copra le condizioni normali e tutti gli scenari di guasto previsti nelle apparecchiature reali. La competizione si pone l'obiettivo di migliorare la tecnologia PHM dei sistemi di propulsione per i veicoli spaziali di futura generazione, permettendogli di diagnosticare condizioni normali, anomalie dovute a bolle, guasti alle elettrovalvole e guasti anomali sconosciuti, andando a utilizzare i dati generati dal simulatore.

1.1.1 Sistema di Propulsione Sperimentale

In Figura 1.1 è raffigurata la struttura del **sistema di propulsione aerospaziale** di riferimento per la Challenge. Il fluido di lavoro corrisponde ad acqua pressurizzata a 2MPa e poi scaricato attraverso

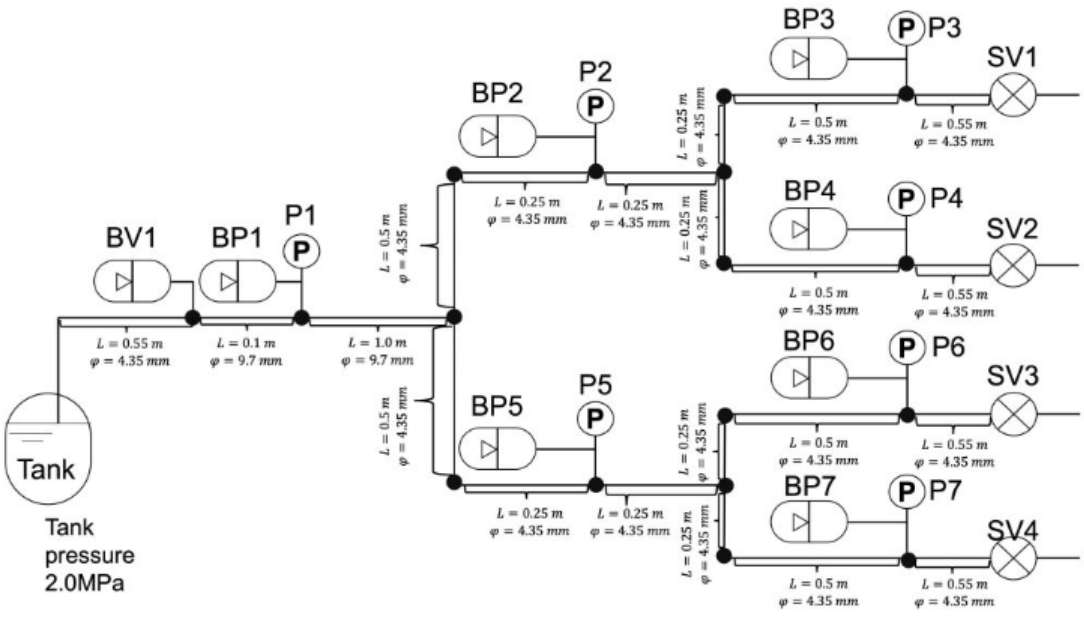


Figura 1.1: Sistema di Propulsione Sperimentale

quattro elettrovalvole (SV1-SV4) che simulano i propulsori. I **sensori di pressione (P1-P8)** raccolgono i dati delle serie temporali a una frequenza di campionamento di `1kHz` con un intervallo temporale da `0` a `1200ms`. Tramite l'apertura e chiusura delle elettrovalvole si possono osservare delle

fluttuazioni di pressione causate da improvvisi cambi di flusso nell'acqua, seguiti poi da modalità acustiche all'interno del sistema di propulsione. A partire da questo modello è possibile ricostruire un segnale che rappresenti un profilo tipico di pressione, raffigurato in Figura 1.2. L'elettrovalvola si apre

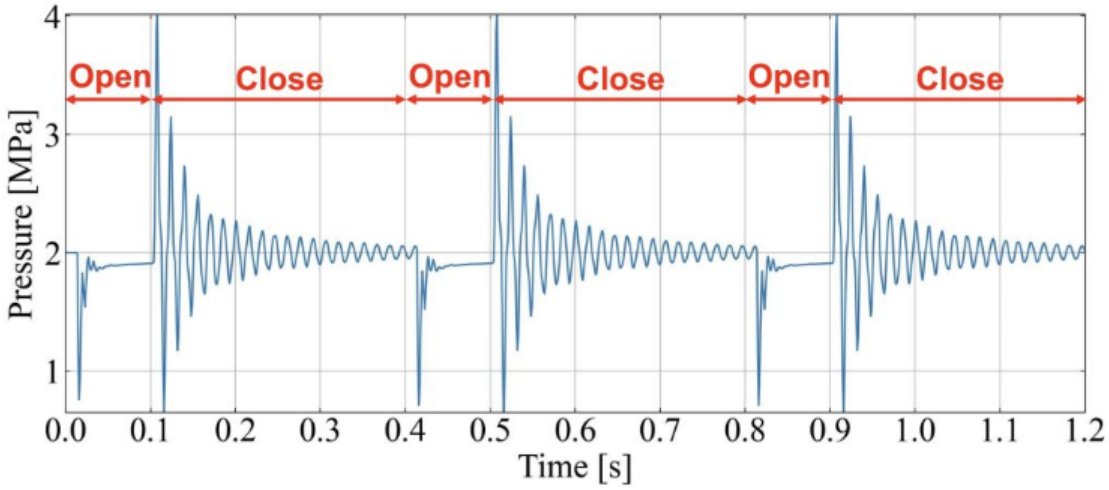


Figura 1.2: Profilo di pressione estratto dal modello

a `100ms` e si chiude a `300 ms`. Per tenere in considerazione le differenze individuali delle elettrovalvole che appaiono in apparecchiature reali, si considera un'incertezza di `1ms` nel movimento della valvola. Nonostante l'incertezza il tempo di apertura e chiusura rimane a `400ms`. Questa sequenza viene eseguita tre volte consecutive, ottenendo una misura totale di `1200ms`.

1.1.2 Tipologie di anomalie

Il progetto mira ad analizzare tre tipologie di anomalie:

- **Anomalie dovute a bolle**: Bolle d'aria possono occasionalmente apparire nei condotti durante il funzionamento reale di un veicolo spaziale. L'occorrenza di queste bolle varia la velocità del suono, causando leggeri disturbi nelle fluttuazioni di pressione. Per semplicità, la quantità di bolle presenti nel sistema di propulsione è considerata costante in tutti i casi. Si ambisce a rilevare la presenza di bolle e la loro posizione tra le otto possibili (BV1, BP1, ..., BP7).
- **Guasti alle elettrovalvole**: Si tratta di una delle principali modalità di guasto nei sistemi di propulsione dei veicoli spaziali. E' richiesta l'individuazione delle elettrovalvole guaste e il loro grado di apertura. Le elettrovalvole si aprono e chiudono rispettivamente con un rapporto di apertura del 100% e dello 0%. Nel caso di malfunzionamento, le elettrovalvole si aprono con un grado compreso tra lo 0% e il 100%, causando così una diminuzione del volume del fluido che le percorre.
- **Anomalia Sconosciuta**: Durante il funzionamento reale di un veicolo spaziale posso verificarsi delle anomalie sconosciute e completamente impreviste; per questo motivo si richiede di distinguere le anomalie sconosciute, senza confonderle con anomalie e guasti noti. Alcune anomalie sconosciute o guasti sono state inserite nel dataset di test.

1.1.3 Obbiettivo del progetto

L'obiettivo del progetto è la realizzazione di un **modello di diagnosi a cascata** che svolga i seguenti task:

1. Determinare i **valori normali o anormali** per tutto il dataset di test
2. Per i dati identificati come anormali, determinare se si trattano di malfunzionamenti dovuti a **contaminazione da bolle, guasti alle elettrovalvole o da guasti sconosciuti**.

3. Per i valori classificati come **malfunzionamento da bolle**, determinare la loro **posizione** tra una delle otto locazioni, BV1 e da BP1 a BP7.

4. Per i dati individuati come **errori di apertura alle elettrovalvole**, determinare **quale delle quattro elettrovalvole**(SV1,..., SV4) ha avuto problemi.

5. Per la valvola con guasto in fase di apertura, individuare la **percentuale di apertura** della stessa.

1.1.4 Dataset

Il dataset fornito per completare la challenge è strutturato come mostrato in Figura 1.3. Nel dataset sono presenti dati sia per l'addestramento che per il testing. In particolare, i dati generati appartengono a 4 spacecraft e questi vengono divisi in una parte per l'addestramento e la restante per il testing. Tutte le varie rilevazioni effettuate su uno spacecraft prende il nome di `Case`, seguito del relativo numero di rilevazione; tutto questo per poterle identificare in modo univoco. Un `Case` è relativo solo a un veicolo spaziale. I dati registrati dai sensori per ogni case, sono contenuti in un intervallo di `1200ms` e tengono nota di tre fasi di aperture e tre fasi di chiusura delle elettrovalvole. In ogni file csv, relativo ai case, la prima colonna si riferisce al tempo di rilevazione e le colonne rimanenti sono misurazioni di pressione prese nei punti di misuramento `P1-P7`, come mostrato in Figura 1.1.
Di seguito viene mostrato come sono suddivisi i dati per l'addestramento.

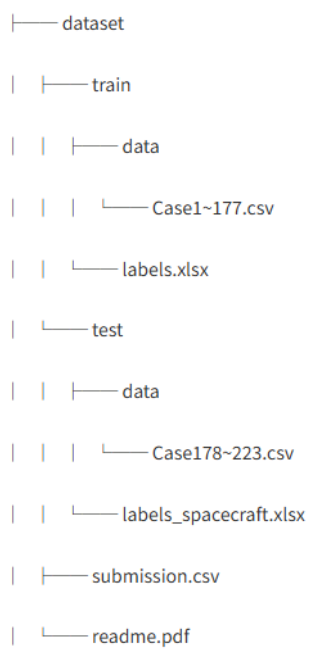


Figura 1.3: Struttura del dataset

- Numero di dati normali per il Veicolo Spaziale-1: 35
- Numero di dati normali per il Veicolo Spaziale-2: 35
- Numero di dati normali per il Veicolo Spaziale-3: 35
- Numero di dati anomali per il Veicolo Spaziale-1: 24
- Numero di dati normali per il Veicolo Spaziale-2: 24
- Numero di dati anomali per il Veicolo Spaziale-3: 24

I dati di training contengono i `Case` che vanno del numero 1 al numero 177 e all'interno della cartella di training, si trova il file `labels.xlsx`, dove sono state salvate tutte le informazione e specifiche (etichette) per ogni caso e per ogni task. I dati di testing sono invece organizzati nel seguente modo:

- Numero di dati per il Veicolo Spaziale-1: 23
- Numero di dati per il Veicolo Spaziale-4: 23

Contengono i case che vanno dal numero `178` al `223` e all'interno della cartella di testing è presente il file `labels_spacecraft.xlsx`, dove sono riportate esclusivamente i valori di riferimento per lo spacecraft e Case. Non sono presenti etichette.
Va fatto notare che, sia la presenza di differenze tra le valvole solenoidi (come spiegato in precedenza), sia la mancanza in fase di addestramento, di dati relativi allo Spacecraft N°4, potrebbe pregiudicare una riduzione delle prestazioni, soprattutto per le rilevazione dello Spacecraft N°4.

1.1.5 Metriche di Valutazione

Le metriche di valutazione del progetto sono le seguenti:

- Classificazione della condizione normale/anomala: **10 punti**.

- Per i dati correttamente rilevati come anomali, classificazione tra anomalia da contaminazione di bolle/guasto della valvola solenoide/guasto sconosciuto: **10 punti**.
- Per i dati correttamente identificati come contaminazione da bolle, identificazione della posizione della bolla: **10 punti**.
- Per i dati correttamente identificati come guasto della valvola solenoide, identificazione della valvola guasta: **10 punti**.
- Per la valvola solenoide correttamente identificata come guasta, previsione del rapporto di apertura: $\max(-|\textbf{valore reale} - \textbf{valore predetto}| + 20, 0)$.
- Per *Spacecraft-4*, i **punteggi sono raddoppiati**, considerando la difficoltà.

1.1.6 Formato della Consegna

I risultati devono essere inviati in un file CSV strutturato nel seguente modo:

- **Id**: L'ID del dato di test.
- **Task 1**: Inserire 0 per condizione normale, 1 per condizione anomala.
- **Task 2**: Per le condizioni anomale rilevate nel Task 1, inserire 1 per condizione sconosciuta, 2 per anomalia da bolle, 3 per guasto della valvola. Inserire 0 per tutti gli altri casi.
- **Task 3**: Per le anomalie da bolle rilevate nel Task 2, inserire: 1 per bolle in BP1, 2 per BP2, 3 per BP3, 4 per BP4, 5 per BP5, 6 per BP6, 7 per BP7, 8 per BV1. Inserire 0 per tutti gli altri casi.
- **Task 4**: Per il guasto della valvola rilevato nel Task 2, inserire: 1 per guasto in SV1, 2 per SV2, 3 per SV3, 4 per SV4. Inserire 0 per tutti gli altri casi.
- **Task 5**: Inserire il rapporto di apertura predetto (Y) con $0 \leq Y < 100$. Inserire 100 per tutti gli altri casi.

1.2 Strumenti utilizzati

Di seguito una breve descrizione introduttiva dei principali strumenti utilizzati per l'implementazione del progetto.

1.2.1 Matlab

MATLAB è una piattaforma di calcolo numerico ampiamente utilizzata per l'analisi dei dati, lo sviluppo di algoritmi e la modellazione di sistemi complessi. E' molto utilizzato in ambiti ingegneristici e scientifici, in quanto offre una vasta gamma di funzionalità, tra cui operazioni matematiche avanzate, visualizzazione grafica e supporto per il machine learning. Grazie alla sua flessibilità, MATLAB è stato utilizzato in questo progetto per integrare i dati raccolti, eseguire analisi preliminari e implementare le pipeline per l'addestramento e il testing dei modelli predittivi. La struttura modulare di MATLAB consente di combinare vari strumenti specializzati descritti in seguito, ottimizzando ogni fase del lavoro.

1.2.2 Diagnostic Feature Designer

Il Diagnostic Feature Designer è uno strumento chiave per l'analisi e l'estrazione delle features dai segnali raccolti dai sensori. Questo applicativo permette di analizzare dati complessi in diversi domini, offrendo un supporto fondamentale per identificare anomalie o guasti nei sistemi. Il processo di analisi prevede l'utilizzo di tecniche avanzate per calcolare parametri diagnostici e per mettere in atto un sistema di ranking delle feature, classificandole in base a differenti modi di ranking, anche in base al tipo di task per cui i dati verranno elaborati in seguito. Questo ha permesso di selezionare le caratteristiche più significative per i modelli di classificazione. Grazie all'integrazione con MATLAB, il tool consente anche di generare ed esportare funzioni personalizzate sotto forma di codice che automatizzano l'estrazione delle features.

1.2.3 Classification Learner

Il Classification Learner è uno strumento che, a partire dalle feature precedentemente estratte dal dataset, è stato utilizzato per sviluppare modelli di classificazione efficaci. Questo strumento offre un ambiente interattivo per testare diversi algoritmi di machine learning, con l'utilizzo di una GUI, permettendo di personalizzare l'addestramento variando i diversi parametri di configurazione. Consente, inoltre, di analizzare e confrontare i risultati utilizzando metriche standard come Accuratezza e Precisione, oltre a visualizzazioni come Matrici di Confusione e Curve di ROC, andando a individuare il modello che ha performato meglio per uno specifico task. Un'altra caratteristica utile del Classification Learner è la possibilità di riservare una porzione del dataset per il testing o di utilizzare tecniche di validazione incrociata, garantendo una valutazione accurata delle prestazioni del modello. Al termine dell'addestramento, i modelli possono essere esportati come funzioni MATLAB, rendendoli immediatamente applicabili ad altri dataset senza la necessità di ulteriori configurazioni manuali.

1.2.4 Regression Learner

Il Regression Learner è un tool progettato per affrontare problemi di regressione, permettendo di prevedere valori continui sulla base di dati esistenti. Questo strumento supporta un'ampia gamma di algoritmi, tra cui Regressione Lineare, Alberi Regressivi e Gaussian Processes. Durante il processo di addestramento, è possibile analizzare le prestazioni dei modelli attraverso metriche quali l'errore quadratico medio (RMSE) e il coefficiente di determinazione (R^2). Questi indicatori aiutano a identificare il modello più adatto per il problema in esame. Il Regression Learner fornisce anche strumenti per la visualizzazione dei risultati, permettendo di confrontare i valori predetti con quelli osservati e di identificare eventuali discrepanze. Come il Classification Learner, anche questo applicativo consente di esportare i modelli come funzioni MATLAB, semplificando l'integrazione in flussi di lavoro più complessi.

1.2.5 Experiment Manager

L'Experiment Manager è uno strumento avanzato che permette di gestire e confrontare in modo sistematico diversi esperimenti di machine learning. Grazie a questo strumento, è possibile organizzare e analizzare le performance di molteplici modelli, tenendo traccia delle configurazioni, delle metriche di valutazione e delle diverse strategie di pre-processing dei dati. Un aspetto chiave dell'Experiment Manager è la possibilità di automatizzare il processo di testing, consentendo di ottimizzare i parametri dei modelli e migliorare le loro prestazioni in maniera iterativa. Inoltre, offre funzionalità per la visualizzazione dei risultati, rendendo più semplice il confronto tra diversi esperimenti e l'individuazione della soluzione più efficace per il problema in esame. Grazie alla sua integrazione con gli altri strumenti di MATLAB, l'Experiment Manager rappresenta un elemento essenziale per il workflow di sviluppo e validazione dei modelli predittivi utilizzati nel progetto.

Capitolo 2

Task 1: Determinazione dello Stato Normale o Anomalo dei Dati di Test

Il Task 1 rappresenta il primo passo nella diagnosi dei sistemi di propulsione dei veicoli spaziali simulati, con l'obiettivo di **distinguere tra condizioni operative normali e anomale** utilizzando i dati di test generati dal simulatore sviluppato in collaborazione con l'Agenzia Spaziale Giapponese (JAXA). Questa fase iniziale costituisce una base fondamentale per l'identificazione e la classificazione delle specifiche anomalie che possono manifestarsi nei sistemi reali. L'obbiettivo principale è quello di costruire un sistema in grado di analizzare automaticamente i dati temporali e determinare se il comportamento osservato è coerente con condizioni normali o se presenta appunto delle anomalie. Si tratta di un compito reso complesso dalla presenza di variabilità individuale tra i veicoli spaziali considerati, dovuta a differenze intrinseche nei componenti, come i tempi di apertura e chiusura delle valvole solenoidi. Tuttavia, la corretta identificazione dello stato normale o anomalo è cruciale per garantire l'affidabilità del sistema diagnostico complessivo e per indirizzare con precisione le analisi successive volte alla classificazione dettagliata delle anomalie. Questo approccio è particolarmente rilevante nel contesto operativo reale, dove la rapida rilevazione e diagnosi delle anomalie possono prevenire guasti critici e garantire il funzionamento sicuro e continuo dei sistemi spaziali.

2.1 Implementazione

Nella pratica, per lo svolgimento di questo primo task di classificazione, abbiamo seguito un approccio strutturato articolato nelle seguenti fasi principali:

1. **Preprocessing** dei dati di training e di test.
2. **Generazione delle features nei dati di training** mediante l'uso del tool *Diagnostic Feature Designer (DFD)*
3. **Splittaggio** dei dati ottenuti in un dataset di training e di test per poter valutare la performance dei modelli addestrati.
4. **Training** di tutti i modelli disponibili sul dataset di training tramite il tool *Classification Learner (CL)*
5. **Testing** dei modelli addestrati sul dataset di test tramite il tool *Classification Learner*, e scelta del modello migliore sulla base della *Test Accuracy*.
6. **Hyperparameter tuning** del modello più performante tramite l'utilizzo del tool *Experiment Manager*, e salvataggio del modello pre-addestrato con la miglior configurazione di parametri, sulla base della *Validation Accuracy*.
7. **Generazione delle features nei dati di test** utilizzando una funzione appositamente esportata tramite il *Diagnostic Feature Designer*.
8. **Predizione sui dati di test** utilizzando il modello pre-addestrato.

9. **Valutazione della performance**.

2.1.1 Preprocessing dei dati

Nella fase di **Preprocessamento**, i dati del dataset di training sono stati organizzati e strutturati secondo le specifiche del problema, al fine di prepararli per l'addestramento dei modelli di classificazione. A partire dai dati a disposizione, cioè il file delle etichette (`labels.xlsx`) e la directory contenente i dati numerici da utilizzare per l'addestramento (`.csv`), abbiamo per prima cosa caricato le labels come una tabella utilizzando il comando `readtable`, che consente di importare i dati strutturati dal file `.xlsx`. In seguito, i nomi delle colonne interpretati come non validi da MATLAB (ad esempio, `Var1`, `Var2`, `Var3`) sono stati rinominati in modo significativo per riflettere i campi descrittivi (`Case`, `Spacecraft`, `Condition`), e i file `.csv` contenenti i dati numerici sono stati letti dalla directory specificata. Ogni file rappresenta un caso distinto e viene salvato all'interno di una *cell array* (`datacontainer`), in cui ciascun elemento corrisponde a una tabella di dati per un caso specifico, come mostrato nel codice 2.1.

Listing 2.1: Caricamento dei dati numerici dai file `.csv` nella directory specificata e salvataggio in una cell array.

```
cases = dir(TrainingDataset + "*.csv");
datacontainer = cell(numel(cases), 1);

for k = 1:numel(cases)
    casepath = TrainingDataset + string(cases(k).name);
    datacontainer{k} = readtable(casepath);
end
```

Come mostrato nel codice 2.2 È stata costruita dunque una tabella `trainingSet` (Figura 2.1a) che associa a ciascun case i dati numerici e le etichette corrispondenti. La tabella ha due colonne:

- **Case**, che contiene i dati di pressione per ciascun caso.
- **Task1**, che indica se il caso opera in condizioni normali (`0`) o anomale (`1`).

Listing 2.2: Creazione della tabella `trainingSet` con due colonne: `Case`, che contiene i dati numerici, e `Task1`, che rappresenta l'etichetta per il Task 1.

```
size = [numel(datacontainer), 2];
varTypes = {'cell', 'double'};
varNames = {'Case', 'Task1'};
trainingSet = table('Size', size, 'VariableTypes', varTypes, 'VariableNames',
    ↪ varNames);
trainingSet(:, 'Case') = datacontainer;
```

I valori delle labels sono stati determinati sulla base della colonna `Condition` nella tabella delle etichette (Codice 2.3). I casi in cui la condizione è `"Normal"` sono stati annotati con `0`, mentre tutti gli altri (guasti e anomalie sconosciute) sono stati annotati con `1`. Questa operazione è stata eseguita utilizzando il seguente codice:

Listing 2.3: Assegnazione delle etichette per il Task 1

```
trainingSet(:, 'Task1') = num2cell(~strcmp(labelsTable.Condition, "Normal"));
```

Il comando `strcmp` confronta i valori nella colonna `Condition` con la stringa `"Normal"`, restituendo un valore booleano. Il complemento logico (`~`) inverte il valore, assegnando `1` ai casi anomali e `0` ai casi normali.

Nella fase di pre-processamento del dataset di test, è stato necessario organizzare i dati in una struttura coerente con il dataset di training, in modo da garantire uniformità e compatibilità per la fase successiva di estrazione delle caratteristiche (**Features Extraction**). Questo approccio si è reso indispensabile poiché il processo di estrazione delle feature mediante il *Diagnostic Feature Designer (DFD)* richiede che i dataset di training e di test siano formattati in modo analogo. Dunque, tutti i file `.csv` presenti nella directory del dataset di test sono stati letti e salvati in un container (*cell array*)

chiamato `testDataContainer`. Ogni elemento del container rappresenta i dati di un singolo case. È stata poi creata una tabella denominata `testSet` (Figura 2.1b), con una struttura identica a quella utilizzata per il dataset di training, con due colonne `Case` e `Task1`. I dati di ciascun case presenti nel `testDataContainer` sono stati assegnati alla colonna `Case` della tabella. Inoltre, la colonna `Task1` è stata inizializzata con valori predefiniti pari a `0`, inizializzazione necessaria in quanto il dataset di test originale non include etichette reali, ma disponiamo di esse all'interno di un file `csv`.

1 Case	2 Task1
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	1
1201x8 table	1
1201x8 table	1
1201x8 table	1
1201x8 table	1
1201x8 table	1
1201x8 table	1
1201x8 table	1
1201x8 table	1
1201x8 table	1

(a) TrainingSet

1 Case	2 Task1
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0
1201x8 table	0

(b) TestSet

2.1.2 Generazione delle features

Per analizzare e generare features significative dai dati di segnale contenuti nel dataset di training, come anticipato, abbiamo utilizzato il tool *MATLAB Diagnostic Feature Designer (DFD)*.
Abbiamo iniziato importando il dataset di training, organizzato nella tabella `trainingSet`, in una nuova sessione del Diagnostic Feature Designer. Durante l'importazione, la colonna `Task1`, contenente le etichette 0 e 1 (indicanti rispettivamente condizioni normali e condizioni anomale), è stata impostata come variabile condizionale (`condition variable`). Questa configurazione ci ha permesso di effettuare successivamente un ranking supervisionato delle feature, ovvero un'analisi per determinare quali feature presentano la maggiore capacità di discriminare tra le condizioni normali e anomale.
La `frame policy` è stata impostata a `0.128` secondi, che rappresenta l'intervallo temporale (o finestra) entro cui sono state calcolate le feature per ciascun segnale. Questa scelta deriva dalla necessità di analizzare segmenti temporali brevi, in linea con la durata delle fluttuazioni osservate (evidenziate nei dati). Si tratta di un valore che rappresenta il giusto trade-off per un'estrazione ottimale delle features, una finestra troppo lunga avrebbe potuto infatti appiattire le variazioni caratteristiche, mentre una troppo corta avrebbe rischiato di generare rumore.
Per ciascun segnale, da P1 a P7, è stato generato il **Signal Trace**, una rappresentazione grafica che mostra il comportamento temporale del segnale per ogni caso di studio presente nel dataset di training (Figura 2.2). Esso consente di visualizzare le variazioni e i trend del segnale nel tempo, facilitando la comprensione del comportamento del sistema e l'identificazione di pattern utili. Abbiamo impostato il `Number of Curves` a `177`, poiché il dataset contiene 177 membri (ossia, 177 casi distinti), e ogni curva rappresenta il segnale associato a un caso. Questo ci ha permesso di analizzare i segnali per ciascun caso in modo completo.
Abbiamo proceduto dunque con l'estrazione di tutte le possibili features nel dominio temporale (*Time-Domain Features*), che forniscono informazioni statistiche e dinamiche sul comportamento del segnale. Le feature calcolate includono:

- **Clearance Factor**: Indica la sensibilità del segnale rispetto a picchi elevati.
- **Crest Factor**: Misura il rapporto tra il valore di picco e l'RMS del segnale.
- **Impulse Factor**: Valuta l'entità dei picchi rispetto al valore medio.
- **Kurtosis**: Misura l'appiattimento o la "punta" della distribuzione del segnale.
- **Mean**: Rappresenta il valore medio del segnale.
- **Peak Value**: Identifica il valore massimo del segnale.
- **RMS (Root Mean Square)**: Fornisce una misura dell'energia del segnale.
- **SINAD (Signal-to-Noise and Distortion Ratio)**: Indica il rapporto tra segnale utile e rumore + distorsione.
- **SNR (Signal-to-Noise Ratio)**: Misura la qualità del segnale rispetto al rumore.
- **Shape Factor**: Rappresenta la forma del segnale.
- **Skewness**: Misura la simmetria della distribuzione del segnale.
- **Standard Deviation (Std)**: Fornisce una misura della variabilità del segnale.
- **THD (Total Harmonic Distortion)**: Indica la distorsione armonica totale nel segnale.

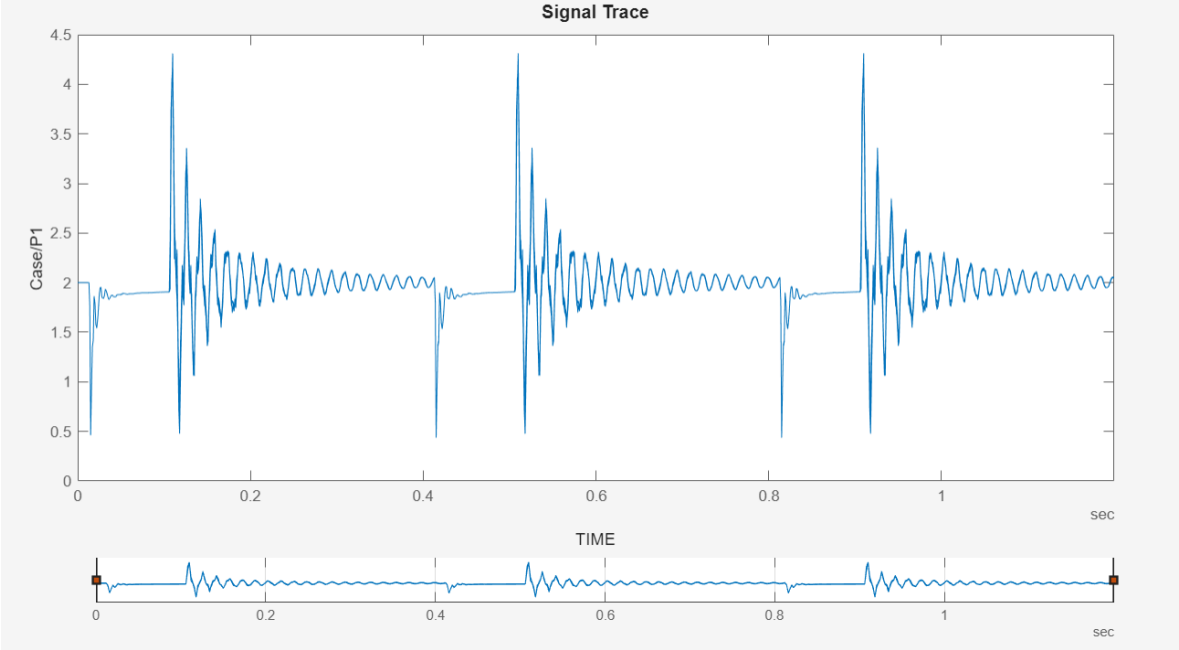


Figura 2.2: Case P1: Signal Trace

Successivamente, abbiamo generato un **modello autoregressivo** per i segnali da P1 a P7 (Figura 2.3). Questo modello è stato costruito per analizzare la dinamica temporale del segnale, modellandolo come una combinazione lineare dei suoi valori passati. Il numero di curve è stato impostato a 177, coerentemente con il numero di membri del dataset.
Abbiamo poi estratto le features nel dominio della frequenza (*Spectral Features*), che analizzano il contenuto frequenziale dei segnali. Durante questa fase, la `Frequency Band` è stata impostata tra `5Hz`e `300Hz` per focalizzarci sulle frequenze rilevanti per il sistema e ridurre il rumore ad alta frequenza. Il `Numero di Picchi` è stato impostato a `2`, considerando che i segnali analizzati presentano al massimo due picchi significativi nella loro distribuzione spettrale. Le features calcolate in questo caso includono:

- **Peak Amplitude**: L'ampiezza massima rilevata nello spettro.

- **Peak Frequency**: La frequenza associata al picco massimo.
- **Band Power**: La potenza complessiva del segnale all'interno della banda specificata.

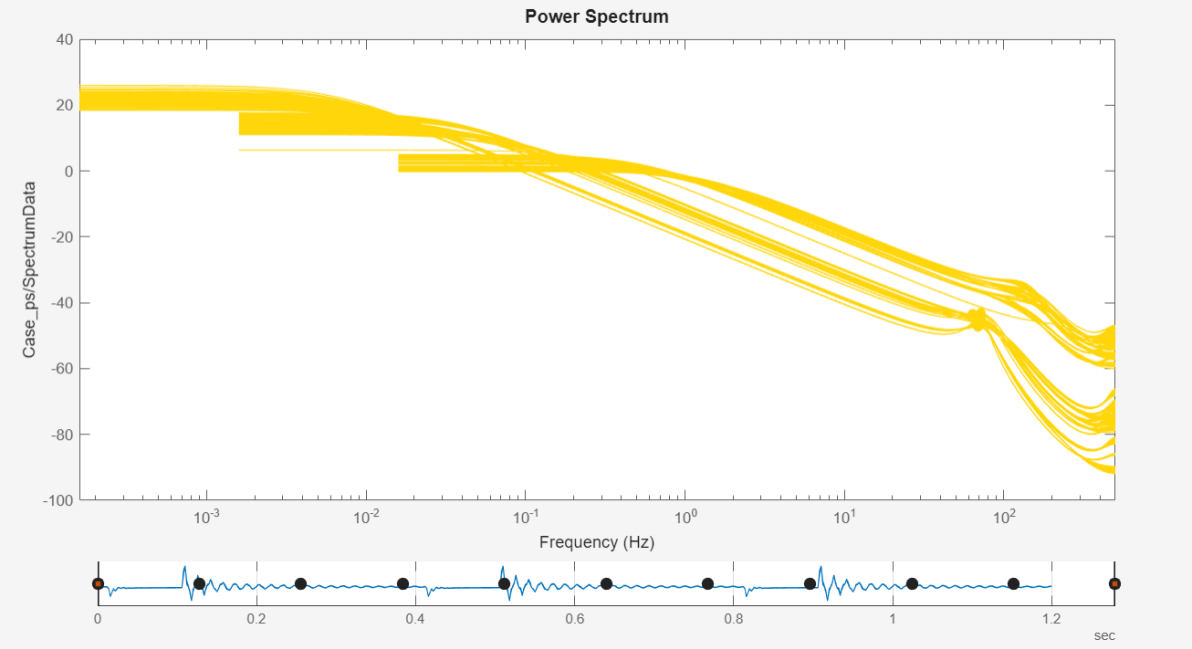


Figura 2.3: Case P1: Power Spectrum

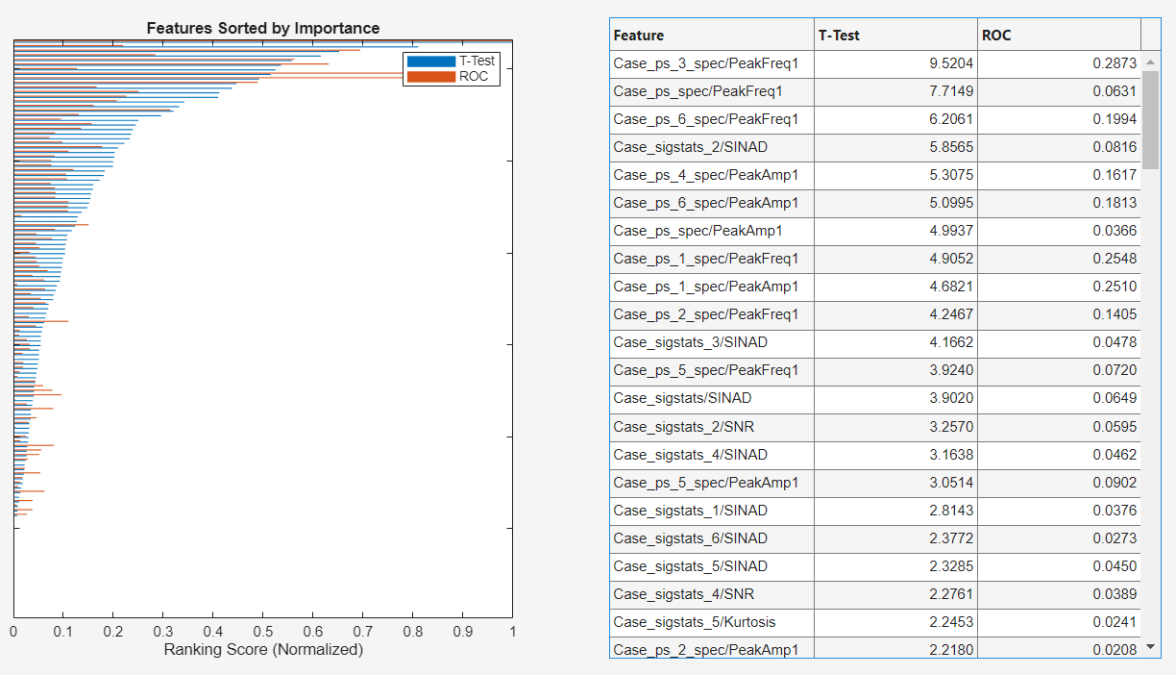
Per valutare l'importanza delle feature estratte, abbiamo poi effettuato un **ranking supervisionato** (Figura 2.4) utilizzando due metriche:

- **T-test**: Un test statistico che valuta la capacità di una feature di distinguere tra condizioni normali e anomale. Le feature con differenze significative tra i due gruppi ottengono un punteggio più alto.
- **ROC (Receiver Operating Characteristic)**: Una metrica che valuta la capacità di discriminazione di una feature in termini di sensibilità e specificità, misurata tramite l'area sotto la curva ROC (AUC). Features con AUC prossima a 1 sono particolarmente efficaci.

Questo ranking ci ha permesso di identificare le features più rilevanti e utili per classificare le condizioni operative del sistema. Tuttavia, in seguito a diversi tentativi, come esposto in Tabella A.6 e Tabella A.7 in Appendice, il miglior risultato relativo a questo primo task è stato ottenuto estraendo tutte le features generate. Tutto ciò ha permesso di costruire un set di dati arricchito e strutturato, contenente informazioni significative sia nel dominio del tempo che della frequenza (Figura 2.5).

2.1.3 Preparazione dei dati di training e Addestramento dei classificatori

Dopo l'estrazione delle features tramite il *Diagnostic Feature Designer*, è stato necessario eseguire uno **split del dataset** ottenuto in un **training set** e un **test set**. Questo passaggio è stato fondamentale per poter addestrare i classificatori sul training set e valutare la loro accuratezza, dei classificatori, sul test set, garantendo così una valutazione imparziale delle loro prestazioni su dati non visti durante l'addestramento, migliorando così la generalizzazione del modello.
Lo split è stato effettuato in modo stratificato, preservando le proporzioni delle diverse classi (condizioni Normale, Fault e Anomaly) nel training e nel test set. Questo approccio è particolarmente importante per evitare che il modello sia sbilanciato verso una delle classi e per garantire che la distribuzione delle condizioni nel test set sia rappresentativa del problema reale.
Abbiamo impostato il 20% del dataset come test set e l'80% come training set, una scelta comune nella progettazione di sistemi di apprendimento automatico. Questo split fornisce un buon compromesso tra



Feature	T-Test	ROC
Case_ps_3_spec/PeakFreq1	9.5204	0.2873
Case_ps_spec/PeakFreq1	7.7149	0.0631
Case_ps_6_spec/PeakFreq1	6.2061	0.1994
Case_sigstats_2/SINAD	5.8565	0.0816
Case_ps_4_spec/PeakAmp1	5.3075	0.1617
Case_ps_6_spec/PeakAmp1	5.0995	0.1813
Case_ps_spec/PeakAmp1	4.9937	0.0366
Case_ps_1_spec/PeakFreq1	4.9052	0.2548
Case_ps_1_spec/PeakAmp1	4.6821	0.2510
Case_ps_2_spec/PeakFreq1	4.2467	0.1405
Case_sigstats_3/SINAD	4.1662	0.0478
Case_ps_5_spec/PeakFreq1	3.9240	0.0720
Case_sigstats/SINAD	3.9020	0.0649
Case_sigstats_2/SNR	3.2570	0.0595
Case_sigstats_4/SINAD	3.1638	0.0462
Case_ps_5_spec/PeakAmp1	3.0514	0.0902
Case_sigstats_1/SINAD	2.8143	0.0376
Case_sigstats_6/SINAD	2.3772	0.0273
Case_sigstats_5/SINAD	2.3285	0.0450
Case_sigstats_4/SNR	2.2761	0.0389
Case_sigstats_5/Kurtosis	2.2453	0.0241
Case_ps_2_spec/PeakAmp1	2.2180	0.0208

Figura 2.4: Feature Ranking

1 EnsembleID_	2 Task1	3 FRM_1/TimeStart	4 FRM_1/TimeEnd	5 FRM_1/Case_sigstats/Clearanc	6 FRM_1/Case_sigstats/CrestFact	7 FRM_1/Case_sigstats/ImpulseF	FRM_1/(
"Member 1"	0	0	0.1280	2.2087	2.1152	2.1767	
"Member 1"	0	0.1280	0.2560	1.3880	1.3707	1.3820	
"Member 1"	0	0.2560	0.3840	1.0683	1.0673	1.0679	
"Member 1"	0	0.3840	0.5120	2.1971	2.1353	2.1776	
"Member 1"	0	0.5120	0.6400	1.6269	1.5839	1.6107	
"Member 1"	0	0.6400	0.7680	1.0701	1.0688	1.0696	
"Member 1"	0	0.7680	0.8960	1.0982	1.0887	1.0942	
"Member 1"	0	0.8960	1.0240	2.0857	2.0009	2.0566	
"Member 1"	0	1.0240	1.1520	1.0946	1.0930	1.0941	
"Member 1"	0	1.1520	1.2800	1.0380	1.0376	1.0379	
"Member 2"	0	0	0.1280	2.2229	2.1288	2.1908	
"Member 2"	0	0.1280	0.2560	1.3878	1.3705	1.3818	
"Member 2"	0	0.2560	0.3840	1.0684	1.0674	1.0680	
"Member 2"	0	0.3840	0.5120	2.2118	2.1495	2.1921	
"Member 2"	0	0.5120	0.6400	1.6199	1.5772	1.6039	

Figura 2.5: Feature Table

l'avere un training set sufficientemente ampio per addestrare i modelli e un test set abbastanza grande per una valutazione affidabile. Poiché il dataset è strutturato in segmenti di segnale derivanti dall'applicazione della frame policy di 0.128 secondi, ogni caso è composto da 10 segmenti (`num_window = 10`). Durante lo split, ogni segmento è stato trattato come un'unità, assicurando che i segmenti derivanti dallo stesso caso fossero distribuiti coerentemente tra il training set e il test set. Abbiamo sfruttato la colonna `Task1`, contenente le etichette delle condizioni operative (0 per Normale, 1 per Fault e Anomaly), per effettuare uno split stratificato. La stratificazione ha garantito che nel test set fossero presenti proporzioni simili di ciascuna classe, rispetto alla distribuzione nel dataset complessivo. Lo scopo dello split è stato dunque quello di garantire una valutazione robusta e oggettiva dei classificatori. Il test set, costituito da segmenti mai utilizzati durante l'addestramento, ha consentito di misurare l'efficacia dei modelli in termini di accuratezza e di generalizzazione. In questo modo, è stato possibile confrontare i diversi classificatori addestrati nel *Classification Learner* e scegliere il modello migliore in base alla performance sul test set, assicurandosi che la scelta non fosse influenzata da overfitting sul training set. Il codice utilizzato per eseguire lo splitting è esposto nel Listing 2.4.

Listing 2.4: Splitting

```
if ismember('Task1', features_table.Properties.VariableNames)
    % stratified splitting 80-20
    % 1 : 105 normal, 106 : 153 fault, 154 : 177 anomaly
    normal_test_perc = int32(105*test_perc/100);
    fault_test_perc = int32((153-105)*test_perc/100);
    anomaly_test_perc = int32((177-153)*test_perc/100);

    anomaly_test = training_table(153*num_window+1:(anomaly_test_perc+153)*num_window
    ↪ ,:);
    training_table(153*num_window+1:(anomaly_test_perc+153)*num_window,:) = [];

    fault_test = training_table(105*num_window+1:(fault_test_perc+105)*num_window,:);
    training_table(105*num_window+1:(fault_test_perc+105)*num_window,:) = [];

    normal_test = training_table(1:normal_test_perc*num_window,:);
    training_table(1:normal_test_perc*num_window,:) = [];

    test_table = [normal_test; fault_test; anomaly_test];

end
```

Dopo aver addestrato e valutato i diversi classificatori utilizzando il *Classification Learner*, i risultati sono stati ordinati in base all'accuratezza sul test set per identificare il modello con le migliori prestazioni. Come mostrato in Figura 2.6, il modello migliore selezionato è risultato essere l'*Ensemble Bagged Trees*, che ha raggiunto una test accuracy pari all'83%. Questo modello utilizza un approccio di *ensemble learning*, combinando molteplici decision tree deboli per ottenere una predizione complessiva più robusta e accurata. Il *bagging (Bootstrap Aggregating)* è una tecnica che migliora la stabilità e la precisione dei modelli di apprendimento, riducendo la varianza attraverso l'addestramento di ciascun albero su un sottoinsieme casuale dei dati.
In dettaglio, dalla matrice di confusione emerge che il modello ha classificato correttamente il 97.5% dei casi normali, dimostrando un'elevata capacità di identificare correttamente i dati appartenenti a questa classe. Ha inoltre identificato correttamente il 66.5% dei casi di fault, mostrando una buona sensibilità verso questa classe. Tuttavia, ha confuso il 33.5% dei casi di fault, classificandoli erroneamente come normali. Questo valore riflette una maggiore difficoltà nell'identificare correttamente tutte le istanze della classe 1. Dunque, nel complesso, il modello ha dimostrato un buon compromesso tra accuratezza complessiva e capacità di generalizzazione sui dati di test, seppur con un margine di miglioramento nella riduzione dei falsi negativi per la classe 1.

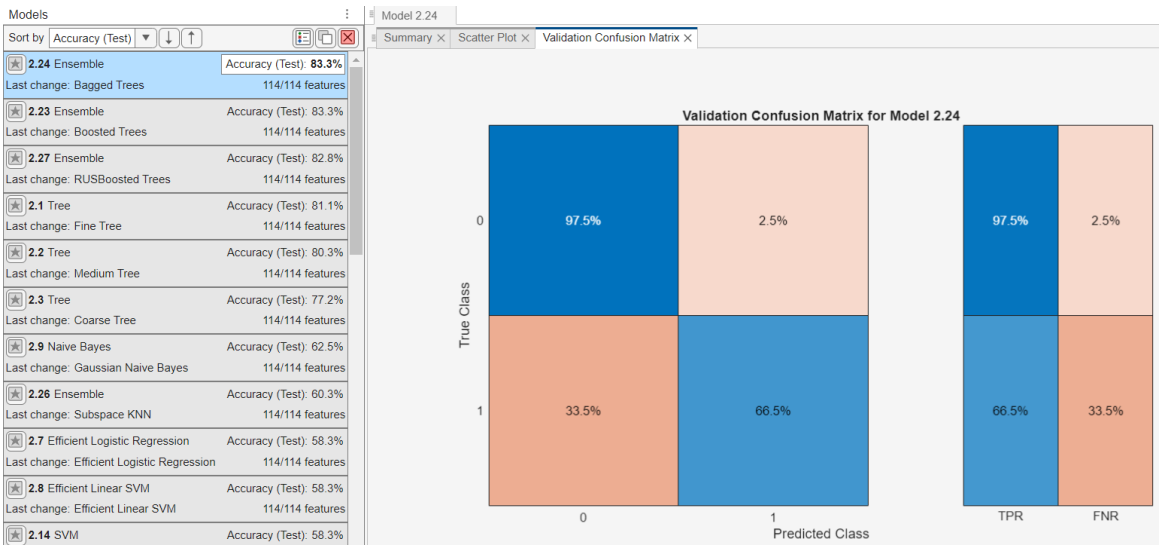


Figura 2.6: Risultati del testing del modello Ensemble Bagged Trees

2.1.4 Hyperparameter Tuning

Per il modello *Ensemble Bagged Trees*, abbiamo dunque utilizzato il tool *Experiment Manager* per eseguire il **Tuning degli Iperparametri**, con l'obiettivo di massimizzare l'accuratezza di validazione del modello. L'*Experiment Manager* è un potente strumento di MATLAB progettato per automatizzare il processo di ottimizzazione dei modelli di Machine Learning e Deep Learning. Consente di eseguire esperimenti su vasta scala, testando combinazioni di iperparametri, valutandone le prestazioni e selezionando automaticamente la configurazione ottimale in base a un criterio specifico (nel nostro caso, massimizzare l'accuratezza di validazione).
La strategia adottata per l'ottimizzazione è stata la *Bayesian Optimization*, che è particolarmente efficace per problemi complessi e costosi in termini computazionali. A differenza di metodi più semplici come la griglia (*Grid Search*) o la ricerca casuale (*Random Search*), la Bayesian Optimization utilizza un approccio probabilistico per esplorare lo spazio degli iperparametri in modo più intelligente ed efficiente. Inizia con una fase di esplorazione iniziale, valutando alcune combinazioni casuali di iperparametri. Costruisce un modello probabilistico dello spazio degli iperparametri, stimando la relazione tra i valori degli iperparametri e la metrica di valutazione (accuratezza). Utilizza una funzione di acquisizione per scegliere nuove combinazioni di iperparametri che bilanciano l'esplorazione (provare nuovi valori) e lo sfruttamento (raffinare i valori promettenti). Ripete poi il processo fino al raggiungimento del numero massimo di tentativi (40 nel nostro caso), o di una soglia di accuratezza desiderata. In questo modo riduce il numero di esperimenti necessari, poiché esplora solo le regioni promettenti dello spazio degli iperparametri.

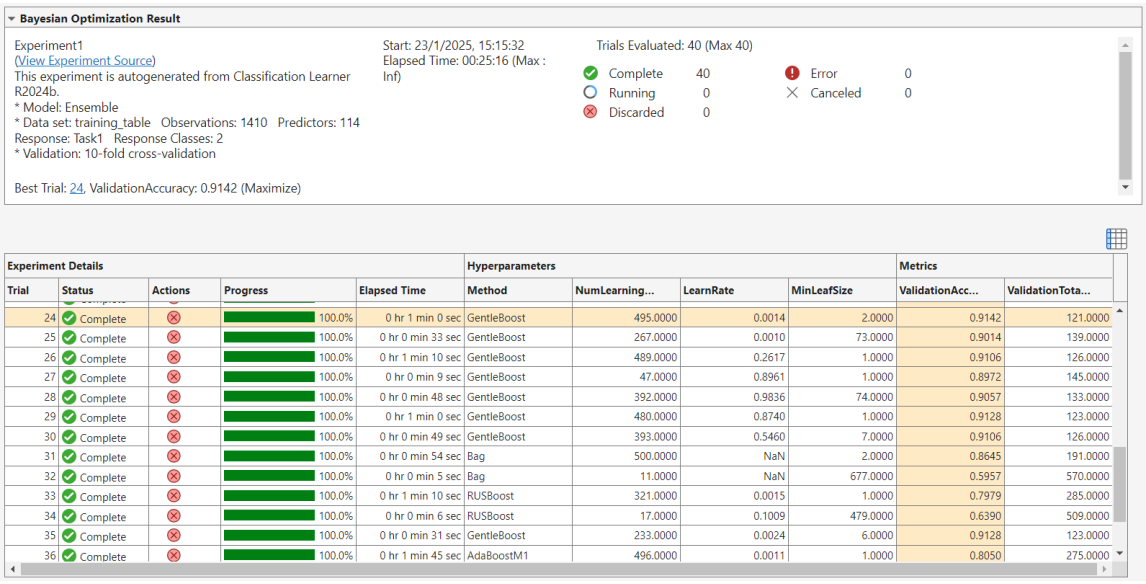
Parameter	**Range**
Method	[”Bag”, ”GentleBoost”, ”LogitBoost”, ”AdaBoostM1”, ”RUSBoost”]
NumLearningCycles	[10,500]
LearnRate	[0.001,1]
MinLeafSize	[1,705]

Tabella 2.1: Parameter tuning ranges for the model Ensemble Boosted Trees

Come mostrato in Tabella 2.1, l'esperimento di tuning è stato impostato per ottimizzare i seguenti iperparametri del modello Ensemble Bagged Trees:

- **Method**: La strategia di ensemble utilizzata per combinare i decision tree. Le opzioni valutate includono:
  - **Bag (Bagging)**: Combina alberi deboli addestrati su sottoinsiemi casuali di dati, con un focus sulla riduzione della varianza.
  - **GentleBoost**: Variante del boosting che aggiorna i pesi dei dati con un impatto minore, rendendolo meno sensibile ai dati rumorosi.
  - **LogitBoost**: Utilizza una funzione logistica per migliorare le predizioni, spesso utile in problemi di classificazione binaria.
  - **AdaBoostM1**: Combina alberi deboli correggendo iterativamente gli errori commessi dai modelli precedenti.
  - **RUS Boost**: Random Undersampling Boosting, adatto per problemi con classi sbilanciate.
- **NumLearningCycles**: Numero di alberi utilizzati nell'ensemble. Questo parametro controlla la complessità del modello e può variare nell'intervallo [10, 500].
- **LearnRate**: Tasso di apprendimento per aggiornare i pesi delle istanze nell'ensemble. Un valore troppo alto può portare a un training instabile, mentre un valore troppo basso può rallentare il processo. L'intervallo considerato è [0.001, 1].
- **MinLeafSize**: Numero minimo di campioni richiesti in una foglia dell'albero decisionale. Questo parametro controlla il grado di potatura degli alberi ed è testato nell'intervallo [1, 705].

Durante ogni iterazione è stato costruito un modello del comportamento del classificatore in funzione degli iperparametri, e la funzione di acquisizione ha scelto la successiva combinazione promettente da testare. Ad ogni configurazione è stata associata una metrica di accuratezza di validazione calcolata sul set di validazione. Dopo i 40 tentativi, Come mostrato in Figura 2.7, Experiment Manager ha restituito la configurazione ottimale degli iperparametri che massimizzava l'accuratezza di validazione del modello Ensemble Bagged Trees.



Experiment Details					Hyperparameters				Metrics	
Trial	Status	Actions	Progress	Elapsed Time	Method	NumLearning...	LearnRate	MinLeafSize	ValidationAcc...	ValidationTota...
24	Complete		100.0%	0 hr 1 min 0 sec	GentleBoost	495.0000	0.0014	2.0000	0.9142	121.0000
25	Complete		100.0%	0 hr 0 min 33 sec	GentleBoost	267.0000	0.0010	73.0000	0.9014	139.0000
26	Complete		100.0%	0 hr 1 min 10 sec	GentleBoost	489.0000	0.2617	1.0000	0.9106	126.0000
27	Complete		100.0%	0 hr 0 min 9 sec	GentleBoost	47.0000	0.8961	1.0000	0.8972	145.0000
28	Complete		100.0%	0 hr 0 min 48 sec	GentleBoost	392.0000	0.9836	74.0000	0.9057	133.0000
29	Complete		100.0%	0 hr 1 min 0 sec	GentleBoost	480.0000	0.8740	1.0000	0.9128	123.0000
30	Complete		100.0%	0 hr 0 min 49 sec	GentleBoost	393.0000	0.5460	7.0000	0.9106	126.0000
31	Complete		100.0%	0 hr 0 min 54 sec	Bag	500.0000	NaN	2.0000	0.8645	191.0000
32	Complete		100.0%	0 hr 0 min 5 sec	Bag	11.0000	NaN	677.0000	0.5957	570.0000
33	Complete		100.0%	0 hr 1 min 10 sec	RUSBoost	321.0000	0.0015	1.0000	0.7979	285.0000
34	Complete		100.0%	0 hr 0 min 6 sec	RUSBoost	17.0000	0.1009	479.0000	0.6390	509.0000
35	Complete		100.0%	0 hr 0 min 31 sec	GentleBoost	233.0000	0.0024	6.0000	0.9128	123.0000
36	Complete		100.0%	0 hr 1 min 45 sec	AdaBoostM1	496.0000	0.0011	1.0000	0.8050	275.0000

Figura 2.7: Hyperparameters tuning results

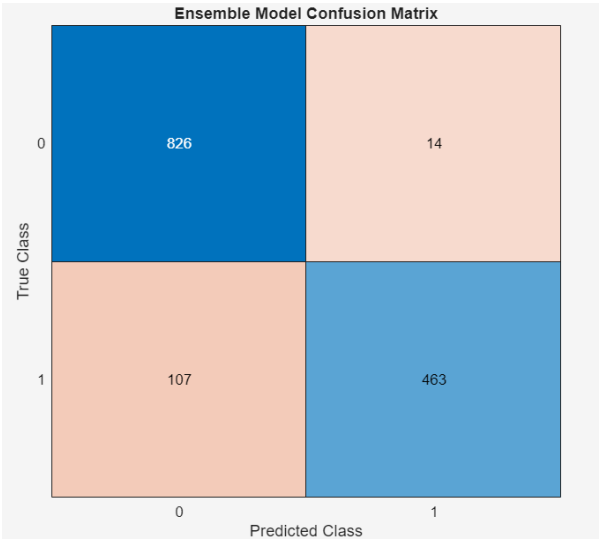


Figura 2.8: Final Model: Confusion Matrix

Analizzando i risultati in Figura 2.8, la nuova matrice di confusione mostra un significativo miglioramento delle prestazioni rispetto alla precedente, evidenziando un modello molto più accurato nel distinguere le due classi. Il modello ha identificato correttamente 826 esempi della classe 0, commettendo soltanto 14 errori (*false positives*). Questo indica una precisione altissima nel prevedere che un esempio appartiene alla classe normale. Allo stesso tempo, il modello ha identificato 463 esempi della classe 1, ma ha commesso 107 errori, classificando questi ultimi come negativi (*false negatives*). Nel complesso, questa matrice riflette un modello equilibrato, che è altamente affidabile nel distinguere la classe 0 e comunque solido nel riconoscere la classe 1. Questo potrebbe essere un modello particolarmente adatto in situazioni dove è fondamentale ridurre al minimo i falsi positivi, ma con un impatto tollerabile per i falsi negativi, come nel nostro caso. Infatti, dai risultati della matrice, otteniamo le seguenti metriche:

- **Accuracy**: misura la proporzione di previsioni corrette rispetto al totale, ed è definita come:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{1289}{1410} \approx 0.9142\,(91.42\%)$$

- **Precision**: Rappresenta la proporzione di previsioni corrette della classe positiva rispetto a tutte le previsioni di quella classe:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{463}{477} \approx 0.9706\,(97.06\%)$$

- **Recall**: misura la capacità del modello di identificare correttamente tutti gli esempi positivi:

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{463}{570} \approx 0.8123\,(81.23\%)$$

- **F1-Score**: Media armonica tra precisione e recall, ed è definito come:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \cdot \frac{0.7886}{1.7829} \approx 0.884\,(88.4\%)$$

- **Specificità**: Misura la capacità del modello di identificare correttamente tutti gli esempi negativi:

$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{826}{840} \approx 0.9833\,(98.33\%)$$

2.1.5 Testing e Valutazione della performance

Abbiamo infine realizzato una **pipeline per valutare le prestazioni** del modello finale pre-addestrato ottenuto dalla fase precedente, sul dataset di test originale. La pipeline inizia caricando i dati preprocessati di addestramento e test tramite gli script esterni (`trainingData.m` e `testData.m`). Le caratteristiche diagnostiche vengono estratte dal set di test tramite la funzione `diagnosticFeaturesExtraction_task1` generata tramite il *Diagnostic Feature Designer*, che genera una tabella contenente le features per la classificazione. Il modello pre-addestrato effettua quindi le predizioni sui dati test. Questi risultati preliminari, indicati come `predictedLabelsArray`, rappresentano le classificazioni finestra per finestra. Dato che ogni campione è suddiviso in 10 finestre (corrispondenti a un frame temporale di 1200ms suddiviso in intervalli di 0.128 ms ciascuno), le predizioni devono essere aggregate per fornire un'unica decisione per campione. Questo passaggio introduce i due sistemi di *voting* testati:

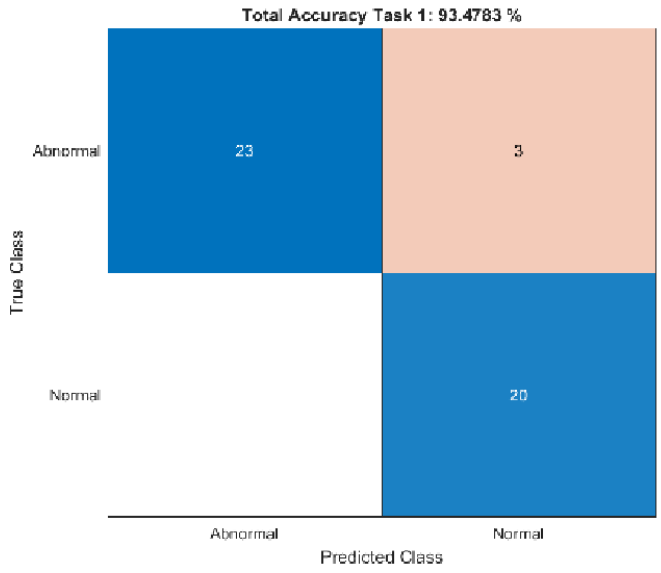


Figura 2.9: Final Test: Confusion Matrix

- **Sistema di aggregazione basato sulla moda (Voting System 1)**: questo primo metodo di aggregazione calcola la moda (*mode*) delle predizioni relative alle 10 finestre associate a ciascun campione. Questo approccio assegna al campione la classe che appare con maggiore frequenza. Le predizioni finali (`predictedPerSample`) vengono quindi salvate in un file e contate per ciascuna classe (`normale` e `anormale`). Questo sistema è semplice e tende a favorire la classe predominante all'interno delle finestre, riducendo l'impatto di singoli errori. Tuttavia, può risultare in prestazioni subottimali quando il numero di finestre con la classe minore si avvicina a quello della classe predominante.

- **Sistema di aggregazione basato su soglia (Voting System 2)**: il secondo metodo di aggregazione introduce una soglia. Per ciascun campione, il metodo calcola quante finestre (su 10) sono classificate come `anormale` (classe 1). Se il numero di finestre di tipo `anormale` supera la soglia del 70%, l'intero campione viene classificato come `anormale`, altrimenti viene assegnata la classe `normale`. La soglia consente di controllare la sensibilità e la specificità del modello, favorendo una decisione più robusta nei casi in cui la moda risulti ambigua.

Le prestazioni del modello sono state poi valutate per ciascun sistema di voting, calcolando l'accuratezza e visualizzando la corrispondente matrice di confusione.
Il sistema di voting basato sulla soglia ha dimostrato prestazioni migliori rispetto a quello basato sulla moda. La flessibilità introdotta dalla soglia permette una gestione più efficace delle situazioni ambigue, migliorando l'accuratezza complessiva del modello. La matrice di confusione ottenuta per il sistema basato sulla soglia è riportata in Figura 2.9.
Sono stati classificati correttamente 23 campioni su un totale di 26 appartenenti alla classe reale `Abnormal`. Solo 3 campioni sono stati erroneamente classificati come normali, rappresentando i falsi negativi (FN). Per la classe `Normal`, sono stati classificati correttamente tutti i 20 campioni su 40 appartenenti alla classe reale `Normal`.
Il sistema di voting basato sulla soglia si è rivelato dunque particolarmente efficace, con un'accuratezza complessiva del **93.48%**. La precisione è risultata perfetta per la classe `Abnormal` (**100%**), poiché non sono stati commessi falsi positivi. Tuttavia, la Recall per la classe `Abnormal` è stato leggermente inferiore (88.46%) a causa di 3 campioni erroneamente classificati come normali. Per la classe `Normal`, i risultati sono perfetti, con tutti i campioni correttamente identificati. Questi risultati indicano che il sistema è altamente affidabile e presenta ottime capacità di discriminazione tra le classi.

Capitolo 3

Task 2: Classificazione dello Stato Anomalo dei Dati di Test

Nell'ambito del miglioramento della tecnologia *PHM (Prognostics and Health Management)* per i sistemi di propulsione spaziale, la corretta identificazione delle anomalie rappresenta un passaggio cruciale per garantire il corretto funzionamento e la sicurezza delle missioni spaziali. Il secondo task del progetto si concentra dunque sulla **classificazione delle anomalie** rilevate in precedenza nei dati di test, distinguendo tra tre possibili categorie:

- **Anomalia dovuta alla contaminazione da bolle**: Durante le operazioni di un sistema di propulsione spaziale, è possibile che si verifichi la presenza di bolle d'aria nelle tubature del circuito idraulico. Questo fenomeno altera la velocità del suono nel fluido, causando variazioni nelle fluttuazioni di pressione. Nel contesto della competizione, le bolle possono essere presenti in otto diverse posizioni del sistema di propulsione: `BV1` e `BP1-BP7`. La loro identificazione è essenziale per garantire un funzionamento ottimale del sistema, evitando instabilità nel flusso del propellente e riducendo il rischio di guasti più gravi.
- **Guasto delle valvole solenoidi**: Le valvole solenoidi (`SV1-SV4`) regolano il flusso del fluido nel sistema di propulsione aprendo e chiudendo con un rapporto di apertura standard del 100% (completamente aperte) o dello 0% (completamente chiuse). Tuttavia, un guasto può causare un'apertura parziale della valvola, con un rapporto di apertura compreso tra 0% e 100%, determinando una riduzione del volume di fluido che passa attraverso la valvola. L'identificazione del guasto e della specifica valvola coinvolta, insieme alla stima del suo grado di apertura, è cruciale per la manutenzione e la previsione di eventuali malfunzionamenti.
- **Anomalia sconosciuta**: Nel funzionamento reale di un sistema di propulsione, possono verificarsi anomalie impreviste non contemplate nei modelli di guasto conosciuti. Il sistema di diagnostica deve essere in grado di riconoscere queste anomalie senza confonderle con i problemi noti (bolle o guasti alle valvole solenoidi). La capacità di individuare un'anomalia sconosciuta consente di migliorare l'affidabilità del sistema di diagnostica e fornisce dati preziosi per l'analisi futura e l'aggiornamento degli algoritmi di previsione dei guasti.

Il nostro obiettivo in questo caso è quindi sviluppare un sistema di diagnosi automatizzato che sia in grado di riconoscere correttamente la natura dell'anomalia, fornendo un contributo significativo alla manutenzione predittiva dei sistemi di propulsione spaziale. Un'identificazione tempestiva e precisa consente di evitare malfunzionamenti critici che potrebbero compromettere la missione, ottimizzare la manutenzione preventiva, riducendo i costi e migliorando l'efficienza operativa, migliorare i modelli predittivi, consentendo un'evoluzione continua della tecnologia PHM, rendere i sistemi di propulsione più resilienti, minimizzando i rischi associati a eventi imprevisti.

3.1 Implementazione

Nella pratica, abbiamo affrontato la difficoltà posta da questo task attraverso un approccio a cascata (Figura 3.1). Infatti, uno dei principali problemi affrontati nello sviluppo del modello è stata l'assenza di esempi di anomalie sconosciute nei dati di training. Per superare questa limitazione, abbiamo suddiviso dunque il task in due fasi distinte:

1. **Identificazione di anomalie note o sconosciute** : Abbiamo utilizzato una *One-Class Support Vector Machine (OC-SVM)*, addestrata esclusivamente sulle anomalie note a disposizione nel nostro dataset, per distinguere se un campione appartenesse a una categoria già vista o se rappresentasse un'anomalia sconosciuta.

2. **Classificazione delle anomalie note**: Una volta identificate le anomalie note, abbiamo utilizzato un classificatore binario per determinare se si trattasse di una contaminazione da bolle o di un guasto delle valvole solenoidi.

Questo approccio gerarchico ci ha permesso di separare il problema in due sottocompiti gestibili, migliorando la robustezza della diagnosi rispetto alle anomalie sconosciute.

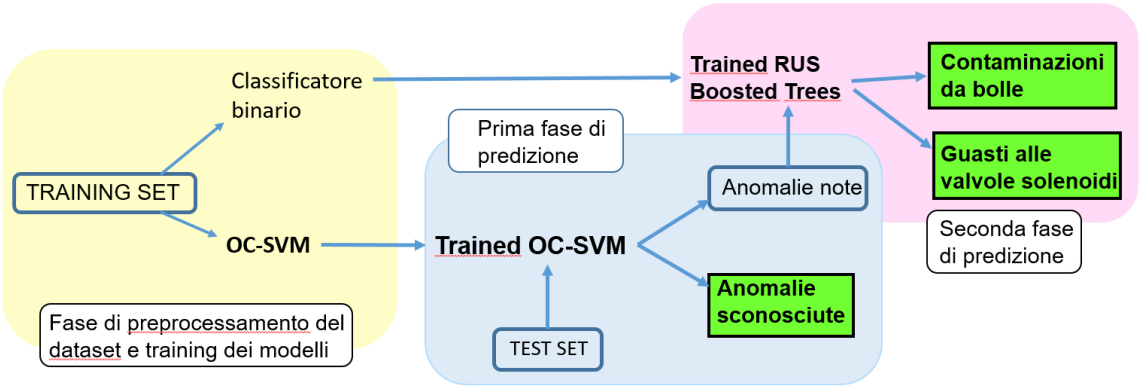


Figura 3.1: Approccio a cascata

3.1.1 Preprocessing dei dati

Come nel caso precedente, abbiamo sottoposto i dati originali ad un'operazione di **Preprocessing** in preparazione all'addestramento, ricavandone due nuovi training sets. Il primo (Figura 3.2a), per il classificatore binario, è un dataset composto da sole anomalie note, labellate rispettivamente come contaminazione da bolle e guasto alle valvole solenoidi. Il secondo (Figura 3.2c) è il dataset di addestramento della *One Class Support Vector Machine (OC-SVM)*, che presenta lo stesso contenuto del precedente, seppur con una unica etichetta rappresentante un'anomalia nota senza distinzione di genere.

Listing 3.1: Preprocessing of training data: Step 1

```
1 for i = 1:numel(data)
2     trainingData{i, 1} = data{i};
3
4     if labelsTable{i, "Condition"} == "Normal"
5         trainingData{i, 2} = 0;
6     elseif labelsTable{i,"SV1"}~=100 || labelsTable{i,"SV2"}~=100 || labelsTable{i,"
      ↪ SV3"}~=100 || labelsTable{i,"SV4"}~=100
7         trainingData{i, 2} = 1; % solenoid valve fault
8     elseif labelsTable{i,"BP1"}=="Yes" || labelsTable{i,"BP2"}=="Yes" || labelsTable{
      ↪ i,"BP3"}=="Yes" || labelsTable{i,"BP4"}=="Yes" || labelsTable{i,"BP5"}=="Yes"
      ↪ || labelsTable{i,"BP6"}=="Yes" || labelsTable{i,"BP7"}=="Yes" || labelsTable{i
      ↪ ,"BV1"}=="Yes"
9         trainingData{i, 2} = 2; % bubbles
```

```
    else
        trainingData{i, 2} = 3; % unknown anomalies (0)
    end
end
```

Come mostrato nel Listing 3.1, abbiamo in prima istanza assegnato le seguenti etichette ai campioni del dataset di training originale:

- `0`: Stato Normale
- `1`: Guasto alle valvole solenoide
- `2`: Contaminazione da bolle
- `3`: Anomalia sconosciuta (campioni assenti nel dataset)

In seguito, abbiamo filtrato solo i campioni appartenenti alle classi 1 e 2, mantenendo le etichette, per ottenere il dataset di training per il task di classificazione binaria delle anomalie note. Abbiamo infine convertito tutte le etichette a 0 (anomalia nota) per l'addestramento della OC-SVM (Listing 3.2).

Listing 3.2: Preprocessing of training data: Step 2

```
% Converts trainingData in a table with two columns: "Case" e "Label"
Case = trainingData(:,1);
Labels = cell2mat(trainingData(:,2));

% Filter only anomalies ( 1 e 2)
anomalyIndices = Labels ~= 0; % finds raws with label != 0
Case = Case(anomalyIndices);
Labels = Labels(anomalyIndices);

% Training data for binary classification
trainingTable_binary = table(Case, Labels, 'VariableNames', {'Case', 'Label'});

% Converts all labels to 0 = known anomaly (for one class SVM)
Labels_SVM = Labels;
Labels_SVM(Labels_SVM == 1 | Labels_SVM == 2) = 0; % known anomalies  0
Labels_SVM(Labels_SVM == 3) = 1; % unknown anomalies  1

% Training data for OC-SVM
trainingTable_SVM = table(Case, Labels_SVM, 'VariableNames', {'Case', 'Label'});
```

Nel processo di Preprocessamento del dataset di test, l'obiettivo principale era renderlo coerente con il dataset di training e filtrare solo i campioni anomali per il task di classificazione delle anomalie. Per facilitare la gestione e il tracciamento dei dati, abbiamo assegnato un ID univoco a ciascun caso, creando un array `caseIDs` che contiene numeri progressivi corrispondenti ai file. Questo ID è stato fondamentale nelle fasi successive per mantenere un riferimento chiaro tra i dati originali e quelli filtrati. Infatti, durante i filtraggi eseguiti nel corso del task, il dataset di test viene più volte filtrato e ridotto. Grazie all'ID, è stato poi possibile riaggregare correttamente tutte le predizioni alla fine del processo. Questo preprocessamento garantisce che il dataset di test sia compatibile con quello di training, contenendo solo anomalie e avendo una struttura omogenea. Inoltre, assicura che le predizioni future possano essere facilmente confrontate e ricollocate nei dati originali, grazie all'utilizzo degli ID univoci. Successivamente, abbiamo costruito la tabella `testSet`, che contiene tre colonne (`ID, Case, Label`), rispettivamente l'identificatore numerico univoco del campione, i dati grezzi di ogni campione di test, un valore inizialmente impostato a 0 per tutti i campioni (placeholder), poiché il dataset di test non contiene le etichette dei dati. Poiché il task riguarda la classificazione di anomalie, sono stati eliminati dal dataset di test tutti i campioni precedentemente classificati come normali nel primo task. Solo i campioni anomali vengono mantenuti, creando la nuova tabella `filteredTestSet` (Figura 3.2b). Infine, è stata creata una tabella `predictionsTable` per raccogliere le future predizioni del modello. Questa tabella è stata inizialmente popolata con valori `NaN`, per poi andare ad assegnare 0 ai campioni normali (quelli esclusi in precedenza), e infine, in maniera graduale, il resto dei valori nel momento dell'ottenimento delle predizioni.

trainingTable_binary

72x2 table

	1 Case	2 Label
46	1201x8 table	1
47	1201x8 table	1
48	1201x8 table	1
49	1201x8 table	2
50	1201x8 table	2
51	1201x8 table	2
52	1201x8 table	2
53	1201x8 table	2
54	1201x8 table	2
55	1201x8 table	2
56	1201x8 table	2
57	1201x8 table	2
58	1201x8 table	2
59	1201x8 table	2
60	1201x8 table	2
61	1201x8 table	2
62	1201x8 table	2
63	1201x8 table	2
64	1201x8 table	2

(a) Training table for Binary Classification

filteredTestSet

23x3 table

	1 ID	2 Case	3 Label
1	1	1201x8 table	0
2	2	1201x8 table	0
3	4	1201x8 table	0
4	7	1201x8 table	0
5	9	1201x8 table	0
6	11	1201x8 table	0
7	13	1201x8 table	0
8	16	1201x8 table	0
9	19	1201x8 table	0
10	20	1201x8 table	0
11	23	1201x8 table	0
12	25	1201x8 table	0
13	27	1201x8 table	0
14	28	1201x8 table	0
15	30	1201x8 table	0
16	32	1201x8 table	0
17	35	1201x8 table	0
18	37	1201x8 table	0
19	39	1201x8 table	0

(b) Filtered Test Set composed by anomalies only

trainingTable_SVM

72x2 table

	1 Case	2 Label
1	1201x8 table	0
2	1201x8 table	0
3	1201x8 table	0
4	1201x8 table	0
5	1201x8 table	0
6	1201x8 table	0
7	1201x8 table	0
8	1201x8 table	0
9	1201x8 table	0
10	1201x8 table	0
11	1201x8 table	0
12	1201x8 table	0
13	1201x8 table	0
14	1201x8 table	0
15	1201x8 table	0
16	1201x8 table	0
17	1201x8 table	0
18	1201x8 table	0
19	1201x8 table	0

(c) Training table for One Class SVM

3.1.2 Generazione delle Features

Una volta preparati i dati di training, siamo passati alla fase di **estrazione delle features** per l'addestramento dei modelli di Machine Learning, sia per la OC-SVM che per il classificatore binario. Anche in questo caso abbiamo fatto uso del tool *Diagnostic Feature Designer (DFD)*, e il processo di estrazione, sia delle *Time Domain Features* che delle *Spectral Features*, è stato pressoché analogo a quello del primo task, pur regolando in maniera specifica una serie di parametri.
Come nel precedente task, abbiamo impostato la `frame policy` a un valore di `0.128` (Figura 3.3), e abbiamo generato la **Signal Trace** per tutti i casi da P1 a P7, ponendo un numero di curve pari al numero di campioni del dataset (72). Abbiamo poi estratto le features nel dominio del tempo, come in precedenza (**Clearance Factor**, **Crest Factor**, **Impulse Factor**, **Kurtosis**, **Mean**, **Peak Value**, **RMS**, **SINAD**, **SNR**, **Shape Factor**, **Skewness**, **Standard Deviation**, **THD**). Siamo poi passati alla generazione del **modello autoregressivo** per i segnali da P1 a P7, e anche in questo caso, il numero di curve è stato impostato a 72, mentre l'ordine del modello è stato impostato a 10. A partire dal **Power Spectrum generato**, abbiamo poi estratto le features nel dominio della frequenza (**Peak Amplitude**, **Peak Frequency**, **Band Power**), impostando la **Frequency Band** tra **5Hz** e **400Hz** con un numero di picchi significativi pari a 2 (Figura 3.4).

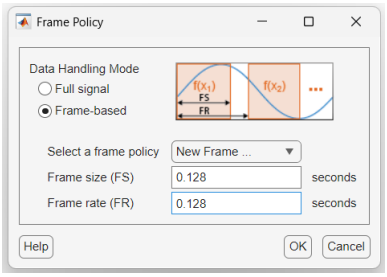


Figura 3.3: Impostazione frame policy

Per quanto riguarda il ranking delle features, abbiamo utilizzato due approcci diversi rispettivamente per il caso della OC-SVM e del classificatore binario. In dettaglio, nel caso della *One Class Support Vector Machine*, il dataset di training era composto da campioni di anomalie note etichettate tutte con **0**, senza distinzione di tipologia. Non essendo per questo motivo possibile effettuare un ranking supervisionato delle features, abbiamo impostato la variabile `Label` come indipendente, e abbiamo utilizzato perciò la **Monotonicity**, estraendo le migliori 30 features (Figura 3.5).
La Monotonicity è un criterio di ranking che misura quanto una feature varia in modo coerente rispetto alla severità dell'anomalia o al degrado di un sistema. In particolare, una feature è considerata monotona se la sua tendenza segue un andamento crescente o decrescente al variare della condizione del sistema. Questo implica che, man mano che il sistema si allontana dallo stato normale, il valore della

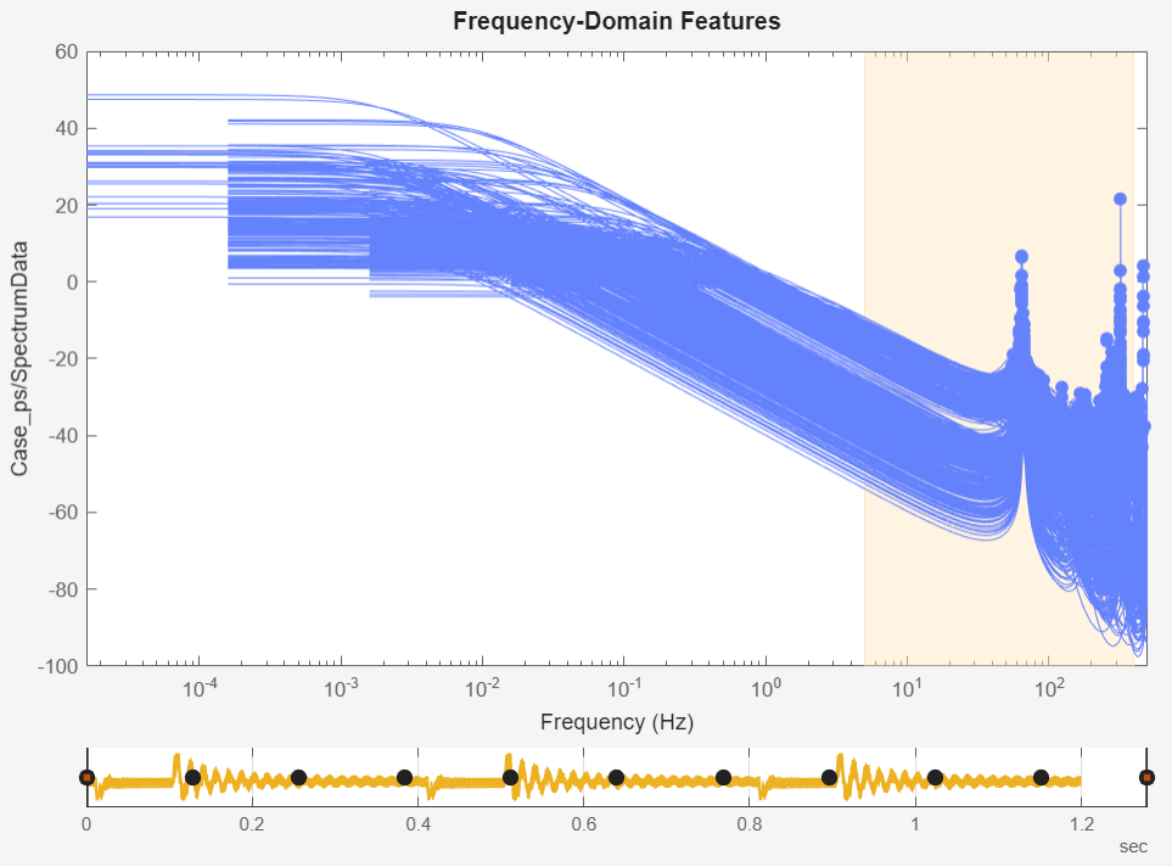
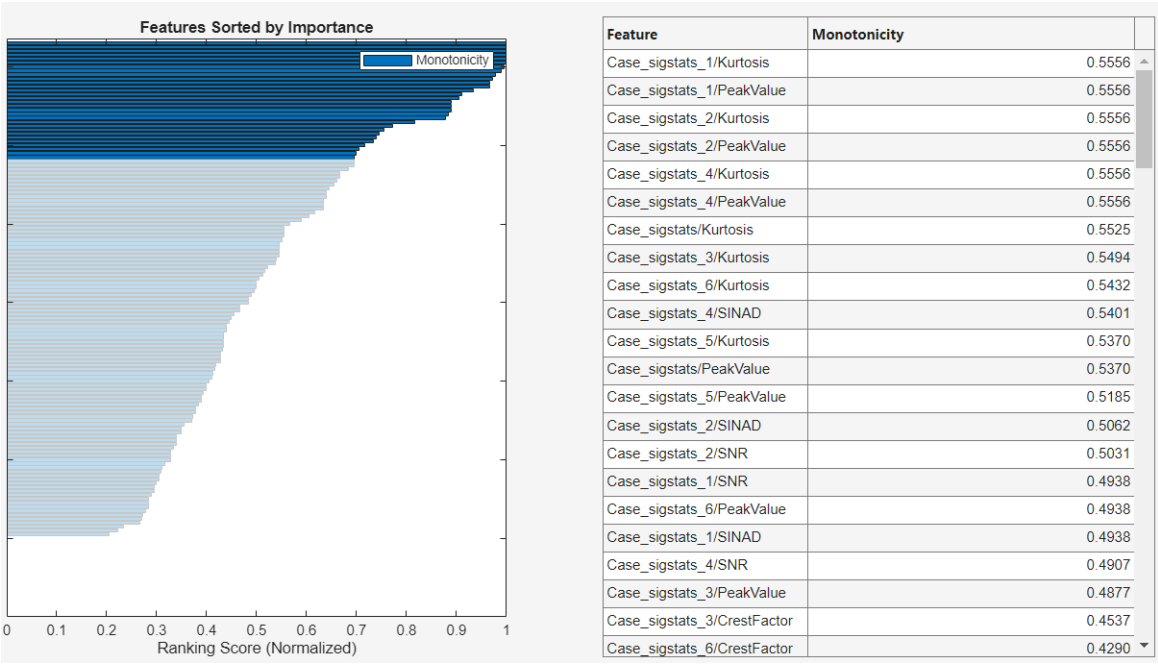


Figura 3.4: Range di frequenze e picchi significativi selezionato



Feature	Monotonicity
Case_sigstats_1/Kurtosis	0.5556
Case_sigstats_1/PeakValue	0.5556
Case_sigstats_2/Kurtosis	0.5556
Case_sigstats_2/PeakValue	0.5556
Case_sigstats_4/Kurtosis	0.5556
Case_sigstats_4/PeakValue	0.5556
Case_sigstats/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/PeakValue	0.5370
Case_sigstats_5/PeakValue	0.5185
Case_sigstats_2/SINAD	0.5062
Case_sigstats_2/SNR	0.5031
Case_sigstats_1/SNR	0.4938
Case_sigstats_6/PeakValue	0.4938
Case_sigstats_1/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

Figura 3.5: Ranking delle features: Estrazione delle prime 30 features sulla base della Monotonicity

feature deve aumentare o diminuire in modo costante, senza inversioni di tendenza significative. Nel Diagnostic Feature Designer, la Monotonicity viene calcolata basandosi su una metrica statistica che

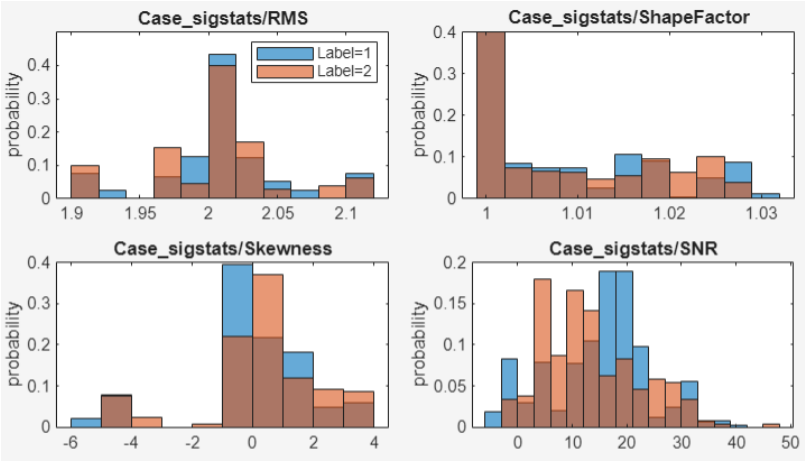


Figura 3.6: Monitoraggio delle features per le labels 1 e 2

analizza la correlazione tra la feature e la progressione dell'anomalia, assegnando un punteggio più alto alle feature che presentano una variazione più regolare e coerente. Questo criterio è particolarmente utile nel contesto della OC-SVM, dove seppur le etichette non consentano un ranking supervisionato, è comunque possibile individuare le feature che meglio separano lo stato normale dalle anomalie in base al loro comportamento monotono.

Il dataset di training per il classificatore binario, al contrario, conteneva campioni di **contaminazione da bolle (2)**, e di **guasto alle valvole solenoidi (1)**, abbiamo perciò impostato la variabile `Label` come variabile condizionale. Nel caso in cui il dataset presenti due etichette distinte, impostare una variabile label come condizionale nel Diagnostic Feature Designer è particolarmente utile per migliorare l'analisi e il ranking delle features. Permette infatti di valutare l'efficacia delle features separatamente per ciascuna classe, rendendo possibile una selezione più mirata di quelle che meglio discriminano tra le due categorie (Figura 3.6). Quando le etichette sono due, senza una gestione condizionale della label, il ranking delle feature potrebbe essere influenzato da una distribuzione sbilanciata dei dati o da differenze nella varianza tra le classi. In questo modo, invece, è possibile identificare più chiaramente le feature che presentano comportamenti distinti tra le due categorie.
In questo caso, abbiamo effettuato un ranking supervisionato, il che significa che è stato effettuato sfruttando le informazioni fornite dalle etichette dei dati. Un ranking supervisionato consente di valutare direttamente la capacità discriminante delle feature rispetto alle classi, anziché basarsi esclusivamente su proprietà intrinseche della distribuzione dei dati. I metodi usati, come nel task 1, sono stati il **T-Test** e la **ROC Curve**. Il primo è una tecnica statistica che misura la differenza tra le medie di due gruppi e valuta se tale differenza è significativa rispetto alla loro varianza. Nel contesto del ranking delle feature, ci permette di individuare le feature che presentano valori significativamente diversi tra le due classi, favorendo quelle che mostrano una separazione netta. La ROC curve (*Receiver Operating Characteristic*) è invece una tecnica basata sull'analisi delle prestazioni di una feature come classificatore binario. La ROC curve misura la capacità di una feature di distinguere tra le due classi, valutando il True Positive Rate (sensibilità) e il False Positive Rate. Un valore elevato dell'area sotto la curva (AUC - Area Under the Curve) indica che la feature è altamente discriminante. Nel ranking supervisionato, le feature con un valore AUC più alto vengono considerate più utili per la classificazione.
In questo caso, abbiamo ottenuto un risultato migliore, estraendo la totalità delle features, ma abbiamo effettuato anche diversi tentativi estraendone un numero limitato, senza ricavare risultati migliori.

3.1.3 Addestramento dei modelli

One Class Support Vector Machine (OC-SVM)

La *One-Class Support Vector Machine (OC-SVM)*, come anticipato, è un algoritmo di apprendimento non supervisionato utilizzato per la rilevazione di anomalie. A differenza dei tradizionali classificatori SVM, che richiedono dati etichettati appartenenti a più classi, l'OC-SVM viene addestrata utilizzando esclusivamente campioni di una sola classe proprio come nel nostro caso, assumendo che questi rappresentino il comportamento "normale" del sistema. L'obiettivo del modello è apprendere i confini di questa distribuzione e identificare eventuali campioni futuri che si discostano da essa come anomalie. Il funzionamento dell'OC-SVM si basa sulla mappatura dei dati in uno spazio ad alta dimensionalità tramite una funzione kernel e sulla costruzione di un iperpiano che separa i dati dalla regione

circostante. I punti che si trovano all'interno di questa regione vengono considerati appartenenti alla classe nota, mentre quelli che cadono al di fuori sono etichettati come anomalie. L'algoritmo cerca quindi di massimizzare il margine attorno ai dati di training, garantendo una buona generalizzazione del modello.
Nella pratica, per l'addestramento del modello in MATLAB, è stata utilizzata la funzione `ocsvm`, che permette di costruire un classificatore One-Class SVM a partire da un set di feature estratto dai dati. Il codice utilizzato per il training è riportato di seguito (Listing 3.3).

Listing 3.3: Addestramento della One Class Support Vector Machine

```
% ============================
% One-Class SVM Training
% ============================

% Load features table
load('FeatureTable_SVM.mat', 'FeatureTable_SVM');

rng("default")
Mdl = ocsvm(FeatureTable_SVM, StandardizeData=true, KernelScale="auto");
```

Nel codice sopra, viene caricata la tabella delle features estratte in precedenza dai dati, utilizzata per addestrare il modello. `rng("default")` imposta il generatore di numeri casuali per garantire che i risultati siano riproducibili. La funzione `ocvsm`, addestra un modello One-Class SVM utilizzando le feature contenute in `FeatureTable_SVM`. Al suo interno, il parametro `StandardizeData=true` normalizza le features per garantire che abbiano media zero e varianza unitaria, migliorando la stabilità numerica dell'algoritmo, mentre `KernelScale="auto"` seleziona automaticamente il parametro di scala del kernel, ottimizzando la separazione dei dati.

Classificazione Binaria

Nel caso della classificazione binaria, come per il precedente task, abbiamo fatto uso del tool *Classification Learner*, e il processo di addestramento dei modelli è stato pressochè analogo, ad eccezione del criterio di scelta del miglior modello. Infatti, a causa della limitatezza di campioni all'interno del dataset di addestramento, abbiamo scelto in questo caso di non effettuare un ulteriore split in training e test set, per poter addestrare il modello con un numero più alto possibile di campioni. Per questo motivo, dopo aver addestrato tutti i modelli messi a disposizione nel CL, li abbiamo valutati utilizzando come metrica la **Validation Accuracy**.
La Figura 3.7 mostra il confronto tra i diversi modelli di classificazione addestrati nel Classification Learner, valutati in base alla Validation Accuracy. L'asse orizzontale rappresenta la percentuale di accuratezza ottenuta in fase di validazione, mentre l'asse verticale mostra i numeri assegnati ai vari modelli. I modelli sono raggruppati e colorati in base alla loro tipologia, rendendo più immediata l'analisi delle prestazioni per ogni classe di algoritmi. In base all'immagine, si può osservare che i migliori modelli in termini di Validation Accuracy appartengono alla famiglia dei **Trees**. In particolare, alcuni modelli di alberi decisionali (*Tree-based models*) ottengono le performance più elevate, con un'accuratezza di validazione superiore all'80%. Questa categoria di modelli include algoritmi come *Decision Trees*, *Boosted Trees* e *Bagged Trees*, che sfruttano strutture gerarchiche per suddividere lo spazio dei dati in regioni sempre più specifiche, migliorando così la capacità di classificazione. Tra questi, il modello con la migliore performance complessiva (Accuracy Validation: **93.5%**) è il *RUS Boosted Trees*.
Il modello *RUS Boosted Trees (Random UnderSampling Boosted Trees)* combina due tecniche:

- *Random UnderSampling (RUS)*: Questa tecnica di undersampling viene applicata per bilanciare dataset sbilanciati. Se una classe ha molti più esempi rispetto all'altra, RUS rimuove alcuni campioni dalla classe dominante per evitare che il modello venga influenzato da questa disparità, migliorando così la capacità di discriminare entrambe le classi in modo equo.

- *Boosting (Ensemble Learning)*: Il boosting è una tecnica di apprendimento ensemble che combina più alberi decisionali (Decision Trees) deboli in un classificatore più forte. Ogni nuovo albero viene addestrato enfatizzando gli errori commessi dai modelli precedenti, migliorando progressivamente la capacità predittiva del modello finale.

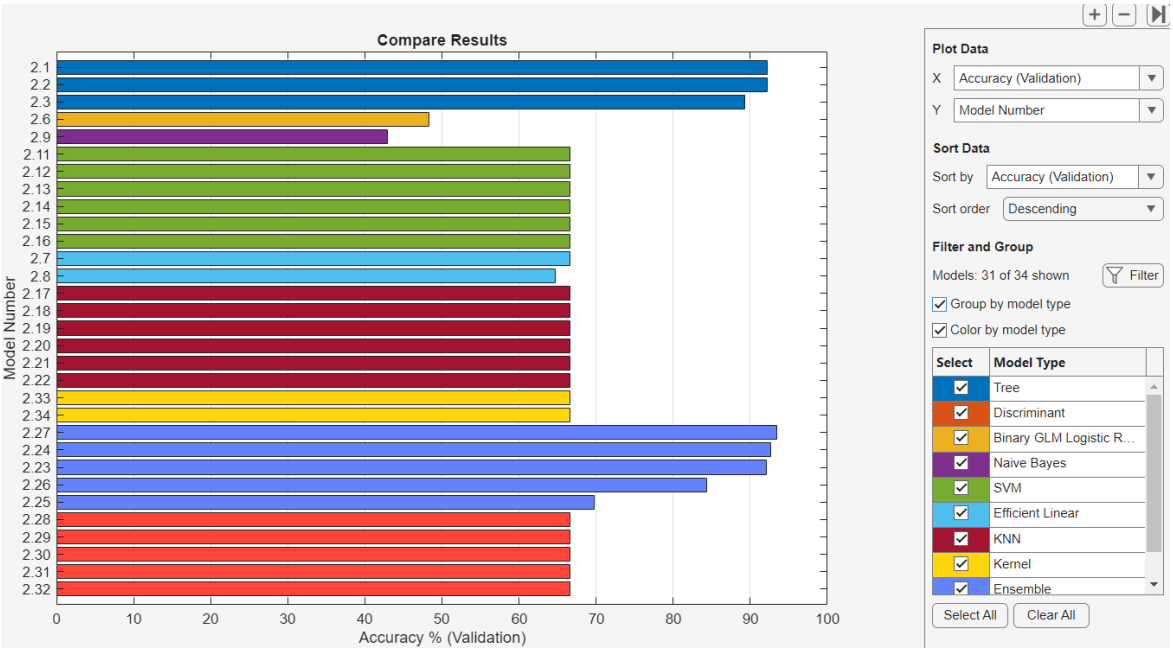
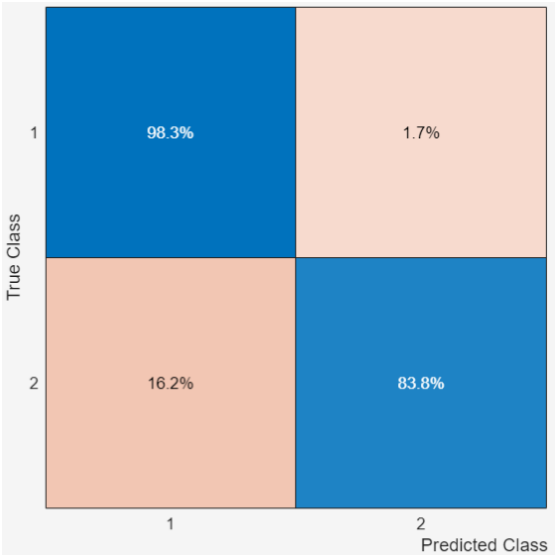


Figura 3.7: Confronto della performance dei modelli trainati

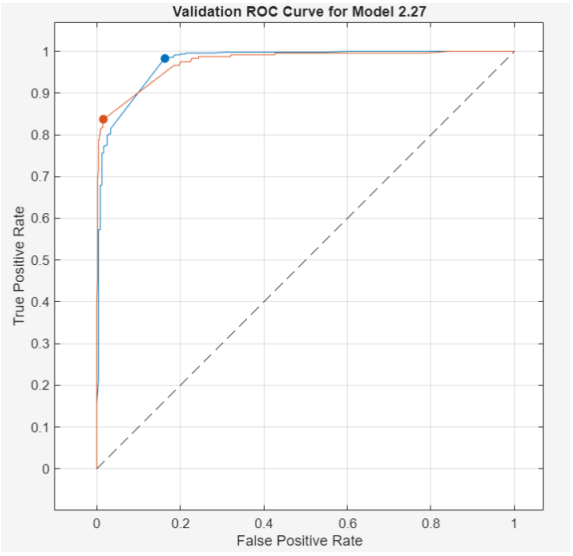
L'uso del RUS Boosted Trees è particolarmente vantaggioso in dataset sbilanciati e con pochi campioni, come nel nostro caso. Infatti, il Random Undersampling riduce il rischio di overfitting sulla classe dominante, mentre il Boosting garantisce un miglioramento della capacità predittiva attraverso l'uso iterativo di alberi deboli. Questa combinazione ha permesso al modello di ottenere un'accuratezza di validazione molto elevata rispetto agli altri classificatori, rendendolo la scelta ottimale per questo problema di classificazione binaria.
Nella matrice di confusione in Figura 3.8a, si osserva che la classe 1 viene classificata correttamente nel 98.3% dei casi (True Positive Rate - TPR), mentre solo l'1.7% dei campioni appartenenti a questa classe viene erroneamente classificato come classe 2. La classe 2, invece, viene classificata correttamente nell'83.8% dei casi, ma presenta un False Negative Rate (FNR) del 16.2%, il che significa che una parte dei campioni appartenenti alla classe 2 viene confusa con la classe 1. Questo suggerisce che il modello è molto efficace nel riconoscere la classe 1, mentre la classe 2 presenta una percentuale di errore più elevata, probabilmente dovuta alla minore disponibilità di dati di addestramento o a una sovrapposizione delle caratteristiche tra le due classi.
Anche nella Curva di ROC(Receiver Operating Characteristic) in Figura 3.8b, che valuta le prestazioni del modello in termini di compromesso tra True Positive Rate (TPR) e False Positive Rate (FPR), si vede che il modello ha un'elevata capacità di discriminazione tra le classi, avvicinandosi all'angolo superiore sinistro del grafico, che rappresenta la condizione ideale. I punti evidenziati sulla curva rappresentano soglie di decisione specifiche, con diversi compromessi tra sensibilità e specificità. La distanza dalla diagonale (linea tratteggiata) indica un buon livello di separabilità tra le classi: più la curva si allontana dalla diagonale, più il modello è efficace.
Complessivamente, il RUS Boosted Trees mostra dunque prestazioni molto solide, con un'elevata accuratezza e un buon bilanciamento tra le due classi, anche se la classe 2 risulta leggermente più difficile da classificare correttamente.

3.1.4 Hyperparameter Tuning

Per migliorare ulteriormente la performance su entrambe le classi, anche in questo caso abbiamo utilizzato il tool *Experiment Manager*, per effettuare un **Hyperparameter Tuning** sul modello *RUS Boosted Trees*, ottenendo ottimi risultati. Come in precedenza, la strategia di ottimizzazione adottata è stata la *Bayesian Optimization*, e abbiamo testato 40 configurazioni dei parametri rappresentati in Tabella 3.1.
Il miglior modello, corrispondente alla decima iterazione del processo, è stato ottenuto con la seguente



(a) RUSBoostedTrees: Confusion Matrix



(b) RUSBoostedTrees: ROC Curve

Parameter	**Range**
Method	[”Bag”, ”GentleBoost”, ”LogitBoost”, ”AdaBoostM1”, ”RUSBoost”]
NumLearningCycles	[10,500]
LearnRate	[0.001,1]
MinLeafSize	[1,360]

Tabella 3.1: Parameter tuning ranges for the model RUS Boosted Trees

configurazione: {`method:GentleBoost, NumLearningCycles:370.0000, LearnRate:0.0013, MinLeafSize:30000`}, e ha raggiunto una Validation Accuracy del 99.58%, migliorando notevolmente i risultati iniziali. Dalla matrice di confusione in Figura 3.9, si osserva infatti che il classificatore ha predetto correttamente tutti i 480 campioni della classe 1, mostrando un'ottima capacità di riconoscere questa classe. Per la classe 2, 237 campioni su 240 sono stati predetti correttamente, mentre solo 3 campioni sono stati erroneamente classificati come classe 1 (False Negatives). Calcolando le metriche di valutazione principali dai dati della matrice, emerge dunque come il modello abbia raggiunto un'accuratezza del 99.58%, il che indica un'elevata capacità di generalizzazione, una precisione del 100%, una recall del 98.75%, e un F1 Score del 99.37%, che dimostra un ottimo equilibrio tra precisione e recall, un modello estremamente performante.

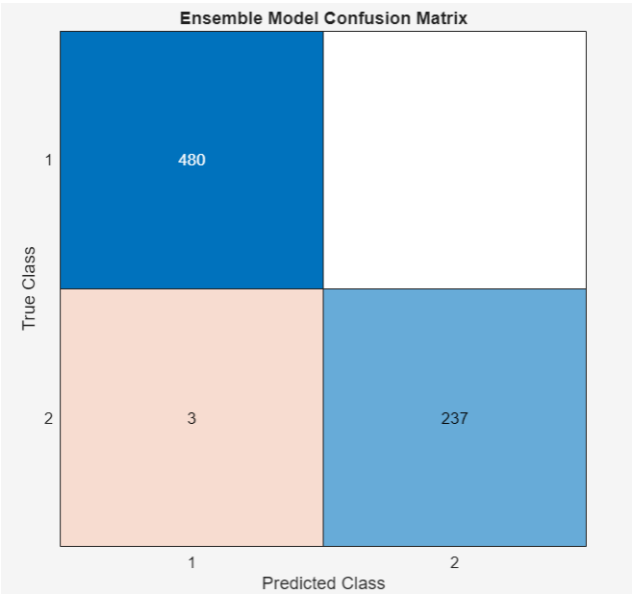


Figura 3.9: Best RUS Boosted Trees: Confusion Matrix

Dati questi risultati, abbiamo dunque salvato questo modello pre-addestrato direttamente dall'Experiment Manager, che ci permette di riutilizzarlo senza doverlo addestrare nuovamente, per poi utilizzarlo per effettuare le predizioni sul dataset di test originale, e valutare le prestazioni su dati mai visti prima.

3.1.5 Testing e Valutazione della performance

Come già spiegato in precedenza, la **pipeline finale di testing** è composta da due fasi a cascata: la classificazione tra anomalie note e sconosciute utilizzando il modello di OC-SVM (One-Class Support Vector Machine) preaddestrato, e la successiva classificazione binaria delle anomalie note come contaminazione da bolle o guasti alla valvola solenoide, utilizzando il modello di RUS Boosted Trees addestrato in precedenza. Inizialmente, è stato caricato il dataset di test, `filteredTestSet`, ricavato in precedenza nella fase di preprocessing del dataset a partire dal dataset di test originale, eliminando tutti quei campioni che nel task 1 erano stati classificati come normali. Successivamente, è stata caricata la tabella che avrebbe contenuto le previsioni finali. Per l'analisi, è stata utilizzata la funzione generata attraverso il Diagnostic Feature Designer per estrarre le caratteristiche dai dati di test, fondamentale per il successivo processo di predizione. In questa fase, il modello OC-SVM è stato impiegato per identificare se ciascuna anomalia fosse già nota (ovvero, una "anomalia conosciuta") o sconosciuta (Listing 3.4). Il modello OC-SVM, essendo un modello di classificazione per anomalie, predice come `1` le anomalie sconosciute e come `0` quelle note.

Listing 3.4: OC-SVM: Features Extraction and Prediction

```
%% Known-Unknown Anomalies Classification (OC-SVM)

% Features Extraction
[testFeatureTable, x1] = featuresExtractionFunction_SVM(filteredTestSet);

% Anomalies Prediction (0 = known, 1 = unknown)
[tf_test, scores_test] = isanomaly(Mdl, testFeatureTable);
```

E' stato poi utilizzato un sistema di voting per aggregare le previsioni del modello OC-SVM, in quanto nella fase di estrazione di features, avendo impostato una frame policy a 0.128, questo faceva si che si avessero 10 predizioni per ogni campione del dataset. Abbiamo dunque prelevato blocchi di dieci previsioni e, se almeno una di queste risultava `1`, l'anomalia veniva etichettata come sconosciuta (Listing 3.5).

Listing 3.5: Sistema di voting utilizzando un treshold di una predizione su dieci

```
for i = 1:num_samples
    start_idx = (i - 1) * num_windows + 1;  % Start of block of 10
    end_idx = start_idx + num_windows - 1;  % End of block

    % If at least one window is 1, we mark the sample as 1
    if any(tf_test(start_idx:end_idx) == 1)
        final_labelsOCSVM(i) = 1;
    else
        final_labelsOCSVM(i) = 0;
    end
end
```

Una volta classificate e identificate le anomalie (note e sconosciute), quelle etichettate come sconosciute sono state rimosse dal dataset, in modo da mantenere solo le anomalie note per la classificazione successiva.
La seconda fase ha visto infatti l'applicazione del RUS Boosted Trees per la distinzione dei due tipi specifici di anomalie. Per fare ciò, sono state estratte nuove caratteristiche dai campioni di anomalie note, utilizzando la funzione generata attraverso il DFD, dedicata alla classificazione binaria. Successivamente, è stato utilizzato il modello binario pre-addestrato per prevedere la classe di ciascun campione. Ogni campione è stato classificato come `bolla di contaminazione` (etichetta 2) o `guasto alla valvola` (etichetta 1) tramite la predizione del modello (Listing 3.6).

Listing 3.6: RUS Boosted Trees: Features Extraction and Prediction

```
%% Classification of the two known anomalies: bubble contamination and valve fault

% Feature Extraction
[testFeatureTable2, x1] = featuresExtractionFunction_binary(testSet_KnownAnomalies);
```

```
5
6 % Prediction using pre-trained model
7 predictedLabelsArray = trainedModel_binary.predictFcn(testFeatureTable2);
```

Come per il task 1, sono stati poi testati due sistemi di aggregazione delle previsioni. Il primo sistema utilizza la **moda**, che assegna a ogni campione la classe più frequente tra le dieci previsioni effettuate. In questo modo, se una classe appare più frequentemente rispetto alle altre, quella classe viene scelta come etichetta finale per il campione. Il secondo sistema si basa su una **soglia** in cui si classifica un campione come `bolla di contaminazione` se questa classe appare un numero sufficiente di volte (superando una soglia definita). In questo secondo task, i risultati rimanevano tuttavia invariati utilizzando i due sistemi, abbiamo perciò utilizzato il sistema di voting basato sul treshold.
Infine, la performance del modello è stata valutata confrontando le etichette previste con quelle reali contenute nel file delle risposte.

Dalla matrice di confusione in Figura 3.10, si osserva che per la classe `Bubble Anomaly`, tutte le istanze sono state correttamente identificate, senza alcun errore di classificazione, e non ci sono stati False Positives o False Negatives. Per la classe `Valve Anomaly`, il modello ha correttamente classificato 8 istanze (True Positives). Tuttavia, 2 guasti alla valvola sono stati erroneamente classificati come `Normal` (False Negatives). Anche in questo caso, non ci sono False Positives, ovvero nessuna altra classe è stata erroneamente classificata come `Valve`. Allo stesso modo, per la classe `Unknown`, il modello ha correttamente identificato 5 anomalie sconosciute (True Positives), ma una anomalia sconosciuta è stata erroneamente identificata come `Normal` (False Negative). Per la classe `Normal`, il modello ha correttamente classificato 20 istanze (True Positives). Tuttavia, 3 istanze anomale sono state erroneamente classificate come `Normal` (False Positives). Da questi risultati, è possibile dunque derivare le diverse metriche di performance per il modello (Tabella 3.2).

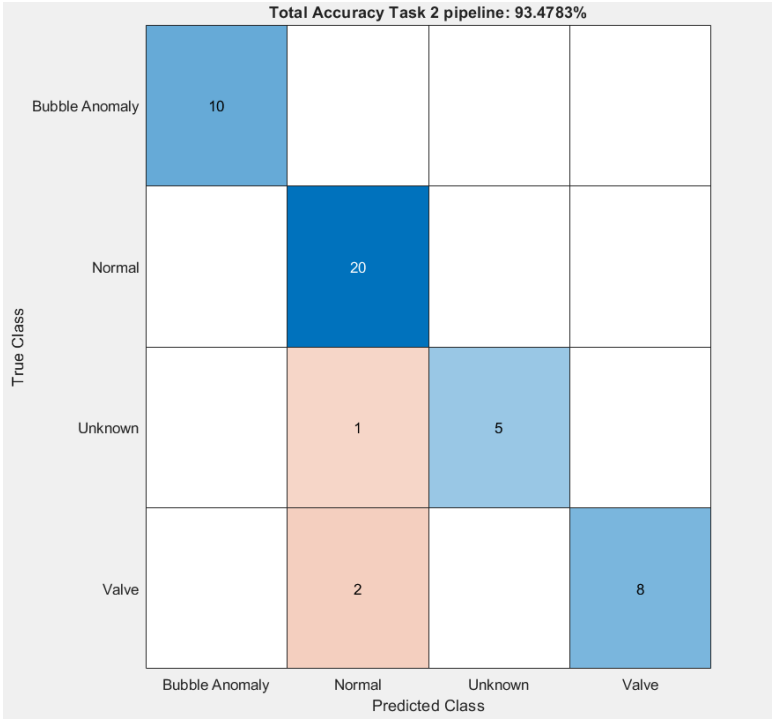


Figura 3.10: Testing Task 2: Confusion Matrix

Classe	Precision	Recall	F1-Score
Normal	1.000	0.870	0.930
Bubble Anomaly	1.000	1.000	1.000
Unknown	0.833	1.000	0.909
Valve	0.727	0.889	0.800
Accuracy: 0.936			

Tabella 3.2: Metriche di valutazione per ciascuna classe.

E' da notare, tuttavia, come i falsi positivi riscontrati nella classe `Normal` derivano interamente da errori commessi nel Task 1, ossia nella fase di filtraggio iniziale delle anomalie. Questo significa che tali errori non sono imputabili al modello di classificazione delle anomalie sviluppato nel Task 2, ma piuttosto a un'errata selezione dei campioni normali nella fase precedente. Se considerassimo il

Task 2 isolatamente, ovvero valutando esclusivamente la capacità del modello di distinguere tra le anomalie note e sconosciute e successivamente tra le diverse categorie di anomalie note, otterremmo una performance perfetta. In questo scenario, la matrice di confusione risulterebbe priva di errori, e tutte le metriche di valutazione sarebbero pari al **100%**. Questo risultato evidenzia che il sistema di classificazione sviluppato nel Task 2 è estremamente affidabile e che il margine di miglioramento risiede principalmente nella fase di classificazione tra campioni normali e anomali iniziale. Migliorando quest'ultima, sarebbe possibile ottenere un'accuratezza globale perfetta su tutto il processo.

Capitolo 4

Task 3: Determinazione della Posizione delle Bolle nel Sistema di Propulsione

Il Task 3 della challenge si concentra sull'**identificazione della posizione delle bolle d'aria** all'interno del sistema di propulsione sperimentale. Questo compito rappresenta una componente critica per il miglioramento delle tecnologie di monitoraggio della salute dei sistemi di propulsione dei veicoli spaziali di nuova generazione. Le bolle d'aria, che possono formarsi nei condotti durante il funzionamento di un veicolo spaziale, influenzano infatti direttamente la velocità del suono all'interno del fluido di lavoro e generano perturbazioni nelle fluttuazioni di pressione. Sebbene la quantità di bolle nel sistema sia considerata costante, la loro posizione può variare in una delle otto locazioni previste: BV1, BP1, BP2, BP3, BP4, BP5, BP6 e BP7 (Figura 4.1). L'identificazione accurata della posizione

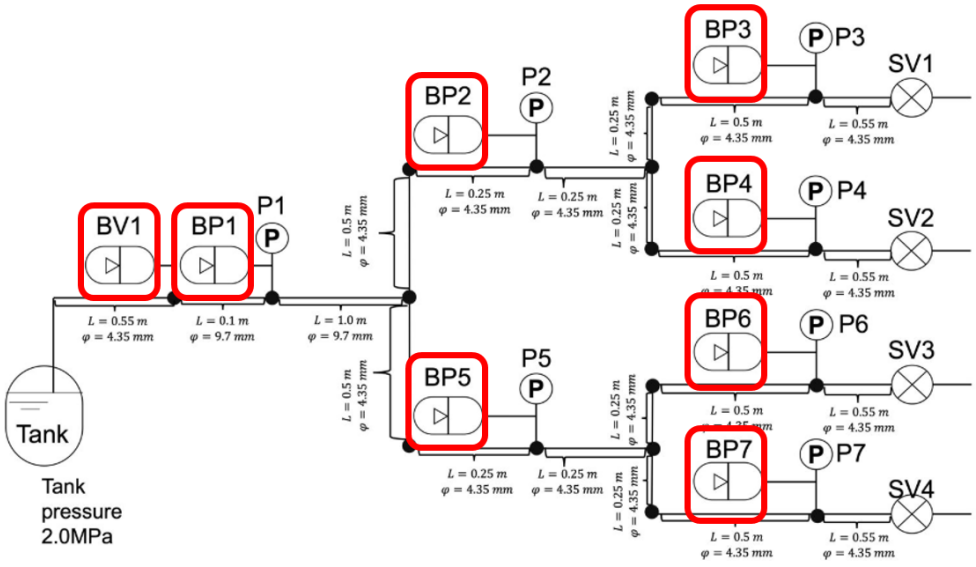


Figura 4.1: Punti in corrispondenza dei quali possono verificarsi anomalie da bolle

di queste bolle è fondamentale per mantenere le prestazioni ottimali del sistema e prevenire potenziali anomalie che potrebbero compromettere le operazioni. Un rilevamento tempestivo e accurato di tali anomalie consente non solo di diagnosticare malfunzionamenti in tempo reale, ma anche di attuare misure correttive efficaci, riducendo il rischio di guasti gravi. Questo task contribuisce dunque significativamente anche al miglioramento delle capacità di manutenzione predittiva, consentendo di

monitorare continuamente lo stato del sistema e di prevedere eventuali anomalie prima che possano causare danni irreparabili.

4.1 Implementazione

Nella pratica, come esposto in seguito, il Task 3 si configura come un problema di classificazione multiclasse, poiché l'obbiettivo è assegnare ciascun set di dati anomali a una delle otto possibili categorie corrispondenti alle posizioni delle bolle (BV1, BP1, BP2, BP3, BP4, BP5, BP6, BP7).
Come di consueto, anche questo task è stato affrontato attraverso un processo composto dalle seguenti fasi: **Preprocessing dei dati**, **Estrazione delle features**, **Addestramento**, **Hyperparameter Tuning**, **Testing e Valutazione della performance**.

4.1.1 Data Preprocessing

Per quanto riguarda il **Preprocessamento** dei dati di training, in maniera analoga al caso precedente abbiamo ricavato dal set di training originale un nuovo dataset di addestramento che contenesse solo i campioni di interesse, in questo caso le anomalie dovute a contaminazione da bolle. Una volta filtrati tali dati, li abbiamo poi etichettati utilizzando labels numeriche corrispondenti alle 8 posizioni possibili: `BV1, BP1, BP2, BP3, BP4, BP5, BP6, BP7` all'interno del sistema di propulsione (Listing 4.1).

Listing 4.1: Task 3: Training Data Preprocessing

```
trainingData = cell(numel(data),2);
for i = 1:numel(data)
    trainingData{i, 1} = data{i};

    if labelsTable{i, "Condition"} == "Normal" || (labelsTable{i,"BP1"}=="No" &&
    ↪ labelsTable{i,"BP2"}=="No" && labelsTable{i,"BP3"}=="No" && labelsTable{i,"BP4
    ↪ "}=="No" && labelsTable{i,"BP5"}=="No" && labelsTable{i,"BP6"}=="No" &&
    ↪ labelsTable{i,"BP7"}=="No" && labelsTable{i,"BV1"}=="No")
        trainingData{i, 2} = 'toDrop';
    elseif labelsTable{i,"BP1"} == "Yes"
        trainingData{i,2} = 1;
    elseif labelsTable{i,"BP2"} == "Yes"
        trainingData{i,2} = 2;
    elseif labelsTable{i,"BP3"} == "Yes"
        trainingData{i,2} = 3;
    elseif labelsTable{i,"BP4"} == "Yes"
        trainingData{i,2} = 4;
    elseif labelsTable{i,"BP5"} == "Yes"
        trainingData{i,2} = 5;
    elseif labelsTable{i,"BP6"} == "Yes"
        trainingData{i,2} = 6;
    elseif labelsTable{i,"BP7"} == "Yes"
        trainingData{i,2} = 7;
    elseif labelsTable{i,"BV1"} == "Yes"
        trainingData{i,2} = 8;
    end

end
```

Per avere inoltre un dataset di test che facesse al caso nostro, dopo aver assegnato un ID univoco a ciascun case, abbiamo invece filtrato i dati del set originale utilizzando le predizioni ottenute nel precedente task, mantenendo solo quei dati che erano stati classificati come anomalia da bolle. Come già spiegato nel precedente capitolo, lo scopo dell'introduzione dell'ID è stata la riaggregazione delle predizioni finali e il mantenimento di un riferimento chiaro tra i dati originali e quelli filtrati. Abbiamo dunque ottenuto il nostro `filteredTestSet`, una tabella composta da tre colonne: `ID`, `Case`, e `Label`. Anche in questo caso abbiamo introdotto nella colonna `Label` dei valori di default settati a 0, poiché il dataset di test fornito inizialmente non includeva informazioni sulle etichette dei dati. Abbiamo infine creato la tabella `predictions` per raccogliere le predizioni del modello, nella quale abbiamo posto a

0 tutti quei casi che nei task precedenti erano stati classificati come dati normali o anomalie di tipo diverso da quella da bolle. Il valore dei restanti campioni è stato determinato poi dal risultato della corrente classificazione multiclasse.

4.1.2 Features Extraction

A seguire, siamo poi passati all'**estrazione delle features** dal dataset di training attraverso il *Diagnostic Feature Designer (DFD)*, grazie al quale abbiamo generato le usuali *Time Domain Features* e le *Spectral Features* in maniera analoga ai casi precedenti (`frame policy:0.128`, `number of curves:24`, `Model Order:10`, `Frequency Band interval:5Hz-400Hz`, `Number of peaks:2`), impostando la variabile `Label` come `Condition Variable` in modo da permettere un'analisi maggiormente accurata (Figura 4.2).

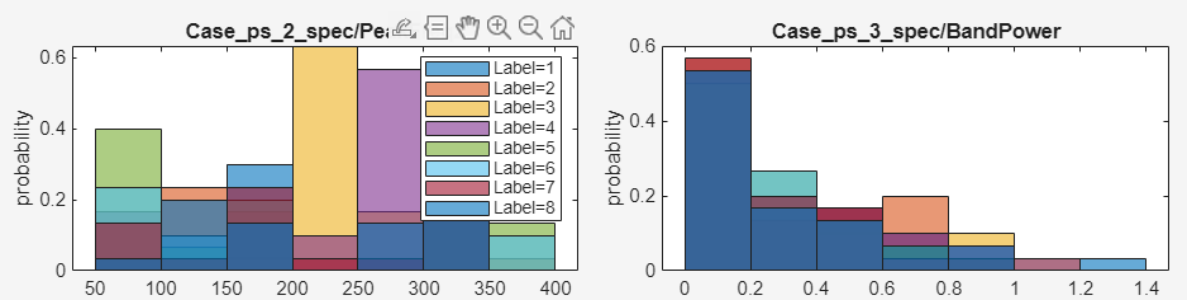


Figura 4.2: Task 3: example of histogram from DFD

In questo caso il ranking finale delle features è stato effettuato attraverso due metodi particolarmente adatti ai problemi di classificazione multiclasse, in cui si hanno più di due etichette da prevedere:

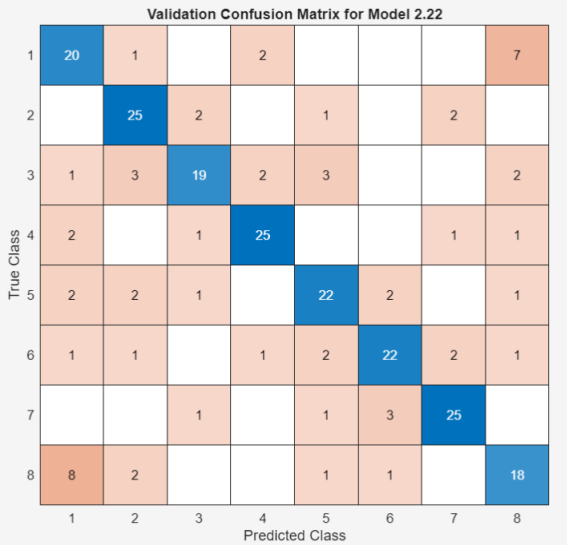
- *One-Way ANOVA*: L'analisi della varianza a una via valuta la significatività statistica delle differenze tra le medie di due o più gruppi. In questo contesto, è stata utilizzata per analizzare se esistono differenze significative nelle caratteristiche estratte dai segnali di ciascuna classe, assegnando un punteggio più alto alle features che mostrano maggiore discriminatività tra le diverse posizioni delle bolle.
- *Kruskal-Wallis*: Questo test non parametrico rappresenta una generalizzazione del test di *Wilcoxon* per più di due gruppi. Risulta particolarmente utile quando non si può assumere che i dati seguano una distribuzione normale. Nel nostro caso, ha permesso di identificare le features che presentano variazioni significative tra le classi, senza fare ipotesi sulla distribuzione dei dati.

Nonostante il ranking effettuato, si è deciso tuttavia di estrarre tutte le features disponibili. Questo approccio consente di mantenere un ampio spettro di informazioni utili, riducendo il rischio di eliminare caratteristiche potenzialmente rilevanti. Inoltre, l'utilizzo di tutte le features permette al modello di machine learning di individuare autonomamente le combinazioni più significative, migliorando così la capacità di generalizzazione e aumentando l'accuratezza nella classificazione delle posizioni delle bolle.

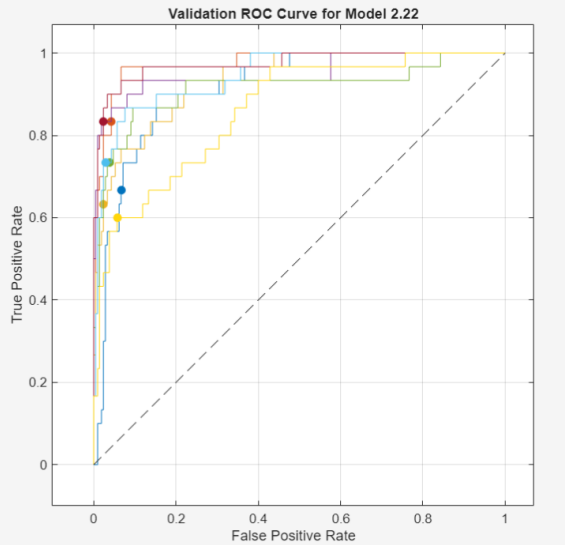
4.1.3 Training e Hyperparameter Tuning

In fase di **training**, come nel secondo task, abbiamo addestrato tutti i modelli messi a disposizione dal tool *Classification Learner (CL)*, e abbiamo valutato il migliore sulla base della `Validation Accuracy`, senza effettuare un ulteriore split preliminare del dataset a causa del numero limitato di campioni (24). Il modello che ha fornito le migliori prestazioni è stato l'*Ensemble Boosted Trees*, con un'accuratezza di validazione pari al **73,3%**. Il modello Ensemble Boosted Trees è un metodo di apprendimento supervisionato che combina molteplici alberi decisionali deboli in un classificatore più forte. L'addestramento avviene in maniera sequenziale: ogni albero successivo viene costruito correggendo gli errori commessi dai modelli precedenti, attraverso l'assegnazione di pesi maggiori ai campioni mal classificati. Questo

approccio incrementale migliora progressivamente la capacità del modello di distinguere tra le diverse classi, rendendolo particolarmente adatto a problemi complessi come quello affrontato in questo task. I vantaggi principali del modello includono dunque una maggiore robustezza rispetto agli alberi decisionali singoli, la capacità di gestire dati rumorosi e un miglioramento delle prestazioni generali anche in presenza di un numero elevato di features, come nel nostro caso.
La matrice di confusione (Figura 4.3a) mostra che le classi BP2, BP4 e BP7 sono state identificate correttamente con un'elevata frequenza (25, 25 e 25 corretti rispettivamente), mentre la classe BV1 e BP8 presentano una maggiore confusione. Queste discrepanze suggeriscono una somiglianza nei pattern di alcune classi, causando ambiguità al modello. Anche la curva ROC multiclasse (Figura 4.3b) mostra buone performance generali, con la maggior parte delle curve che si avvicinano all'angolo in alto a sinistra, indicando un buon compromesso tra tasso di veri positivi e falsi positivi. Tuttavia,



(a) Ensamble Boosted Trees: Confusion Matrix



(b) Ensamble Boosted Trees: ROC Curve

alcune classi, come BV1 e BP8, presentano curve leggermente più distanti, suggerendo una minore capacità del modello di separarle perfettamente dalle altre classi.
Questa analisi conferma che, nonostante l'accuratezza generale soddisfacente e una buona capacità di apprendimento del modello, vi sono sfide nella classificazione di classi con pattern simili. Abbiamo perciò migliorato le prestazioni attraverso un **Hyperparameter Tuning**, effettuato grazie al tool *Experiment Manager*. La strategia di ottimizzazione adottata anche qui è stata la *Bayesian Optimization*, e abbiamo testato 50 configurazioni dei parametri rappresentati in Tabella 4.1. Il miglior modello, corrispondente alla ventesima iterazione del processo, è stato ottenuto con la seguente configurazione: {`method:Bag, NumLearningCycles:265.0000, LearnRate:Nan, MinLeafSize:10000`}, e ha raggiunto una Validation Accuracy del **84.17%**, migliorando i risultati iniziali. La nuova matrice di confusione (Figura 4.4) mostra un miglioramento generale nell'accuratezza di classificazione. Seppur permangono alcune difficoltà per le classi 1 e 8, le classi BP2, BP3, BP4, BP5, BP6 e BP7 presentano prestazioni eccellenti, con solo pochissime osservazioni classificate erroneamente. Il tuning ha portato perciò a una maggiore stabilità nelle predizioni, rendendo il modello più robusto e accurato, mantenendo alti standard di prestazio-

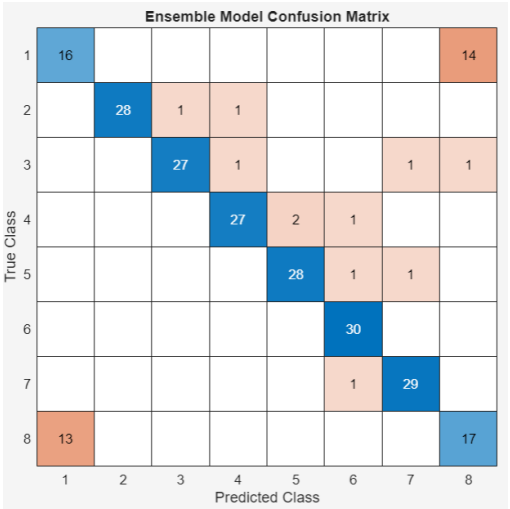


Figura 4.4: Tuned Ensamble Boosted Trees: Confusion Matrix

Parameter	Range
Method	[”Bag”, ”AdaBoostM1”, ”RUSBoost”]
NumLearningCycles	[10,500]
LearnRate	[0.001,1]
MinLeafSize	[1,120]

Tabella 4.1: Parameter tuning ranges for the model Ensamble Boosted Trees

ne e migliorando sensibilmente la classificazione per la maggior parte delle classi rispetto al modello iniziale.

4.1.4 Testing e Valutazione della performance

A questo punto abbiamo potuto testare il modello sul dataset di test. La **fase di testing** è stata eseguita caricando inizialmente il dataset di test `filteredTestSet`, contenente le anomalie di contaminazione da bolle, filtrate dai risultati del Task 2. Successivamente, è stata caricata la tabella con le predizioni (`predictions`) e la funzione di estrazione delle features (`featuresExtraction_task3_function`), insieme al modello pre-addestrato `trainedModel`. La classificazione delle anomalie di contaminazione da bolle è stata effettuata estraendo le features dal dataset di test tramite la funzione di estrazione caricata. Il modello pre-addestrato è stato utilizzato per prevedere le etichette di ciascun campione. Il sistema di voto utilizzato in questo caso è basato sulla moda, implementato per aggregare le predizioni effettuate su ciascun campione, composto da 10 finestre di analisi. Questo sistema, come già detto in precedenza, assegna a ciascun campione l'etichetta più frequente tra le 10 predizioni effettuate. Le predizioni aggregate sono state poi aggiornate nella tabella delle predizioni, e infine la performance del modello è stata valutata caricando il file delle risposte corrette (`answer.csv`), confrontando le etichette vere (trueLabels) con quelle predette (`predictedLabels`). I risultati ottenuti dal modello di classificazione mostrano un'accuratezza del **100%** sul set di test, che è un risultato molto positivo e indica che il modello ha classificato correttamente tutte le istanze testate. Anche dalla matrice di confusione (Figura 4.5) si osserva come non ci siano errori di classificazione, tutte le istanze sono state infatti correttamente assegnate alle rispettive classi.

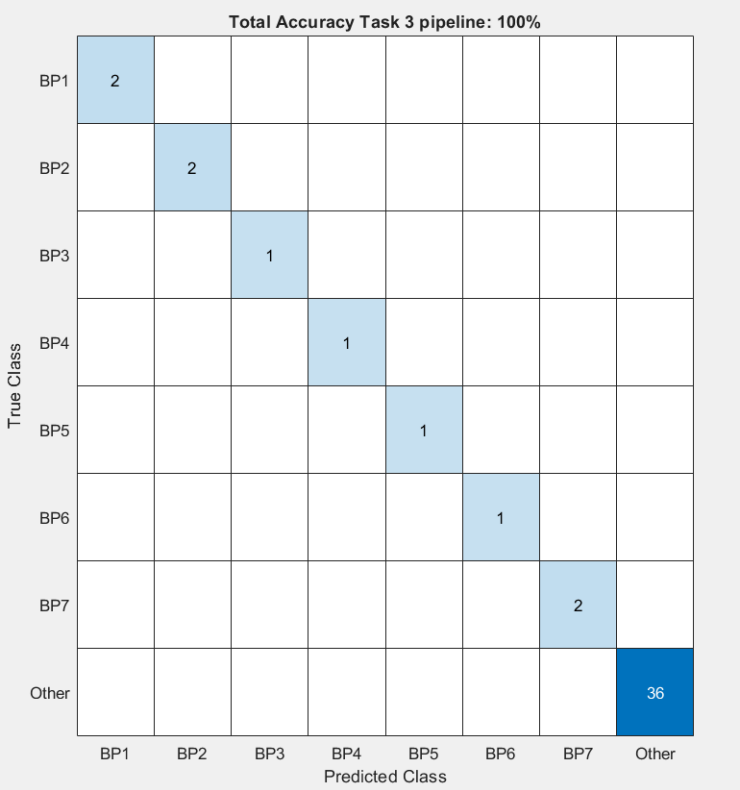


Figura 4.5: Testing Task 3: Confusion Matrix

Capitolo 5

Task 4: Determinazione delle valvole soggette a guasto nel Sistema di Propulsione

Nel contesto del monitoraggio della salute e della manutenzione predittiva (PHM) del sistema di propulsione spaziale, è fondamentale identificare con precisione quale delle quattro valvole solenoidi (SV1, SV2, SV3, SV4) presenti nel sistema abbia subito un guasto. Nel task 4 abbiamo affrontato

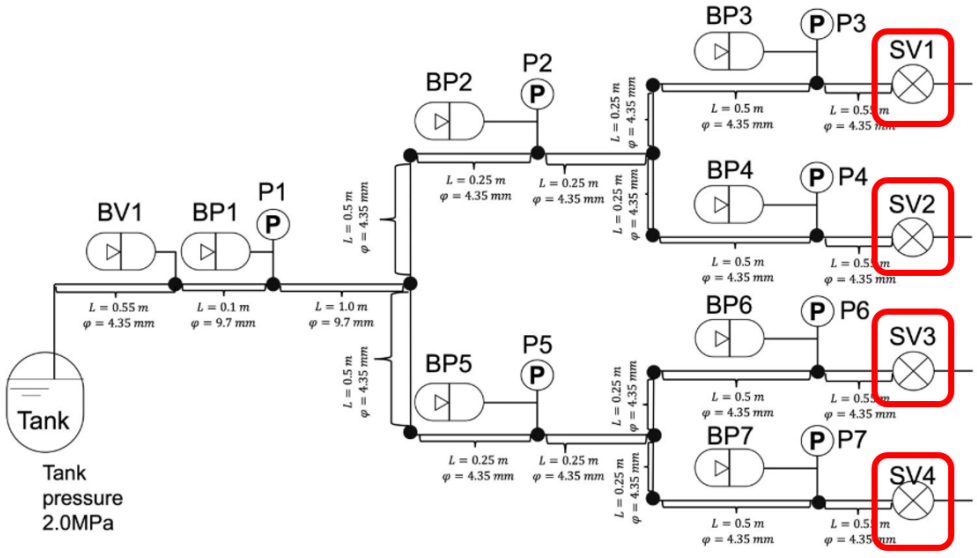


Figura 5.1: Valvole Solenoidi

dunque questo problema, utilizzando le informazioni fornite sul rapporto di apertura delle valvole. Infatti, come già anticipato, queste ultime regolano il flusso del fluido nel sistema di propulsione aprendo e chiudendo con un rapporto di apertura standard del 100% (completamente aperte) o dello 0% (completamente chiuse). Tuttavia, un guasto può causare un'apertura parziale della valvola, con un rapporto di apertura compreso tra 0% e 100%, determinando una riduzione del volume di fluido che passa attraverso la valvola. Il corretto funzionamento di queste valvole è essenziale per garantire la corretta erogazione del fluido di lavoro e la risposta dinamica del sistema di propulsione. Infatti, un'anomalia nel loro comportamento può compromettere le prestazioni della missione o, nei casi più critici, causare un malfunzionamento totale del sistema di propulsione. L'identificazione della posizione specifica dei guasti classificati come anomalie alle valvole solenoidi garantisce dunque la sicurezza e l'affidabilità del sistema di propulsione, prevenendo situazioni di rischio, ottimizzando la manutenzione

e la diagnostica, e riducendo i tempi di intervento permettendo di adottare strategie di controllo o mitigazione prima che il problema si aggravi.

5.1 Implementazione

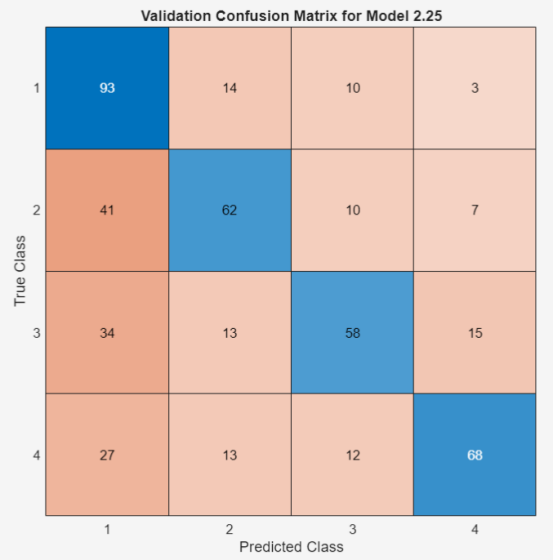
Nella pratica, analogamente al precedente task, anche il Task 4 si configura come un problema di classificazione multiclasse, nel quale le labels corrispondono alle 4 valvole solenoidi del sistema (SV1, SV2, SV3, SV4), mostrate in Figura 5.1.
Il Task 4 è stato affrontato seguendo un approccio pressoché analogo a quello adottato per il Task 3. Come in precedenza, il flusso di lavoro si è articolato in diverse fasi: **Preprocessing dei dati**, **Estrazione delle features**, **Addestramento del modello**, **Ottimizzazione degli iperparametri**, **Testing** e **Valutazione della performance**.

5.1.1 Data Preprocessing e Features Extraction

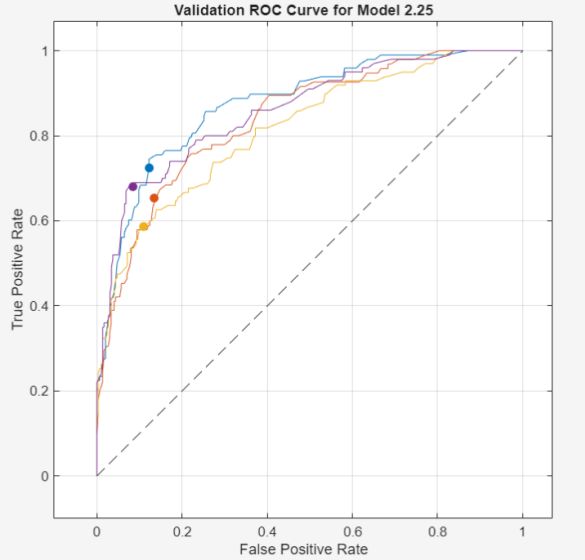
Durante il **preprocessing**, sono stati costruiti i dataset di training e di test contenenti esclusivamente i dati di interesse (anomalie alle valvole), etichettando poi i dati di addestramento in base alla loro posizione all'interno del sistema.
Successivamente, attraverso il *Diagnostic Feature Designer (DFD)*, sono state estratte sia *Time Domain Features* sia *Spectral Features* (`frame policy:0.128`, `Number of Curves:48`, `model order:10`, `Frequency Band Interval:30Hz-300Hz`, `Number of peaks:2`, `Number of Cross Validation Folds:10`). Abbiamo poi utilizzato il criterio di selezione *One-Way ANOVA* , al fine di identificare le variabili più discriminanti per la classificazione. In seguito a numerosi tentativi effettuati (Tabella A.4, Tabella A.5 in Appendice), abbiamo ottenuto il miglior risultato estraendo 15 features.

5.1.2 Training e Hyperparameter Tuning

L'**addestramento** è stato condotto utilizzando diversi modelli disponibili nel *Classification Learner (CL)*, con una selezione finale basata sulla Validation Accuracy. In questo caso, il modello con le migliori prestazioni è risultato essere il *Subspace KNN*. Il modello *Subspace K-Nearest Neighbors (Subspace KNN)* rappresenta un'evoluzione del metodo K-Nearest Neighbors (KNN) e si basa sull'idea di migliorare la capacità di classificazione attraverso la creazione di sottospazi casuali delle features. A differenza del KNN tradizionale, che considera tutte le variabili disponibili per ogni classificazione, il Subspace KNN adotta un approccio basato sull'*ensemble learning*. Questo significa che il modello costruisce più classificatori, ognuno dei quali lavora su un diverso sottoinsieme di feature selezionato casualmente, per poi combinare le loro predizioni in una decisione finale più robusta e accurata.
Il funzionamento del modello può essere descritto in diverse fasi. In primo luogo, vengono selezionate in modo casuale alcune feature del dataset, riducendo la complessità del problema e mitigando gli effetti della maledizione della dimensionalità. Successivamente, su ogni sottospazio così definito, viene addestrato un classificatore KNN indipendente, che acquisisce una prospettiva parziale ma diversificata sulla struttura dei dati. Infine, le predizioni di tutti i classificatori vengono aggregate attraverso un metodo di voto, che può basarsi sulla scelta più frequente o sulla media delle distanze dai vicini più prossimi.
Uno dei principali vantaggi di questa tecnica è la capacità di ridurre il rischio di overfitting, migliorando la generalizzazione del modello soprattutto in presenza di dati rumorosi o con un elevato numero di variabili irrilevanti. Inoltre, il Subspace KNN si dimostra particolarmente efficace nei contesti in cui le feature disponibili sono numerose e in parte ridondanti, poiché sfrutta la diversità introdotta dalla selezione casuale per ottenere prestazioni più stabili.
Nel contesto del Task 4, l'applicazione di questo modello ha permesso di ottenere un'accuratezza di validazione pari al **58.50%**. Sebbene questa percentuale non sia particolarmente elevata, suggerisce che il modello abbia comunque individuato alcune strutture nei dati utili per la classificazione, anche se la distinzione tra alcune classi potrebbe risultare più complessa a causa della natura delle anomalie e della distribuzione delle feature nel dataset.
L'analisi dei risultati ottenuti con il modello *Subspace KNN* evidenzia alcune caratteristiche chiave riguardo le sue prestazioni. Osservando la matrice di confusione (Figura 5.2a), si nota che il modello ha una buona capacità di classificare correttamente la prima e la quarta classe, rispettivamente con



(a) Subspace KNN: Confusion Matrix



(b) Subspace KNN: ROC Curve

93 e 68 predizioni corrette. Tuttavia, si evidenziano anche alcuni errori significativi, in particolare nella seconda e terza classe, dove il modello tende a confondere un numero considerevole di istanze con le altre categorie. Ad esempio, nella seconda classe, 41 campioni sono stati erroneamente classificati come appartenenti alla prima classe, mentre nella terza classe, si osserva una distribuzione di errori più variegata, con una quantità rilevante di dati assegnati in modo errato alla prima e alla quarta classe.
Dal punto di vista della capacità discriminativa del modello, la curva di ROC (Figura 5.2b) offre ulteriori spunti di valutazione. Si può notare che le curve associate alle varie classi mostrano un buon compromesso tra la True Positive Rate e la False Positive Rate, indicando una discreta capacità di separare correttamente le classi, anche se non in maniera ottimale. Il fatto che la curva ROC non sia perfettamente aderente all'angolo superiore sinistro del grafico suggerisce che il modello potrebbe beneficiare di ulteriori miglioramenti, ad esempio mediante una più accurata selezione delle feature o un'ottimizzazione più spinta degli iperparametri.
Nonostante i vari tentativi, l'ottimizzazione degli iperparametri in questo caso non ha tuttavia permesso di migliorare le prestazioni del modello, con un valore di accuratezza di validazione pari a 42.92%. Anche in questo caso è stata utilizzata la *Bayesian Optimization*, e abbiamo testato 50 configurazioni dei parametri rappresentati in Tabella 5.1, cioè:

Parameter	**Range**
NumLearningCycles	[10,500]
NumNeighbors	[1,240]
Distance	["cityblock","chebychev","correlation","cosine","euclidean" ,"hamming","jaccard","mahalanobis","minkowski","seuclidean", "spearman"]
Standardize	["true","false"]

Tabella 5.1: Parameter tuning ranges for the model Subspace KNN

- **NumLearningCycles**
- **NumNeighbors**: indica il numero di vicini più prossimi considerati nel processo di classificazione. Valori bassi rendono il modello più sensibile al rumore, mentre valori elevati possono portare a un'eccessiva generalizzazione.
- **Distance**: specifica la metrica utilizzata per calcolare la distanza tra i punti. Opzioni come *euclidean* e *cityblock* misurano distanze geometriche, mentre *correlation* e *cosine* valutano similarità angolari tra i vettori di caratteristiche.

- **Standardize**: determina se le feature devono essere normalizzate prima dell'addestramento. La normalizzazione aiuta a ridurre l'impatto delle scale di misura diverse tra le variabili.

5.1.3 Testing e Valutazione della performance

Infine, il modello migliore è stato testato sul dataset di test filtrato, contenente le sole anomalie date da valvole, in maniera analoga al caso precedente. I risultati risultano soddisfacenti, l'analisi della matrice di confusione finale relativa alla fase di testing (Figura 5.3) evidenzia un'accuratezza complessiva del 91.30%, suggerendo il mantenimento di ottime prestazioni nel riconoscere correttamente le varie classi. Tuttavia, è importante distinguere i risultati ottenuti per il Task 4 da quelli derivanti dalle fasi precedenti. Osservando la matrice, si nota che la classe `Other`, che raccoglie gli esiti parziali dei task precedenti, è stata classificata correttamente in 36 casi, con un numero molto limitato di errori. Questo indica che il modello è stato in grado di distinguere efficacemente i dati appartenenti a questa categoria dalle classi specifiche del Task 4. Focalizzandosi sulle prestazioni relative esclusivamente al Task 4, ovvero sulle classi SV1, SV2, SV3 e SV4, si riscontra una certa difficoltà nella distinzione tra alcune classi. In particolare, la classe SV1 è stata in parte confusa con la classe `Other`, con 2 campioni erroneamente assegnati, mentre un ulteriore errore ha portato un'istanza di SV1 a essere scambiata per SV2. Per la classe SV2, si osservano due predizioni errate, una delle quali assegnata a SV4 e l'altra a SV3. Analogamente, la classe SV3 presenta due casi in cui i dati reali sono stati assegnati erroneamente alla stessa classe, suggerendo una certa ambiguità nei pattern riconosciuti. Infine, la classe SV4 ha registrato due classificazioni corrette, confermando una discreta capacità del modello di riconoscere questa specifica categoria.
Nel complesso, i risultati ottenuti dimostrano che il sistema è in grado di riconoscere le classi con sufficiente precisione, sebbene ci siano ancora alcune incertezze nella distinzione tra categorie specifiche del Task 4.

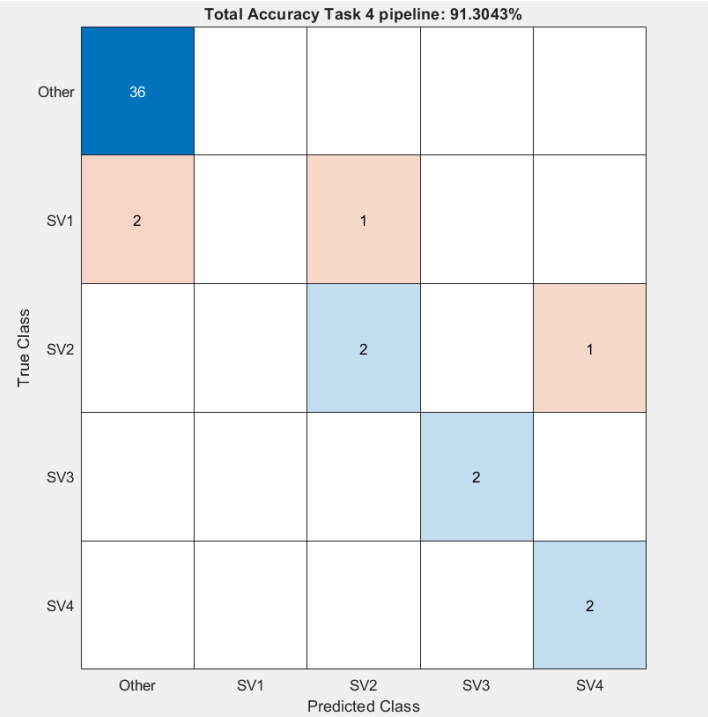


Figura 5.3: Testing Task 4: Confusion Matrix

Capitolo 6

Task 5: Predizione del Rapporto di Apertura delle Valvole Solenoidi

Il Task 5 prevede la **predizione del rapporto di apertura delle valvole solenoidi** identificate come guaste. Questo valore, compreso tra 0% e 100%, rappresenta il grado di apertura della valvola e fornisce informazioni cruciali sul funzionamento del sistema.
L'importanza di questo task è legata alla necessità di garantire il corretto funzionamento del sistema di propulsione. Una valvola che non si apre completamente o non si chiude del tutto può infatti compromettere le manovre del veicolo, riducendo l'efficienza del sistema e aumentando il rischio di malfunzionamenti. Identificare con precisione l'apertura delle valvole consente di adottare strategie di compensazione e di manutenzione preventiva, migliorando l'affidabilità complessiva del sistema.
Nella pratica, il problema è stato affrontato come un **task di regressione**, poiché l'obiettivo è stimare un valore continuo, ossia il grado di apertura della valvola. A differenza della classificazione, che assegna un'osservazione a una categoria discreta (ad esempio, "normale" o "anomalo"), la regressione è adatta a problemi in cui l'output varia su un intervallo continuo. In questo caso, il rapporto di apertura può assumere qualsiasi valore tra 0% e 100%, rendendo la regressione il metodo ideale per predire questa quantità in funzione delle caratteristiche del segnale di pressione registrato dai sensori. Utilizzare un modello di regressione permette inoltre di catturare sottili variazioni nei dati, migliorando la precisione della diagnosi e supportando eventuali decisioni operative.

6.1 Implementazione

Come di consueto, anche l'implementazione di quest'ultimo task consisteva nelle usuali fasi di **Preprocessing dei dati**, **Estrazione delle features**, **Training**, **Hyperparameter Tuning** e **Testing**, seppur con alcuni accorgimenti per adattare il procedimento al problema di regressione affrontato.

6.1.1 Data Preprocessing

Come nei casi precedenti, abbiamo in primo luogo sottoposto i dati a un'operazione di **Preprocessamento** in preparazione alle fasi di Training e Testing, ricavandone i due nuovi dataset. Il dataset di training, in particolare, contiene in questo caso i dati relativi al rapporto di apertura delle valvole solenoidi, ed è stato ottenuto tramite lo stesso tipo di operazioni di filtraggio eseguite nei precedenti task.
Allo stesso modo, il dataset di test è stato ottenuto filtrando solo i dati relativi alle anomalie alle valvole solenoidi nei dati di test originali. Un unico accorgimento adottato in questo caso è stato nella creazione della tabella `predictions5`, in cui abbiamo assegnato un valore di `100` ai dati relativi ai campioni normali, alle anomalie sconosciute e alle anomalie da bolle, anziché `0`, ad indicare un funzionamento corretto delle valvole.

6.1.2 Estrazione delle Features

La fase successiva è stata l'**estrazione delle features** dal dataset di training ottenuto, tramite il tool *Diagnostic Features Designer*. Le *Time Domain Features* e le *Spectral Features* sono state ottenute impostando i seguenti valori: `frame policy:0.128`, `Number of Curves:48`, `model order:10`, `Frequency Band Interval:30Hz-300Hz`, `Number of peaks:2`. Abbiamo poi testato i modelli **ANOVA**, **Trendability** e **Monotonicity** per valutare il ranking delle features.
Il criterio di **Trendability** introdotto nel presente task misura la capacità di una feature di mostrare una tendenza monotona rispetto alla progressione di una condizione di guasto o degrado nel tempo. In altre parole, una feature con alta Trendability dovrebbe aumentare o diminuire in modo consistente man mano che il sistema si deteriora. Il calcolo si basa sulla correlazione tra i valori della feature e l'ordine temporale o la severità del guasto. MATLAB in particolare utilizza principalmente il *Coefficiente di Spearman* o altri metodi di correlazione per quantificare quanto la feature segua una relazione monotona con la progressione del guasto. Se la feature aumenta costantemente (o diminuisce) durante la progressione del guasto, avrà un valore elevato di Trendability. Viceversa, se la feature mostra variazioni casuali senza una tendenza chiara, il valore sarà basso.
Il miglior metodo ad ogni modo è risultato essere il criterio di **Monotonicity**, e in base a quest'ultimo abbiamo dunque estratto le `15` migliori features.

6.1.3 Training

Per la predizione del rapporto di apertura, è stato utilizzato il *Regression Learner*, un tool di MATLAB che, in maniera analoga al *Classification Learner*, consente di addestrare e confrontare diversi modelli di regressione automaticamente. Dopo aver caricato le features nel tool, sono stati dunque addestrati tutti i modelli di regressione disponibili per individuare il migliore, utilizzando per l'addestramento una **Cross Validation** con 10 folders. Il miglior modello è stato individuato valutando l'**Errore Quadratico Medio della Radice (RMSE)**, che misura la differenza media tra i valori predetti e quelli reali. La formula dell'RMSE è la seguente:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{6.1}$$

dove:

- y_i rappresenta il valore reale del rapporto di apertura della valvola solenoide,
- $\hat{y}_i$ è il valore predetto dal modello,
- n è il numero totale di osservazioni nel dataset.

L'RMSE è stato scelto come metrica principale poiché è interpretabile nella stessa unità dei dati originali, facilitando la comprensione dell'errore medio del modello. Inoltre, a differenza del MSE, l'RMSE non amplifica eccessivamente gli errori più grandi, offrendo una misura più equilibrata della precisione predittiva. Un valore basso di RMSE indica che il modello è in grado di generalizzare bene sui dati, mentre un valore elevato suggerisce una maggiore deviazione tra valori reali e stimati, suggerendo possibili miglioramenti nel modello o nei dati di input.
In base a questa metrica di valutazione, tra tutti i vari tentativi effettuati (Tabella A.1, Tabella A.2, Tabella A.3), il miglior modello è stato identificato nel *Gaussian Process Regression (GPR)* con *Kernel Rational Quadratic*, essendo il modello con il valore più basso di RMSE (`14.983`), per garantire il miglior compromesso tra accuratezza e capacità di generalizzazione.
Il modello GPR è una tecnica di regressione bayesiana che assume che i dati possano essere modellati come una distribuzione gaussiana su uno spazio infinito di funzioni. Il kernel *Rational Quadratic* è una combinazione di più kernel di tipo *RBF (Radial Basis Function)* con scale di lunghezza diverse, il che permette di catturare variazioni sia a lungo che a breve raggio nei dati. La formulazione del kernel Rational Quadratic è la seguente:

$$K(x, x') = \sigma^2 \left(1 + \frac{(x - x')^2}{2\alpha l^2}\right)^{-\alpha} \tag{6.2}$$

dove:

- σ^2 è la varianza,
- l è il parametro di lunghezza,
- α controlla la distribuzione delle scale di lunghezza.

I risultati sono mostrati nel *Response Plot* in Figura 6.1. Il Response Plot mostra il confronto tra

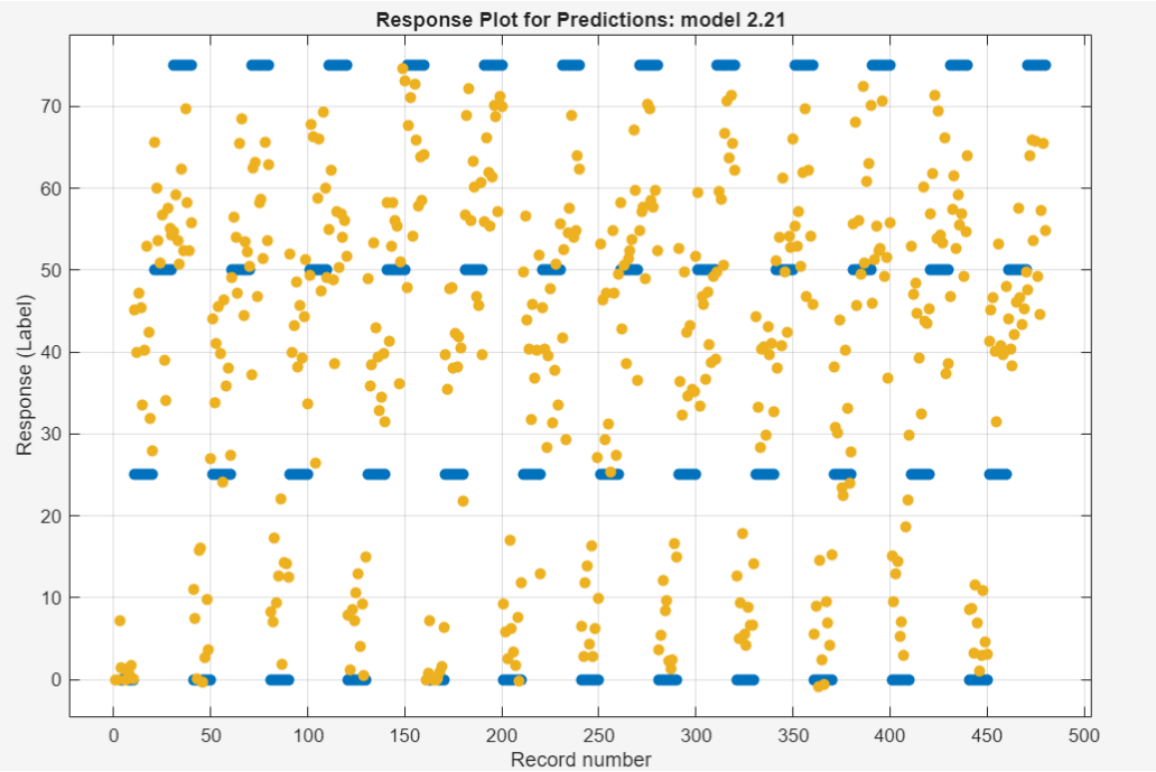


Figura 6.1: Response Plot for Predictions: Gaussian Process Regression (GPR)

i valori reali del rapporto di apertura delle valvole solenoidi, rappresentati in blu, e i valori predetti dal modello, indicati con punti gialli. Sull'asse orizzontale è riportato il numero di record analizzati, mentre sull'asse verticale è rappresentata la percentuale di apertura della valvola.
Osservando il grafico, si nota che i valori reali tendono a essere concentrati su livelli ben distinti, suggerendo che la variabile target potrebbe assumere principalmente valori discreti. Al contrario, le predizioni del modello risultano più disperse, distribuendosi in modo più continuo e meno strutturato attorno ai valori reali. Questo fenomeno evidenzia come il modello tende a interpolare i dati, generando valori che spesso non coincidono esattamente con quelli reali.
La presenza di una forte varianza nelle predizioni potrebbe indicare che il modello fatica a catturare la struttura esatta del fenomeno. Questa difficoltà potrebbe essere dovuta a diverse ragioni, come la mancanza di informazioni sufficienti nei dati di input, la scelta di un modello non del tutto adeguato alla natura del problema, oppure un possibile problema di overfitting o underfitting. Se il modello non è abbastanza complesso, potrebbe non riuscire a cogliere le relazioni tra le variabili, mentre se è eccessivamente complesso potrebbe adattarsi troppo ai dati di addestramento senza riuscire a generalizzare correttamente sui dati di test. In generale, il grafico evidenzia dunque che, pur essendo in grado di fornire una stima del rapporto di apertura, il modello mostra margini di miglioramento, specialmente nella capacità di adattarsi alla distribuzione effettiva dei dati reali.

6.1.4 Hyperparameter Tuning

Dati i risultati precedenti, abbiamo dunque proceduto con un **Hyperparameter Tuning** del modello, utilizzando il tool *Experiment Manager*, come in precedenza. In dettaglio, utilizzando come di

consueto la *Bayesian Optimization*, abbiamo ottenuto il miglior risultato testando `50` configurazioni dei parametri in Tabella 6.1, cioè:

Parameter	**Range**
Sigma	[0.0001,279.8001]
Standardize	[”true”,”false”]

Tabella 6.1: Parameter tuning ranges for Gaussian Process Regression (GPR) model.

- **Standardize**
- **Sigma**: appresenta la deviazione standard del rumore nei dati, controllando il livello di smoothing del modello. Un valore basso implica un adattamento molto preciso (rischio di overfitting), mentre un valore alto aumenta l'incertezza (rischio di underfitting). L'ottimizzazione di Sigma aiuta a bilanciare la capacità predittiva del modello.

Abbiamo in questo modo ottenuto un minimo miglioramento, con un valore di RMSE pari a **14.9040**.

I risultati sono rappresentati in Figura 6.2. Il grafico mostra il confronto tra i valori reali e quelli predetti dal modello GPR dopo l'ottimizzazione degli iperparametri. La linea nera rappresenta l'andamento ideale, in cui le predizioni coincidono perfettamente con i valori reali. Sebbene il modello catturi la struttura generale della distribuzione, si osservano gruppi di punti con una certa dispersione attorno ai valori target, suggerendo che il modello potrebbe ancora soffrire di incertezze nelle predizioni per specifici intervalli. L'ottimizzazione ha migliorato l'allineamento complessivo, ma la presenza di variabilità residua indica margini di miglioramento nella capacità predittiva. Dopo l'ottimizzazione degli iperparametri, il modello GPR con le migliori prestazioni è stato selezionato e salvato per essere poi testato sul dataset di test, consentendo di valutare le sue capacità predittive su dati non visti in precedenza.

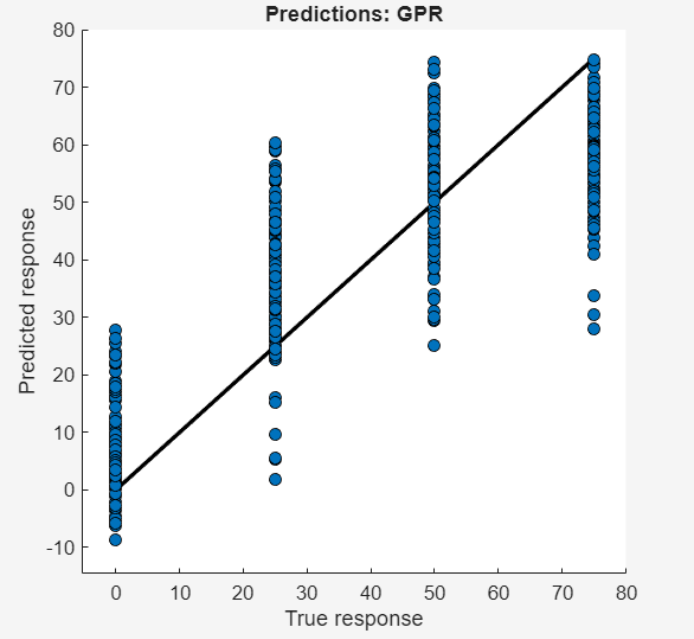
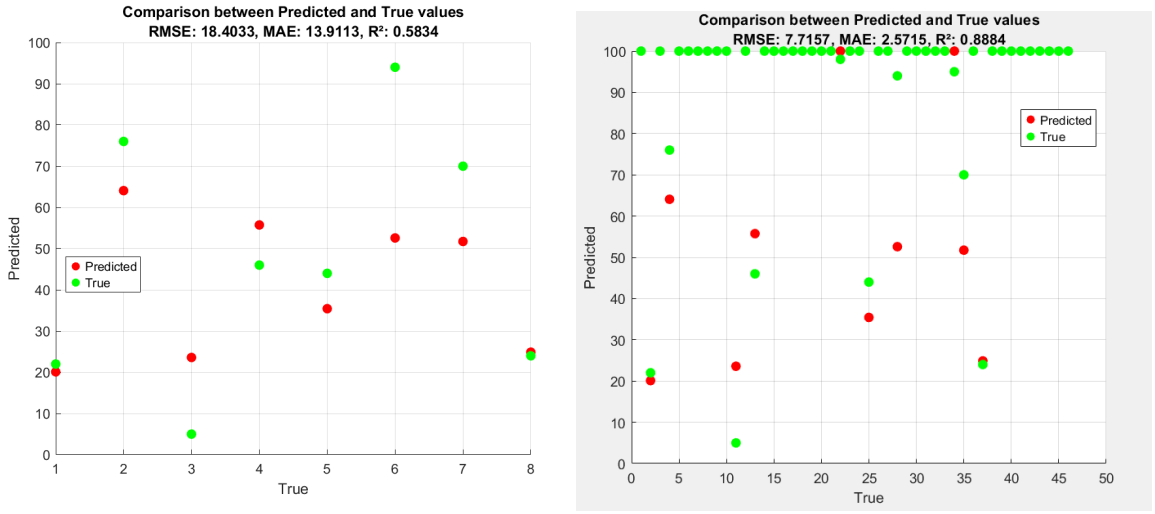


Figura 6.2: Actual vs Predicted (Validation)

6.1.5 Testing e Valutazione della Performance

E' stata dunque sviluppata una **pipeline di test** per valutare le prestazioni del modello addestrato nella previsione dell'apertura delle valvole in condizioni di guasto. Dopo aver caricato il dataset di test filtrato, il modello pre-addestrato e la funzione di estrazione delle caratteristiche, sono state estratte le feature necessarie per la classificazione. Il modello ha quindi generato una prima serie di predizioni a livello di finestra temporale, che sono state successivamente aggregate a livello di campione tramite un sistema di voting, questa volta basato sulla **Mediana**. In particolare, dato che ogni campione è suddiviso in 10 finestre, il valore finale assegnato a ciascun campione è stato calcolato come la mediana delle 10 predizioni corrispondenti, riducendo così la sensibilità a eventuali outlier o rumore nei dati.
Le predizioni finali sono state salvate in un file CSV e confrontate con le etichette reali presenti nel file delle risposte. Per valutare le prestazioni del modello, sono state calcolate metriche di regressione come l'*Errore Quadratico Medio (RMSE)*, l'Errore Assoluto Medio (MAE) e il *Coefficiente di Determinazione(R^2)*, sia per il sottoinsieme di test relativo ai guasti delle valvole che per l'intero

dataset. Infine, sono stati generati scatter plot (Figura 6.3) per visualizzare la distribuzione delle predizioni rispetto ai valori reali, consentendo di identificare eventuali errori sistematici o discrepanze nel comportamento del modello.



(a) Partial Testing Results

(b) Total testing Results

Figura 6.3: Testing Results: task 5

In dettaglio, nel primo grafico (Figura 6.3a), relativo ai soli risultati del quinto task5 (in particolare dei soli campioni affetti da guasto alle valvole nel dataset di test), l'errore quadratico medio (RMSE) è relativamente elevato (18.40), e il Coefficiente di Determinazione è pari a 0.5834, indicando una correlazione moderata tra predizioni e valori reali. In particolare si può osservare che per i campioni 1 e 8 si ha quasi una sovrapposizione perfetta tra valore reale e valore predetto; invece nei campioni 2, 4 e 5 si evidenzia un discostamento leggero con una differenza di 10 unità, mentre per i campioni 3 e 7 si ha una differenza maggiore di 20 unità e infine il campione 6 che si discosta di molto. Nel complesso si può appurare che il modello presenta delle discrepanze rilevanti in alcuni punti più che in altri, suggerendo che il modello ha difficoltà a catturare alcune relazioni nei dati.
Nel secondo grafico (Figura 6.3b) invece, le metriche sono decisamente migliori, con un RMSE più basso (7.71), un MAE molto contenuto (2.57) e un R^2 di 0.8884, che suggerisce un'elevata capacità predittiva. La distribuzione dei punti è più allineata alla diagonale ideale, segnalando una migliore aderenza tra predizioni e valori reali.
Nel contesto del Task 5, il modello di regressione ha incontrato diverse potenziali difficoltà che hanno compromesso le sue prestazioni. In primo luogo, è da considerare il fatto che la relazione tra le variabili potrebbe essere altamente non lineare e caratterizzata da una forte variabilità, rendendo complessa la capacità del modello di adattarsi correttamente ai dati.
Un altro fattore determinante potrebbe essere stato lo squilibrio nei dati di training: se il modello è stato addestrato su un insieme di dati non sufficientemente rappresentativo del Task 5, è probabile che abbia avuto difficoltà a generalizzare correttamente su questa particolare situazione. A proposito di ciò, è possibile osservare in Figura 6.1 come nel dataset di training i valori del rapporto di apertura nei campioni assumano solo 4 valori, a differenza dei campioni nel dataset di test in cui si ha una maggior variabilità nei valori assunti. A questo si aggiunge il ruolo delle feature estratte, che potrebbero non essere state sufficientemente descrittive o informative per il tipo di previsione richiesta, limitando la capacità del modello di identificare pattern significativi.
Questi fattori spiegano perché nel Task 5 il modello ha ottenuto un errore più elevato e un Coefficiente di Determinazione più basso rispetto alla valutazione complessiva su tutti i task, dove ha avuto modo di beneficiare di dati più strutturati e pattern più chiari.

Capitolo 7

Risultati Finali

7.1 Pipeline e Punteggio Complessivo

Al raggiungimento degli obbiettivi della challenge abbiamo costruito la nostra pipeline finale per poter implementare i cinque task a cascata, e abbiamo posto i risultati nel formato richiesto per poter valutare le predizioni effettuate confrontandole con le risposte corrette. Alla fine dell'elaborazione, viene infatti calcolato un punteggio complessivo normalizzato, espresso anche in percentuale, per fornire una misura delle prestazioni ottenute.

La valutazione dei vari task è stata eseguita seguendo le indicazioni fornite dalla challenge e la metrica valuta i seguenti aspetti:

1. `Task 1 (Classificazione Normale/Anormale):` la corretta classificazione viene premiata con 10 punti;
2. `Task 2 (Classificazione Anormale):` per la corretta identificazione della condizione specifica di guasto (anomalia contaminazione bolle/guasto elettrovalvola/guasto sconosciuto) nei dati correttamente classificati come anormali nel Task 1 assegna 10 punti;
3. `Task 3 (Identificazione della posizione della bolla):` per la corretta identificazione della posizione della bolla nei dati correttamente classificati come contaminati da bolla nel Task 2, sono assegnati 10 punti;
4. `Task 4 (Identificazione del guasto sulla valvola):` per la corretta identificazione della posizione della valvola guasta nei dati correttamente classificati come guasti sulla valvola nel Task 2, sono assegnati 10 punti;
5. `Task 5 (Previsione grado di apertura della valvola guasta):` per la corretta previsione del grado di apertura della valvola guasta nei dati correttamente classificati come guasti sulla valvola nel Task 2, il punteggio è calcolato come `max(|valore reale - previsione| + 20, 0)`. Questa formula incentiva l'accuratezza entro una tolleranza di 20 unità. Le previsioni entro questa tolleranza ricevono un punteggio positivo, che diminuisce man mano che l'errore aumenta, fino a un massimo di 20. Le previsioni con errori superiori a 20 unità ricevono un punteggio di 0, indicando una performance insoddisfacente.

Per il veicolo spaziale-4, i punteggi ottenuti sono raddoppiati, in considerazione della maggiore difficoltà.

Nella pipeline finale, vengono eseguite tutte le pipeline di test per ciascun task, utilizzando il comando `run` per richiamare gli script corrispondenti (Listing 7.1). Di seguito un esempio per la prima pipeline:

Listing 7.1: Task 1: Esecuzione della pipeline di test

```
1 run("task1_last\testing\testingPipeline.m");
```

Le predizioni ottenute vengono memorizzate in una tabella denominata `all_predictions` (Listing 7.2), dove ogni colonna rappresenta un task specifico:

Listing 7.2: All predictions table

```
all_predictions = table(prediction(:,1), prediction(:,2),'VariableNames', {'ID', '
    ↪ Task1'});
all_predictions.Task2 = table2array(predictionsTable(:,1));
all_predictions.Task3 = table2array(predictionsTable(:,1));
all_predictions.Task4 = table2array(predictionsTable(:,1));
all_predictions.Task5 = table2array(predictionsTable5(:,1));
```

Alla fine dello script, viene calcolato il punteggio totale sommando i punteggi parziali ottenuti nei diversi task (Listing 7.3):

Listing 7.3: Calcolo del punteggio totale

```
max_score = score_max_task1 + sum(score_max_other_tasks) + score_max_task5;
prediction_score = prediction_score/max_score*100;
```

In dettaglio, i punteggi ottenuti per ogni task sono i seguenti:

- **task1**: 650/690
- **task2**: 350/390
- **task3**: 150/150
- **task4**: 90/150
- **task5**: 150/300

Per un totale di 1390/1680 punti, pari a uno score finale di **82.76%**.

7.2 Esecuzione del Progetto

Per eseguire il progetto sviluppato, si consiglia di seguire i seguenti passaggi. In primo luogo, il codice sorgente deve essere clonato dal repository GitHub dedicato ([ZM25]), utilizzando il comando:

```
git clone https://github.com/PaceElisa/PHM_Asia_Pacific_Progetto_C1.git
```

Successivamente, aprire MATLAB e navigare all'interno della cartella del progetto. L'organizzazione del progetto prevede una suddivisione in diverse cartelle, ciascuna dedicata a uno specifico compito di analisi, come specificato in dettaglio nel `Readme` di progetto:

- **task1**: classificazione normale/anormale;
- **task2**: classificazione del tipo di anomalia;
- **task3**: rilevamento della posizione dell'anomalia di tipo bolla;
- **task4**: identificazione del guasto della valvola solenoide;
- **task5**: regressione del rapporto di apertura della valvola.

Ciascun task presenta una struttura ben definita, comprendente sottocartelle per la fase di preprocessing, estrazione delle feature, addestramento e testing. All'interno di queste ultime, per favorire una maggiore comprensione, sono state salvate le sessioni relative agli strumenti Classification Learner, Regression Learner, Diagnostic feature designer, e gli esperimenti eseguiti tramite Experiment manager per il tuning dei parametri.
L'esecuzione dell'intera pipeline di analisi è gestita dallo script principale `Pipeline_Finale.m`, che permette di eseguire in modo sequenziale tutti i task. Per avviare l'intero workflow, è sufficiente eseguire il seguente comando all'interno di MATLAB:

```
run Pipeline_Finale.m
```

In alternativa, per testare singolarmente ciascun task, è possibile eseguire gli script di testing presenti all'interno delle rispettive cartelle.

Appendice A

Tentativi	Frame Policy	Model Order	Range (HZ)	Picchi	Ranking	Features	Cross-Fold
1	0.128	10	30-300	2	Anova	15	10
2	0.128	10	30-300	2	Anova	15	10
3	0.128	10	30-300	2	Trendability	15	10
4	0.128	10	30-300	2	Trendability	15	10
5	0.128	10	30-300	2	Monotonicity	15	10
6	**0.128**	**10**	**30-300**	**2**	**Monotonicity**	**15**	**10**
7	0.128	10	30-300	2	Monotonicity	15	10
8	0.128	10	30-300	2	Monotonicity	15	10

Tabella A.1: task 5: configurazioni testate I

Tentativi	RMSE Train	Algoritmo1 (RMSE)	MAE Train	Algoritmo2 (MAE)	R^2 Train	Algoritmo3 (R^2)
1	21.239	Bagged Trees	17.717	Boosted Trees	0.420	Bagged Trees
2	21.239	Bagged Trees	177.17	Boosted Trees	0.420	Bagged Trees
3	15.605	RQ-GPR	12.767	RQ-GPR	0.690	RQ-GPR
4	15.605	RQ-GPR	12.767	RQ-GPR	0.690	RQ-GPR
5	14.983	RQ-GPR	12.450	RQ-GPR	0.710	RQ-GPR
6	**14.983**	**RQ-GPR**	**12.450**	**RQ-GPR**	**0.710**	**RQ-GPR**
7	14.983	RQ-GPR	12.450	RQ-GPR	0.710	RQ-GPR
8	14.983	RQ-GPR	12.450	RQ-GPR	0.710	RQ-GPR

Tabella A.2: task 5: configurazioni testate II

Tentativi	RMSE Hyper	RMSE Test	MAE Test	R^2 Test	RMSE All Test	MAE All test	R^2 All test
1	19.679	28.062	22.050	0.031	11.730	3.987	0.742
2	19.592	27.502	21.601	0.070	11.496	3.909	0.752
3	-	24.718	19.471	0.249	10.339	3.358	0.800
4	18078	24.480	20.733	0.263	10.240	3.758	0.803
5	-	20.670	15.645	0.475	8.657	2.873	0.860
6	**14.904**	**18.403**	**13.911**	**0.583**	**7.716**	**2.572**	**0.888**
7	-	21.125	15.610	0.451	8.846	2.867	0.853
8	14.679	18.403	13.911	0.583	7.716	2.572	0.888

Tabella A.3: task 5: configurazioni testate III

Tentativi	Frame Policy	Model Order	Range (HZ)	Picchi	Ranking	Features	Cross-Fold
1	0.128	10	30-300	2	Anova	15	10
2	0.128	10	30-300	2	Anova	20	10
3	**0.128**	**10**	**30-300**	**2**	**Anova**	**15**	**10**
4	0.128	10	30-300	2	Kruskal-Wallis	15	10
5	0.128	10	30-300	2	Anova	20	10
6	0.128	10	30-300	2	Anova	20	5
7	0.128	10	30-300	2	Anova	20	22
8	0.128	10	30-300	2	Anova	20	20
9	0.128	10	30-300	2	Anova	15	15
10	0.128	10	30-300	2	Anova	15	17
11	0.128	10	30-300	2	Anova	17	17
12	0.128	10	5-300	2	Anova	tutte	50
13	0.128	10	5-300	2	Anova	15	15
14	0.128	10	5-300	2	Anova	15	10
15	0.128	10	5-300	2	Anova	15	5
16	0.128	10	30-400	2	Anova	15	15
17	0.128	10	30-400	2	Anova	15	10
18	0.128	10	30-400	2	Anova	13	10

Tabella A.4: task 4: configurazioni testate I

Tentativi	Test Set %	Algoritmo	Accuracy Train%	Accuracy Hyper%	Accuracy Test%
1	20	Subspace KNN	64.5%	60.30%	82.61%
2	20	Subspace KNN	55.20%	46.61%	82.61%
3	**nan**	**Subspace KNN**	**58.50%**	**42.92%**	**91.31%**
4	nan	Subspace KNN	58.50%	43.96%	80.43%
5	nan	Subspace KNN	56.90%	44.58%	84.78%
6	nan	Subspace KNN	51.90%	43.33%	82.61%
7	nan	Subspace KNN	55.40%	43.75%	84.78%
8	nan	Ensemble Bagged Trees	59.20%	67.92%	86.96%
9	nan	Subspace KNN	57.70%	43.54%	89.13%
10	nan	Subspace KNN	60.20%	42.92%	80.84%
11	nan	Subspace KNN	57.50%	43.54%	80.43%
12	nan	Ensemble Bagged Trees	61.70%	76.46%	86.96%
13	nan	Ensemble Bagged Trees	58.80%	68.96%	86.96%
14	nan	Subspace KNN	57.30%	42.08%	89.13%
15	nan	Subspace KNN	54.80%	42.08%	86.96%
16	nan	Subspace KNN	60.80%	46.04%	82.61%
17	nan	Subspace KNN	60.40%	44.37%	82.61%
18	nan	Subspace KNN	61.30%	46.46%	84.79%

Tabella A.5: task 4: configurazioni testate II

Tentativi	Frame Policy	Model Order	Range (HZ)	Picchi	Features
1	0.128	10	5-225	2	20
2	0.128	10	5-225	2	25
3	0.128	10	5-225	2	25
4	0.128	10	5-225	2	25
5	0.128	10	5-225	2	25
6	0.128	10	5-225	2	25
7	0.128	10	5-300	2	25
8	0.128	10	5-300	2	25
9	0.128	10	5-300	2	25
10	**0.128**	**4**	**5-300**	**2**	**All**

Tabella A.6: task 1: configurazioni testate I

Tentativi	Cross-Fold	Test Set	Algoritmo	Accuracy Train	Accuracy Test
1	10	20	Bagged Trees	79,9%	91.30%
2	10	20	Bagged Trees	80.60%	91.30%
3	10	20	Bagged Trees	78.90%	82.90%
4	10	20	Bagged Trees	76.40%	93.48%
5	10	20	Bagged Trees	76.40%	89.13%
6	10	20	RUSBoosted Tree	78.60%	91.30%
7	10	20	Bagged Trees	80.60%	78.26%
8	10	20	Boosted Tree	77.50%	84.78%
9	10	20	RUSBoosted Tree	78.60%	84.78%
10	**10**	**20**	**Bagged Trees**	**83.3%**	**93.48%**

Tabella A.7: task 1: configurazioni testate II

Bibliografia

[(JA23] Japan Aerospace Exploration Agency (JAXA). Data challenge - phm asia pacific 2023. https://phmap.jp/program-data/, 2023.

[ZM25] Fiorani Andrea Zazzarini Micol, Elisa Pace. Phm23 spacecraft propulsion system anomaly detection. https://github.com/PaceElisa/PHM_Asia_Pacific_Progetto_C1.git, 2025. Repository GitHub del progetto.