Università Politecnica delle Marche

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
Corso di Laurea Magistrale in
Ingegneria Informatica e dell'Automazione

Progetto Manutenzione Preventiva

Riconoscimento guasti

Fabio Carosi
Monica Marconi Sciarroni
Cristina Rossetti

Indice

1 Introduzione **1**

2 Descrizione dataset **2**
2.1 Dataset Processed 3

3 Sviluppo **5**
3.1 Preprocessing 5
3.2 Estrazione features diagnostiche 8
3.3 Sviluppo modello diagnostico 9
3.4 Addestramento modello diagnostico 11

4 Risultati **13**
4.1 Undersampling 13
4.2 Classi più rappresentate 18
4.3 Sampling di no_fault 22
4.4 Classificazione senza bilanciamento 23
4.5 SMOTE con Sampling di No_Fault 27
4.6 SMOTE senza Sampling di No_Fault 31

5 Problematiche riscontrate **35**

6 Conclusioni e sviluppi futuri **36**

Elenco delle figure

2.1 Struttura del dataset 3

4.1 Cross Validation Scores 14
4.2 Matrix Confusion 15
4.3 Cross Validation Scores 19
4.4 Matrix Confusion 20
4.5 Cross Validation Scores 22
4.6 Matrix Confusion 23
4.7 Cross Validation Scores 24
4.8 Matrix Confusion 25
4.9 Cross Validation Scores 28
4.10 Matrix Confusion 28
4.11 Cross Validation Scores 32
4.12 Matrix Confusion 33

Capitolo 1

Introduzione

L'articolo [1] presenta il AirLab Failure and Anomaly (ALFA) Dataset, ossia un set di dati raccolti da diversi voli autonomi di un UAV (Unmanned Aerial Vehicle) ad ala fissa, che descrivono i guasti alle superfici di controllo e ai motori dell'aeromobile. Il set di dati comprende 23 scenari di guasto al motore e 24 scenari di guasto ad altri sette tipi di superfici di controllo, per un totale di 66 minuti di volo normale e 13 minuti di volo dopo il guasto. Inoltre, sono presenti molte ore di dati di volo autonomo, assistito da autopilota e manuale, con decine di scenari di guasto. L'obiettivo del progetto è quello di progettare un classificatore diagnostico, basandosi sui dati presenti nel dataset ALFA.

Capitolo 2

Descrizione dataset

Questo capitolo descrive i dati contenuti nel dataset in modo più dettagliato, si illustra l'elencazione dei tipi di guasto e delle anomalie presenti nel dataset e si discute gli strumenti forniti per lavorare con i dati al fine di valutare un metodo FDI o metodo di rilevamento delle anomalie.

Il dataset è composto da 4 tipi di dati:

- `Processed` → Sequenze di volo elaborate per contenere solo dati di volo autonomo e per includere i dati di ground truth solo in caso di guasto. Ogni file contiene una sequenza di volo con al massimo un guasto.
- `Raw` → Sequenze di volo grezze in cui i dati di volo in tutte le modalità sono riportati senza alcuna preelaborazione. Alcuni file possono includere scenari di test multipli di fallimento, mentre gli altri possono contenere nessun volo autonomo.
- `Telemetry` → Registri di tutti i dati di telemetria dal computer di bordo del TX2 durante i test, senza alcuna preelaborazione. I file non contengono informazioni sul ground truth del guasto e possono essere utili per i metodi di rilevamento non supervisionati.
- `Dataflash` → Registri Dataflash sull'autopilota Pixhawk dai test, senza alcuna pre-elaborazione. Anche questi file non contengono informazioni sul ground truth del guasto e possono essere utili per i metodi di rilevamento non supervisionati.

La struttura delle directory del dataset completo è mostrata nella Figura 2.1.

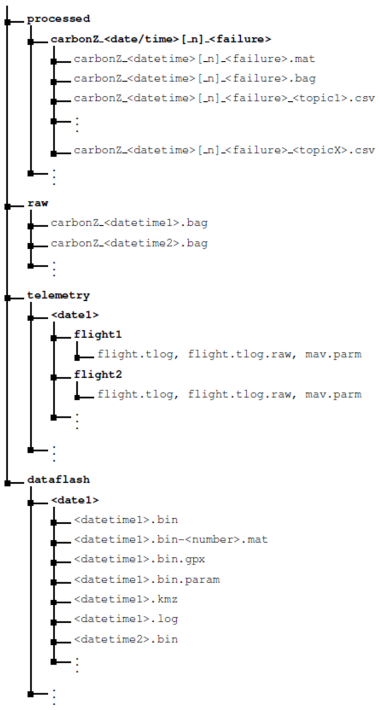

Figura 2.1: Struttura del dataset

L'interesse principale di questo progetto riguarda il primo tipo di dati, ossia le sequenze di volo elaborate, e la prossima sezione descrive questi file in modo più dettagliato.

2.1 Dataset Processed

Il dataset Processed contiene informazioni sui tipi di guasto. I tipi di guasti attualmente forniti sono elencati nella Tabella 2.1

Come si può ben notare, gran parte del dataset riguarda i guasti del motore (`engine_failures`). Questo costituisce un problema in fase di addestramento del modello di classificazione, di cui si spiega nel capitolo dedicato.

Per lo sviluppo del lavoro, sono stati presi in considerazione le sequenze di file contenute nei file csv, ed ogni file csv include un topic. I topic sono illustrati nella Tabella 2.2.

Fault Types	**# of Test Cases**
Engine Full Power Loss	23
Rudder Stuck to Left	1
Rudder Stuck to Right	2
Elevator Stuck at Zero	2
Left Aileron Stuck at Zero	3
Right Aileron Stuck at Zero	4
Both Ailerons Stuck at Zero	1
Rudder and Aileron at Zero	1
No Fault	10
Total	47

Tabella 2.1: Descrizione dei tipi di guasto

Topic	**Valore**	**Descrizione**
`failure_status/engines`	vero	quando si verifica un guasto al motore
`failure_status/aileron`	non-zero	quando si verifica un'avaria agli alettoni
	1	quando si verifica un guasto sull'alettone destro
	2	quando si verifica un guasto sull'alettone sinistro
	3	quando si verifica entrambi gli alettoni in avaria
`failure_status/rudder`	non-zero	quando si verifica un'inversione di rotta del timone
	1	quando il timone è bloccato in posizione zero
	2	quando il timone è bloccato a sinistra
	3	quando il timone è bloccato a destra.
`failure_status/elevator`	non-zero	quando si verifica un guasto all'elevatore
	1	quando l'elevatore è bloccato nella posizione zero
	2	quando è bloccato fino in fondo.

Tabella 2.2: Descrizione dei topic

Capitolo 3

Sviluppo

L'obiettivo del progetto consiste nella realizzazione di un modello di classificazione che sia capace di identificare il tipo di guasto che colpisce un dispositivo di volo, quindi un classificatore diagnostico. Il lavoro si è sviluppato in tre fasi principali:

- **fase di pre-processing**, in cui si elaborano tutti i file del dataset al fine di ottenere un unico file di lavoro;
- **fase di estrazione delle features**, in cui si applicano funzioni delle librerie messe a disposizione in Python per il calcolo delle features nel tempo ed in frequenza;
- **fase di sviluppo del classificatore**, in cui si avvia la costruzione di un modello che sia in grado di riconoscere il tipo di guasto e classificarlo in base alla label di riferimento.

3.1 Preprocessing

Inizialmente, si procede con il **caricamento dei dati** di una singola cartella rispetto alle 47 cartelle totali, cioè una per ogni volo. All'interno di questa cartella, si caricano solo i file csv utili nei DataFrame, uno per ogni file. L'elenco dei file è qui riportato:

- `"failure_status.csv"`
- `"nav_info-airspeed.csv"`
- `"nav_info-errors.csv"`
- `"nav_info-pitch.csv"`
- `"nav_info-roll.csv"`
- `"nav_info-velocity.csv"`

- "nav_info-yaw.csv"
- "local_position-velocity.csv"
- "imu-data.csv"

All'interno di ogni file, poi, sono stati utilizzati i dati indicati:

- failure_status-<name>
  - %time
  - field.data
- nav_info-airspeed
  - field.commanded
  - field.measured
- nav_info-roll
  - field.commanded
  - field.measured
- nav_info-pitch
  - field.commanded
  - field.measured
- nav_info-yaw
  - field.commanded
  - field.measured
- nav_info_velocity
  - field.des_x
  - field.des_y
  - field.des_z
  - field.meas_x
  - field.meas_y
  - field.meas_z
- nav_info_errors
  - field.alt_error
  - field.aspd_error
  - field.xtrack_error
- imu-data
  - field.orientation.x
  - field.orientation.y
  - field.orientation.z
  - field.linear_acceleration.x
  - field.linear_acceleration.y
  - field.linear_acceleration.z
- local_position-velocity
  - field.twist.angular.x
  - field.twist.angular.y
  - field.twist.angular.z
  - field.twist.linear.x
  - field.twist.linear.y
  - field.twist.linear.z

Si nota che il primo dato rappresenta la colonna '%time': questa indica il timestamp dell'acquisizione del dato nell'unità di misura dei nanosecondi, mentre il dato `field.data` indica la label di guasto. Si selezionano le informazioni inerenti all' `airspeed` ossia la

velocità del veivolo rispetto all'aria, al `roll` ossia l'oscillazione attorno al proprio asse longitudinale, al `pitch` che si riferisce all'oscillazione al proprio asse trasversale e infine `yaw` che rappresenta l'oscillazione del veivolo attorno ad un asse verticale passante per il baricentro. Di questi, sono stati presi i campi `commanded` e `measured` per calcolare il delta, ovvero la differenza tra il valore desiderato e il valore rilevato. A tal proposito, vengono ottenuti i seguenti attributi: `delta_airspeed, delta_roll, delta_pitch, delta_yaw`. Seguendo la stessa logica, sono stati calcolati anche i delta della velocità desiderata e misurata, ottenendo `delta_x, delta_y, delta_z.` I dati del `nav_info_errors` indicano informazioni sugli errori di altitudine, di airspeed e di track lungo l'asse x. Poi, sono state prese anche le informazioni rilevate dal sensore IMU relative all'accelerometro e giroscopio, in particolare i dati relativi all'orientamento del veivolo e alla sua accelerazione lineare. Infine, sono stati considerati i dati inerenti a velocità lineare e angolare di torsione.

L'attributo "`delta_airspeed`" viene successivamente rimosso perchè, analizzando il dataframe, ci si è accorti di anomalie nei valori dovute al malfunzionamento del sensore.

Partendo da questi dati, l'obiettivo è ottenere un singolo dataframe che unisca in un unico timestamp e in una frequenza comune (pari alla frequenza più alta dei dataframe in input) tutti i valori in un range di tempo di 5 centesimi di secondo. In poche parole, si vuiole standardizzare i dati in base alla frequenza di campionamento più alta.

Per fare ciò, si procede con l'**analisi delle frequenze**. Per ognuno dei dataframe ottenuti dalla cartella in considerazione, si calcola la frequenza dei dati ottenuti.

A tal proposito, si definisce una funzione "`get_frequency()`" che viene utilizzata per calcolare le frequenze di campionamento dei dati all'interno di diversi dataframes.

Per esempio, le informazioni relative alle frequenze dei file contenuti nella prima cartella in esame sono le seguenti:

La frequenza del dataframe airspeed è: 18.67 Hz.
La frequenza del dataframe velocity è: 18.67 Hz.
La frequenza del dataframe roll è: 19.12 Hz.
La frequenza del dataframe pitch è: 19.12 Hz.
La frequenza del dataframe yaw è: 19.12 Hz.
La frequenza del dataframe errors è: 19.12 Hz.
La frequenza del dataframe `local_position` è: 4.11 Hz.
La frequenza del dataframe `imu_data` è: 2.47 Hz.
La frequenza del dataframe `failure_status` è: 5.00 Hz.

Da tutto ciò, si estrae che la frequenza massima approssimata è pari a 20 Hz.

Si procede con l'operazione di **resample** per ognuno dei dataframe, portando tutti i dati alla frequenza massima di 20 Hz, cioè ogni 50 millisecondi.

Per ottenere ciò, si definisce una funzione "`resample()`" che prende un dataframe come input e restituisce un nuovo dataframe con la frequenza di campionamento desi-

derata, ossia 20Hz. Per semplicità, si converte la colonna "%time" del dataframe in un indice di data/ora utilizzando la funzione "`pd.to_datetime()`". All'interno di questa funzione di resampling, si propaga l'ultimo valore disponibile in avanti utilizzando la funzione "`ffill()`" di Pandas. Questo è necessario per evitare valori mancanti nel nuovo dataframe. Questa funzione, poi, viene applicata a tutti i dataframe prima specificati, per ottenere dataframes alla frequenza di campionamento desiderata.

Al termine di questa fase si è creato un dataframe per ogni sottocartella presente nella cartella processed del dataset originario, quindi si realizza un dataframe contenente tutte le informazioni inerenti ad un singolo volo.

3.2 Estrazione features diagnostiche

Dopo questa operazione di pre-processing, si sceglie la finestra temporale nella quale andare ad effettuare il calcolo delle features nel tempo e in frequenza: la scelta è di 1 secondo (pari a 20 campioni a frequenza 20Hz), con una sovrapposizione tra le finestre di 0.5s (step di 10 campioni a frequenza 20Hz).

Si procede con il calcolo della trasformata di Fourier e poi il calcolo delle metriche nel tempo e in frequenza indicate di seguito:

- features nel tempo:
  - Mean
  - Variance
  - MAE (Mean Absolute Error)
  - RMS (Root Mean Square)
  - RMSE (Root Mean Square Error)
  - 25° percentile
  - 50° percentile
  - 75° percentile
  - Kurtosis
  - Skewness
  - Impulse Factor
- features nella frequenza:
  - Peak Frequency
  - Peak Power
  - Band Power
  - Mean
  - 25° percentile
  - 50° percentile
  - 75° percentile

Per il calcolo sia delle features nel tempo sia di quelle in frequenza, si prendono tutti i dataframes dei voli inerenti ad una singola tipologia di guasto, e per ogni dataframe di un volo si suddivide in blocchi di 1 secondo con un overlap di 0,5 secondi; per ogni blocco di

dati risultante si calcolano le features indicate. Riguardo le features in frequenza, prima di effettuare la suddivisione in blocchi, si applica il calcolo della Trasformata di Fourier mediante la funzione `fft()`.

Tutte i nuovi dati calcolati vengono salvati in un nuovo dataframe, il quale contiene tutte le features riguardanti i diversi voli.

Durante questa operazione, si è notato che alcuni guasti avevano la stessa label. Ciò creava una situazione di ambiguità, a tal proposito, si applica un cambio del valore associato ad ogni tipo di guasto, impostandoli come mostrato nella tabella 3.1.

Fault Type	**Label**
Engine Full Power Loss	1
Rudder Stuck to Left	2
Rudder Stuck to Right	3
Elevator Stuck at Zero	4
Left Aileron Stuck at Zero	5
Right Aileron Stuck at Zero	6
Both Ailerons Stuck at Zero	7
Rudder and Aileron at Zero	8
No Fault	0

Tabella 3.1: Valori della colonna Label relativi ai guasti

Risolto questo problema di ambiguità, al fine di ottenere un unico dataframe contenente tutte le informazioni sui guasti per la fase di classificazione, si effettua il `'concat'` di tutti i dataframes finora elaborati.

3.3 Sviluppo modello diagnostico

Ottenuto il dataset completo, si inizia la fase vera e propria di sviluppo del modello di classificazione.

Anzitutto, si esegue una funzione di **normalizzazione dei dati**. Si legge il file CSV salvato nella fase precedente e si crea un DataFrame a partire dai dati contenuti nel file, si imposta l'indice del DataFrame su una colonna specifica, in questo caso %time e si rimuovono le righe che contengono valori NaN per ottenere un DataFrame pulito. Poi, si rimuove la colonna "label" per poter applicare la normalizzazione dei valori rimanenti in modo che siano compresi tra 0 e 1, producendo un nuovo DataFrame normalizzato. Infine, a questo dataframe viene aggiunta la colonna "label" precedentemente rimossa, così che il nuovo dataframe risultante sia utilizzabile per l'analisi dei dati.

Si procede con la **selezione delle features potenzialmente più utili** ai fini della classificazione. A tal proposito, si decide di applicare un ranking mediante una funzione

che utilizza il metodo di selezione delle caratteristiche "`SelectKBest`" fornito dal modulo "`sklearn.feature_selection`" di scikit-learn per selezionare le k migliori colonne. In questo caso, il metodo "`f_regression`" viene utilizzato come funzione di punteggio per valutare l'importanza delle diverse colonne. Una volta selezionate le k migliori colonne, la funzione restituisce un nuovo DataFrame contenente solo queste colonne. Eseguendo vari test, è stato ritenuto opportuno impostare il parametro K pari a 50, ottenendo le seguenti features:

- `field.orientation.y_rms`
- `field.linear_acceleration.y_rms`
- `field.twist.linear.x_mean`
- `field.twist.linear.x_p25`
- `field.twist.linear.x_p50`
- `delta_roll_mae`
- `delta_roll_rmse`
- `delta_roll_rms`
- `field.orientation.x_fft_p50`
- `field.orientation.y_fft_mean`
- `field.orientation.y_fft_p25`
- `field.orientation.y_fft_p50`
- `field.orientation.y_fft_p75`
- `field.angular_velocity.x_fft_mean`
- `field.angular_velocity.x_fft_p25`
- `field.angular_velocity.x_fft_p50`
- `field.angular_velocity.x_fft_p75`
- `field.angular_velocity.y_fft_mean`
- `field.angular_velocity.y_fft_p25`
- `field.angular_velocity.y_fft_p50`
- `field.angular_velocity.y_fft_p75`
- `field.angular_velocity.z_fft_mean`
- `field.angular_velocity.z_fft_p25`
- `field.angular_velocity.z_fft_p50`
- `field.angular_velocity.z_fft_p75`
- `field.linear_acceleration.y_fft_mean`
- `field.linear_acceleration.y_fft_p25`
- `field.linear_acceleration.y_fft_p50`
- `field.linear_acceleration.y_fft_p75`
- `field.linear_acceleration.z_fft_p25`
- `field.linear_acceleration.z_fft_p50`
- `field.linear_acceleration.z_fft_p75`
- `field.twist.angular.x_fft_mean`
- `field.twist.angular.x_fft_p25`
- `field.twist.angular.x_fft_p50`
- `field.twist.angular.x_fft_p75`
- `field.twist.angular.y_fft_mean`
- `field.twist.angular.y_fft_p25`
- `field.twist.angular.y_fft_p50`
- `field.twist.angular.y_fft_p75`
- `field.twist.angular.z_fft_mean`

- `field.twist.angular.z_fft_p25`
- `field.twist.angular.z_fft_p50`
- `field.twist.angular.z_fft_p75`
- `delta_yaw_fft_mean`
- `delta_yaw_fft_p75`
- `delta_roll_fft_mean`
- `delta_roll_fft_p25`
- `delta_roll_fft_p50`
- `delta_roll_fft_p75`

Il dataframe così ottenuto presenta dati fortemente sbilanciati, ovvero il numero di righe relative, per esempio, ai guasti del motore (engine failure) sono molte di più rispetto alle righe relative agli altri guasti. Nella tabella 3.2 si riporta il risultato del conteggio dei record associati ad ogni valore della colonna "label":

Fault	**Label**	**counts**
No fault	0	6721
Engine Full Power Loss	1	718
Rudder Stuck to Left	2	18
Rudder Stuck to Right	3	64
Elevator Stuck at Zero	4	46
Left Aileron Stuck at Zero	5	252
Right Aileron Stuck at Zero	6	319
Both Ailerons Stuck at Zero	7	71
Rudder and Aileron at Zero	8	54

Tabella 3.2: Numero di dati per ogni label di guasto

Considerando il forte sbilanciamento delle classi, sono state pensate diverse soluzioni per l'addestramento dei vari classificatori disponibili, al fine di confrontare i risultati per la scelta della configurazione migliore. Tra queste soluzioni, vi sono operazioni come *undersampling* o *oversampling*.

3.4 Addestramento modello diagnostico

Per l'addestramento del modello diagnostico, si suddivide il DataFrame in due dataset, uno per il training (85%) dei e uno per il testing (15%).

In seguito, viene definita una lista di tutti i classificatori che sono stati utilizzati per i diversi test:

- "Random Forest Classifier"
- "K-Neighbors Classifier"

- "Decision Tree Classifier"
- "Gradient Boosting Classifier"
- "Logistic Regression"
- "SVC"
- "Ada Boost Classifier"
- "MLP Classifier"

Si nota che non tutti i classificatori sopra elencati vengono utilizzati in tutti i test, in quanto la scelta del loro utilizzo dipende dalle performance ottenute.

Durante l'addestramento, viene utilizzata la tecnica della k-fold cross-validation, scegliendo 5 fold.

Inoltre, viene calcolata l'accuratezza del modello sui dati di test utilizzando la funzione `"accuracy_score()"`, la matrice di confusione per il modello utilizzando la funzione `"confusion_matrix()"` e infine viene utilizzata la funzione `"cross_val_score()"` per calcolare l'accuratezza media e la deviazione standard della cross-validation.

Capitolo 4

Risultati

Come è stato anticipato nel capitolo precedente, sono state eseguite numerose prove nella fase di addestramento. Di seguito, vengono presentati tutti i risultati ottenuti.

4.1 Undersampling

In questo test è stato effettuato un bilanciamento in base alla classe con meno elementi, ovvero la classe Rudder Left, che possiede solo 18 campioni. Il training set contiene 187 campioni, mentre il test set 25, con la seguente suddivisione:

```
NO FAULT:    6
AILERON LEFT:    5
RUDDER LEFT:    4
RUDDER ZERO:    2
BOTH AILERON:    2
RUDDER RIGHT:    2
ELEVATOR:    2
AILERON RIGHT:    1
ENGINE:    1
```

Poiché il test set risulta inevitabilmente povero di informazioni, sono stati utilizzati più classificatori per poter testare diverse configurazioni, ottenendo i seguenti risultati:

```
Accuracy: 0.68   ---> RandomForestClassifier
Accuracy: 0.6   ---> DecisionTreeClassifier
Accuracy: 0.56   ---> KNeighborsClassifier
Accuracy: 0.44   ---> SVC
Accuracy: 0.44   ---> LogisticRegression
Accuracy: 0.48   ---> MLPClassifier
```

I valori di accuracy non sono soddisfacenti. Il RandomForest sembra essere il migliore, restituendo un'accuracy comunque molto più bassa rispetto ad altri test.

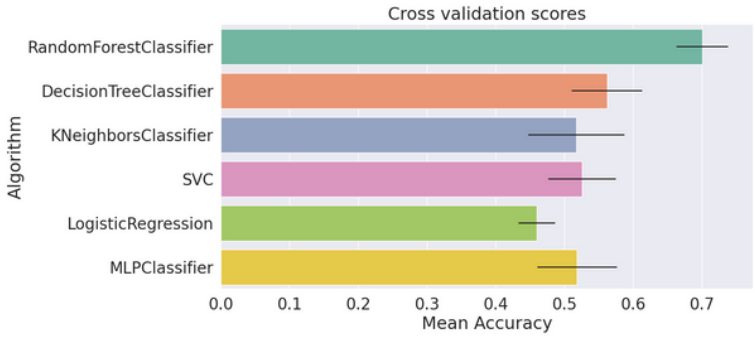

Figura 4.1: Cross Validation Scores

Analizzando le matrici di confusione, è possibile notare che i risultati non sono omogenei. La classe No Fault viene individuata quasi sempre bene, così come Rudder Left.

Figura 4.2: Matrix Confusion

Di seguito si possono analizzare tutte le diverse metriche per ogni classificatore. I risultati sono molto buoni per alcune classi, ma anche pessimi per altre classi. Questo è dovuto al fatto che il training set è molto piccolo e i classificatori non riescono ad addestrarsi bene su tutte le classi. Anche il test set, essendo molto ridotto, non riesce a rappresentare bene la realtà.

```
RandomForestClassifier Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.67      0.80         6
           1       0.00      0.00      0.00         1
           2       0.80      1.00      0.89         4
           3       1.00      1.00      1.00         2
           4       0.40      1.00      0.57         2
```

```
           5       0.67      0.80      0.73         5
           6       0.00      0.00      0.00         1
           7       0.00      0.00      0.00         2
           8       1.00      0.50      0.67         2

    accuracy                           0.68        25
   macro avg       0.54      0.55      0.52        25
weighted avg       0.69      0.68      0.66        25

DecisionTreeClassifier Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.67      0.73         6
           1       0.00      0.00      0.00         1
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         2
           4       0.67      1.00      0.80         2
           5       0.50      0.60      0.55         5
           6       0.00      0.00      0.00         1
           7       0.00      0.00      0.00         2
           8       0.00      0.00      0.00         2

    accuracy                           0.60        25
   macro avg       0.44      0.47      0.45        25
weighted avg       0.59      0.60      0.59        25

KNeighborsClassifier Classification Report:
              precision    recall  f1-score   support

           0       0.40      0.33      0.36         6
           1       0.00      0.00      0.00         1
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         2
           4       0.50      1.00      0.67         2
           5       1.00      0.40      0.57         5
           6       0.00      0.00      0.00         1
           7       0.00      0.00      0.00         2
           8       0.50      1.00      0.67         2

    accuracy                           0.56        25
   macro avg       0.49      0.53      0.47        25
```

```
weighted avg       0.62      0.56      0.55        25

SVC Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         6
           1       0.50      1.00      0.67         1
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         2
           4       0.29      1.00      0.44         2
           5       0.67      0.40      0.50         5
           6       0.00      0.00      0.00         1
           7       0.00      0.00      0.00         2
           8       0.00      0.00      0.00         2

    accuracy                           0.44        25
   macro avg       0.38      0.49      0.40        25
weighted avg       0.42      0.44      0.40        25

LogisticRegression Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         6
           1       0.00      0.00      0.00         1
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         2
           4       0.40      1.00      0.57         2
           5       0.67      0.40      0.50         5
           6       0.00      0.00      0.00         1
           7       0.00      0.00      0.00         2
           8       0.33      0.50      0.40         2

    accuracy                           0.44        25
   macro avg       0.38      0.43      0.39        25
weighted avg       0.43      0.44      0.42        25

MLPClassifier Classification Report:
              precision    recall  f1-score   support

           0       0.25      0.17      0.20         6
           1       0.00      0.00      0.00         1
```

```
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         2
           4       0.50      0.50      0.50         2
           5       0.75      0.60      0.67         5
           6       0.00      0.00      0.00         1
           7       0.00      0.00      0.00         2
           8       0.50      0.50      0.50         2

    accuracy                           0.48        25
   macro avg       0.44      0.42      0.43        25
weighted avg       0.53      0.48      0.50        25
```

4.2 Classi più rappresentate

In questo test sono stati addestrati i classificatori sulle classi più popolose, al fine di valutare al meglio la bontà di classificazione. In particolare, sono state selezionate le seguenti classi: No Fault, Engine, Aileron Left, Aileron Right. Prima dell'addestramento, è stato effettuato un resampling in base alla classe meno popolosa, ovvero Aileron Left (252 campioni), ottenendo pertanto un training set di dimensione 856 e un training set con 152 campioni. Di seguito la suddivisione del test set:

```
RIGHT AILERON:    41
NO FAULT:    39
LEFT AILERON:    37
ENGINE:    35
```

Si vuole far presente che in questa classificazione, è stato aggiunto un parametro di Shuffle nella K-Fold Cross Validation, ma confrontando i risultati con quelli ottenuti senza shuffle, non si sono riscontrati netti miglioramenti.

Di seguito i risultati della classificazione:

```
Accuracy: 0.86   ---> RandomForestClassifier
Accuracy: 0.75   ---> DecisionTreeClassifier
Accuracy: 0.69   ---> KNeighborsClassifier
Accuracy: 0.54   ---> SVC
Accuracy: 0.47   ---> LogisticRegression
Accuracy: 0.62   ---> MLPClassifier
```

Anche qui il RandomForest sembra essere il migliore.

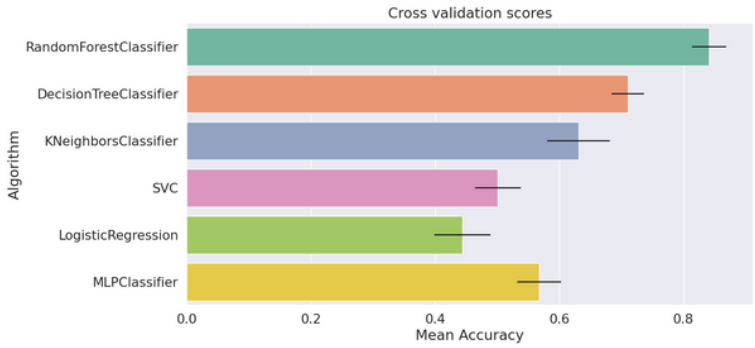

Figura 4.3: Cross Validation Scores

Dalle matrici di confusione è possibile notare come la maggior parte dei true positive viene correttamente identificata, mostrando quindi, complessivamente, una buona classificazione.

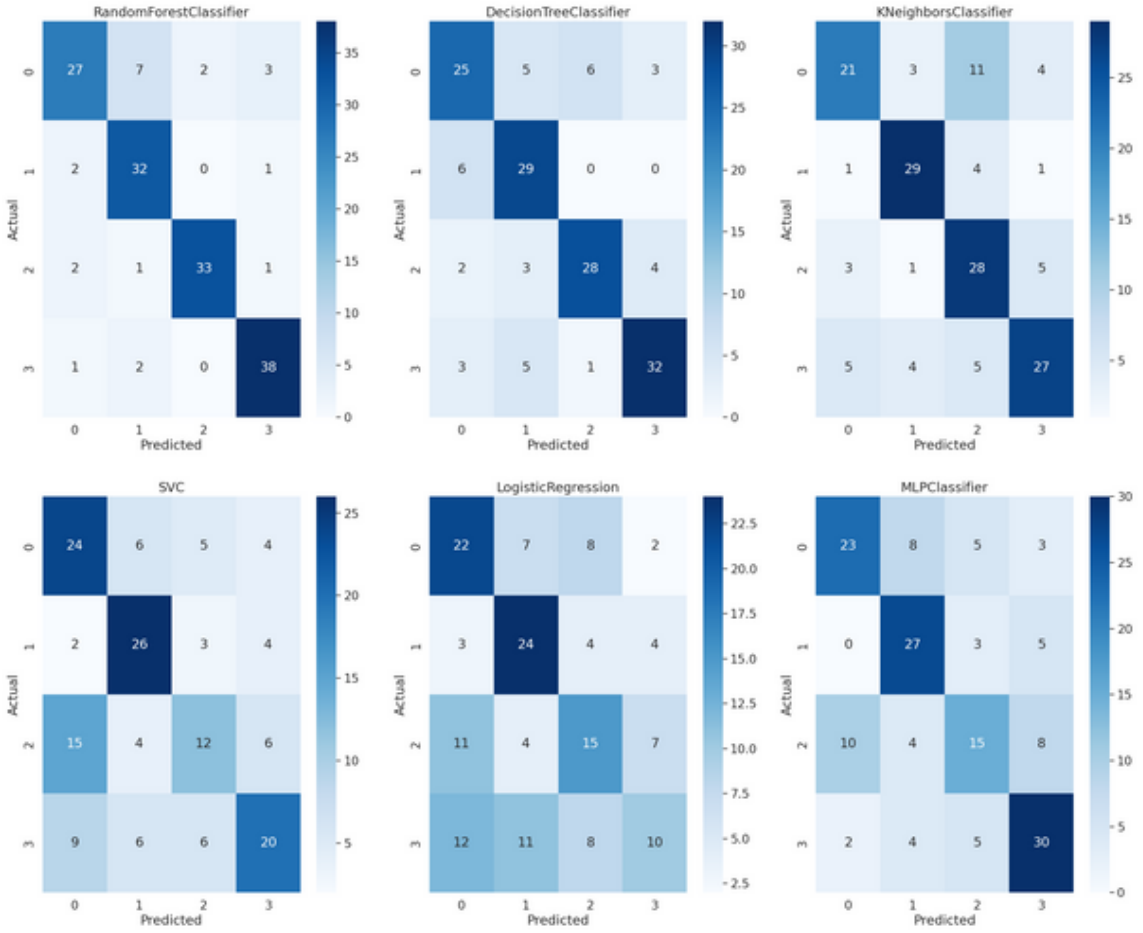

Figura 4.4: Matrix Confusion

Nei risultati seguenti è possibile confermare la bontà dei risultati ottenuti. Soprattutto per quanto riguarda il classificatore Random Forest, tutte le metriche sembrano avere valori molto buoni. Anche per i guasti all'ala destra e sinistra ci sono risultati soddisfacenti.

```
RandomForestClassifier Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.69      0.76        39
           1       0.76      0.91      0.83        35
           5       0.94      0.89      0.92        37
           6       0.88      0.93      0.90        41

    accuracy                           0.86       152
```

```
   macro avg       0.86      0.86      0.85       152
weighted avg       0.86      0.86      0.85       152

DecisionTreeClassifier Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.64      0.67        39
           1       0.69      0.83      0.75        35
           5       0.80      0.76      0.78        37
           6       0.82      0.78      0.80        41

    accuracy                           0.75       152
   macro avg       0.75      0.75      0.75       152
weighted avg       0.75      0.75      0.75       152

KNeighborsClassifier Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.54      0.61        39
           1       0.78      0.83      0.81        35
           5       0.58      0.76      0.66        37
           6       0.73      0.66      0.69        41

    accuracy                           0.69       152
   macro avg       0.70      0.70      0.69       152
weighted avg       0.70      0.69      0.69       152

SVC Classification Report:
              precision    recall  f1-score   support

           0       0.48      0.62      0.54        39
           1       0.62      0.74      0.68        35
           5       0.46      0.32      0.38        37
           6       0.59      0.49      0.53        41

    accuracy                           0.54       152
   macro avg       0.54      0.54      0.53       152
weighted avg       0.54      0.54      0.53       152

LogisticRegression Classification Report:
              precision    recall  f1-score   support
```

```
              0       0.46      0.56      0.51        39
              1       0.52      0.69      0.59        35
              5       0.43      0.41      0.42        37
              6       0.43      0.24      0.31        41

    accuracy                            0.47       152
   macro avg       0.46      0.47      0.46       152
weighted avg       0.46      0.47      0.45       152

MLPClassifier Classification Report:
              precision    recall  f1-score   support

              0       0.66      0.59      0.62        39
              1       0.63      0.77      0.69        35
              5       0.54      0.41      0.46        37
              6       0.65      0.73      0.69        41

    accuracy                            0.62       152
   macro avg       0.62      0.62      0.62       152
weighted avg       0.62      0.62      0.62       152
```

4.3 Sampling di no_fault

```
Accuracy: 0.55   ---> LogisticRegression
Accuracy: 0.74   ---> DecisionTreeClassifier
Accuracy: 0.62   ---> SVC
Accuracy: 0.79   ---> RandomForestClassifier
```

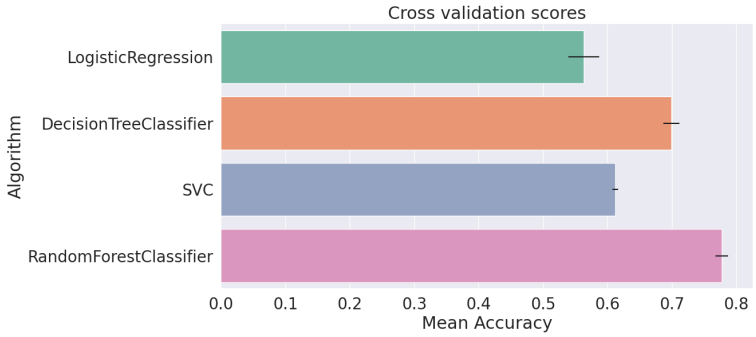

Figura 4.5: Cross Validation Scores

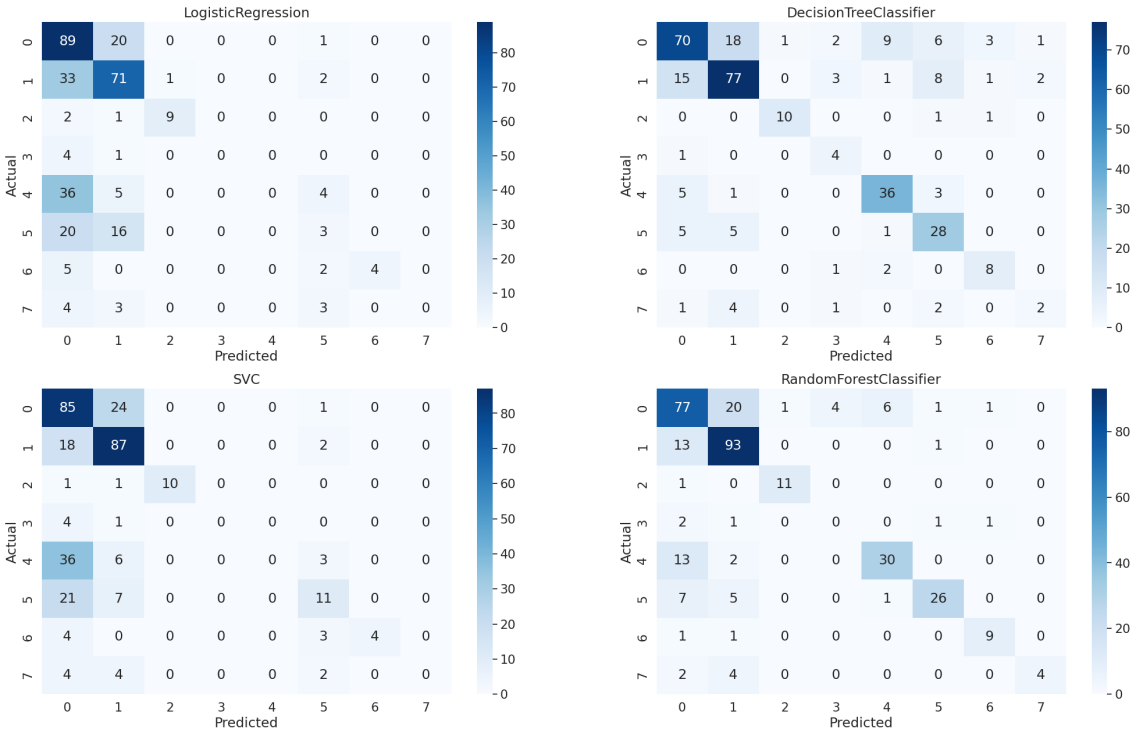

Figura 4.6: Matrix Confusion

4.4 Classificazione senza bilanciamento

Il primo test è stato eseguito con l'addestramento dei classificatori senza effettuare il bilanciamento delle classi. Il training set è costituito da 7023 campioni. Il test set è invece costituito da 1240 campioni, così suddivisi:

```
NO FAULT: 1010
ENGINE: 101
AILERON RIGHT: 50
AILERON LEFT: 38
RUDDER ZERO: 11
AILERON BOTH: 9
RUDDER RIGHT: 8
ELEVATOR:  8
RUDDER LEFT: 5
```

Di seguito vengono riportati i valori di accuracy ottenuti dai diversi classificatori con K-Fold Cross validation:

```
Accuracy: 0.9   ---> RandomForestClassifier
```

```
Accuracy: 0.89   ---> KNeighborsClassifier
Accuracy: 0.85   ---> DecisionTreeClassifier
```

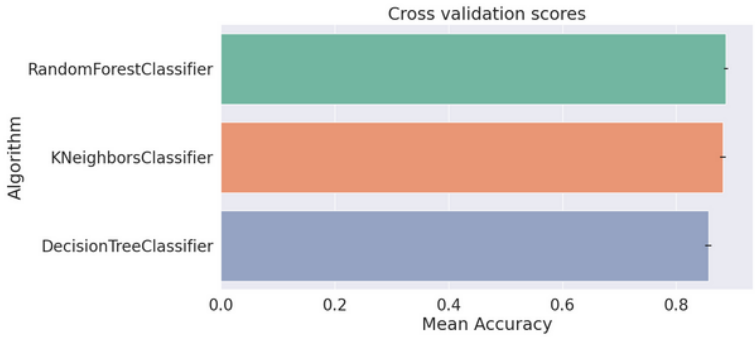

Figura 4.7: Cross Validation Scores

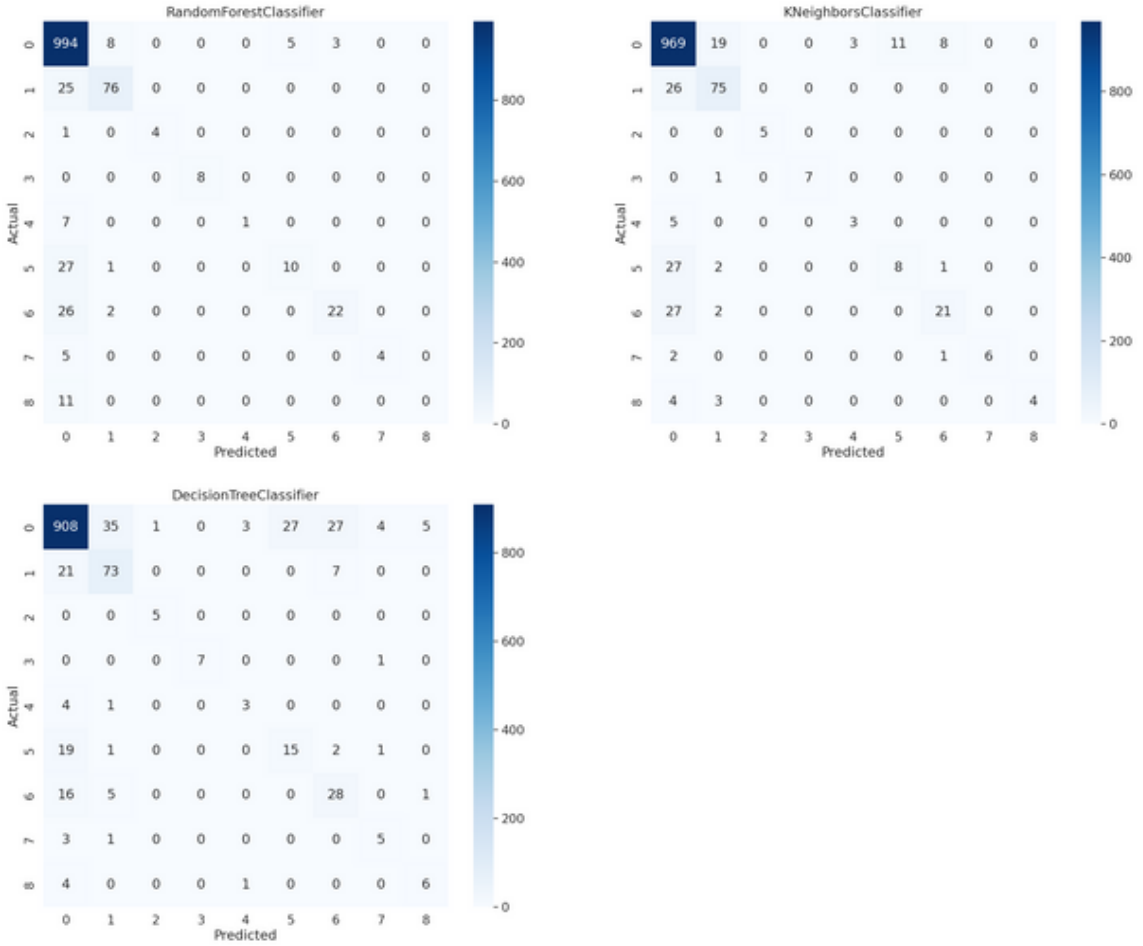

Figura 4.8: Matrix Confusion

I risultati di accuracy sembrano essere ottimi. Il classificatore RandomForest è quello che ottiene il punteggio di accuracy più elevato. Analizzando nel dettaglio le matrici di confusione ottenute, si nota subito come i risultati sul test set sono ottimi specialmente per quanto riguarda la classe No Fault. Tuttavia, si hanno risutati buoni anche per le altre classi. Infatti, di seguito si possono visualizzare i punteggi di precision, recall, f1-score e supporto, oltre che accuracy, macro average e weighted average. Analizzando il classificatore migliore (RandomForest), si nota come per le classi 2, 3, 4 e 7 la precision è pari 1, quindi tutti i true positive vengono individuati correttamente. La classe 8, tuttavia, inerente alla rottura di entrambe le ali del veivolo, ha precision 0: infatti, nessun elemento appartenente a questa classe viene individuato come true positive. Il classificatore KNeighbors, invece, individua correttamente tutti gli elementi della classe 8, restituendo però valori peggiori complessivamente. Vediamo comunque che per tutte le metriche, i valori migliori sono ottenuti per la classe 0: questo risultato è giustificato

dal fatto che è la classe più popolosa.

RandomForestClassifier Classification Report:

	precision	recall	f1-score	support
0	0.91	0.98	0.94	1010
1	0.87	0.75	0.81	101
2	1.00	0.80	0.89	5
3	1.00	1.00	1.00	8
4	1.00	0.12	0.22	8
5	0.67	0.26	0.38	38
6	0.88	0.44	0.59	50
7	1.00	0.44	0.62	9
8	0.00	0.00	0.00	11
accuracy			0.90	1240
macro avg	0.81	0.53	0.60	1240
weighted avg	0.89	0.90	0.89	1240

KNeighborsClassifier Classification Report:

	precision	recall	f1-score	support
0	0.91	0.96	0.94	1010
1	0.74	0.74	0.74	101
2	1.00	1.00	1.00	5
3	1.00	0.88	0.93	8
4	0.50	0.38	0.43	8
5	0.42	0.21	0.28	38
6	0.68	0.42	0.52	50
7	1.00	0.67	0.80	9
8	1.00	0.36	0.53	11
accuracy			0.89	1240
macro avg	0.81	0.62	0.69	1240
weighted avg	0.87	0.89	0.88	1240

DecisionTreeClassifier Classification Report:

	precision	recall	f1-score	support
0	0.93	0.90	0.91	1010
1	0.63	0.72	0.67	101

```
              2       0.83      1.00      0.91         5
              3       1.00      0.88      0.93         8
              4       0.43      0.38      0.40         8
              5       0.36      0.39      0.38        38
              6       0.44      0.56      0.49        50
              7       0.45      0.56      0.50         9
              8       0.50      0.55      0.52        11

    accuracy                            0.85      1240
   macro avg       0.62      0.66      0.64      1240
weighted avg       0.86      0.85      0.85      1240
```

4.5 SMOTE con Sampling di No_Fault

In questo test è stato prima effettuato un resampling della classe No Fault, portando il numero di campioni pari al numero di elementi della classe Engine. Dopodiché è stato applicata la tecnica di SMOTE che ha lo scopo di effettuare un oversampling delle classi meno popolose, portando il numero di campioni pari a quello della classe più numerosa. In questo modo, si ottiene un training set con ben 5499 campioni, dove ogni classe possiede 611 elementi. Il test set ha, invece, dimensione pari a 339, con la seguente suddivisione:

```
NO FAULT:   110
ENGINE:    107
AILERON LEFT:     45
AILERON RIGHT:     39
RUDDER RIGHT:     12
BOTH AILERON:     11
RUDDER ZERO:     10
ELEVATOR:      5
```

Si ottengono i seguenti risultati:

```
Accuracy: 0.83   ---> RandomForestClassifier
Accuracy: 0.71   ---> DecisionTreeClassifier
Accuracy: 0.7   ---> KNeighborsClassifier
Accuracy: 0.52   ---> SVC
Accuracy: 0.41   ---> LogisticRegression
Accuracy: 0.59   ---> MLPClassifier
```

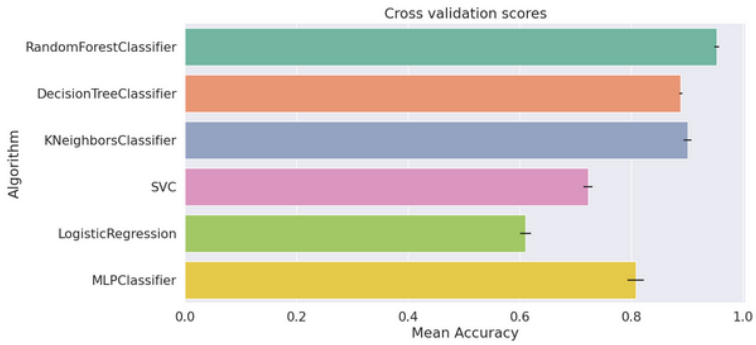

Figura 4.9: Cross Validation Scores

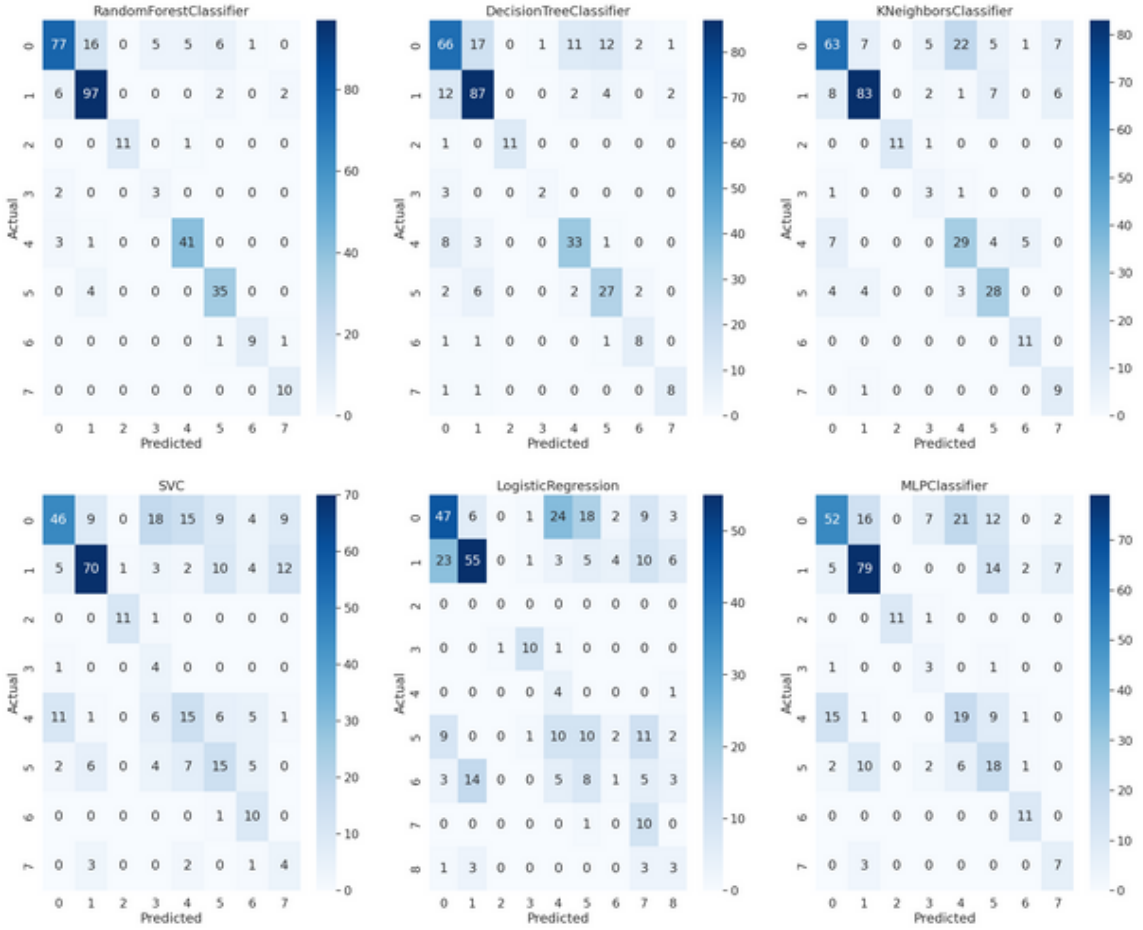

Figura 4.10: Matrix Confusion

Di seguito è possibile analizzare tutte le metriche:

RandomForestClassifier Classification Report:

	precision	recall	f1-score	support
0	0.88	0.70	0.78	110
1	0.82	0.91	0.86	107
3	1.00	0.92	0.96	12
4	0.38	0.60	0.46	5
5	0.87	0.91	0.89	45
6	0.80	0.90	0.84	39
7	0.90	0.82	0.86	11
8	0.77	1.00	0.87	10
accuracy			0.83	339
macro avg	0.80	0.84	0.81	339
weighted avg	0.84	0.83	0.83	339

DecisionTreeClassifier Classification Report:

	precision	recall	f1-score	support
0	0.70	0.60	0.65	110
1	0.76	0.81	0.78	107
3	1.00	0.92	0.96	12
4	0.67	0.40	0.50	5
5	0.69	0.73	0.71	45
6	0.60	0.69	0.64	39
7	0.67	0.73	0.70	11
8	0.73	0.80	0.76	10
accuracy			0.71	339
macro avg	0.73	0.71	0.71	339
weighted avg	0.72	0.71	0.71	339

KNeighborsClassifier Classification Report:

	precision	recall	f1-score	support
0	0.76	0.57	0.65	110
1	0.87	0.78	0.82	107
3	1.00	0.92	0.96	12
4	0.27	0.60	0.37	5
5	0.52	0.64	0.57	45
6	0.64	0.72	0.67	39

```
           7       0.65      1.00      0.79        11
           8       0.41      0.90      0.56        10

    accuracy                           0.70       339
   macro avg       0.64      0.77      0.68       339
weighted avg       0.74      0.70      0.71       339

SVC Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.42      0.53       110
           1       0.79      0.65      0.71       107
           3       0.92      0.92      0.92        12
           4       0.11      0.80      0.20         5
           5       0.37      0.33      0.35        45
           6       0.37      0.38      0.37        39
           7       0.34      0.91      0.50        11
           8       0.15      0.40      0.22        10

    accuracy                           0.52       339
   macro avg       0.47      0.60      0.47       339
weighted avg       0.62      0.52      0.54       339

LogisticRegression Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.43      0.49       110
           1       0.71      0.51      0.59       107
           2       0.00      0.00      0.00         0
           3       0.77      0.83      0.80        12
           4       0.09      0.80      0.15         5
           5       0.24      0.22      0.23        45
           6       0.11      0.03      0.04        39
           7       0.21      0.91      0.34        11
           8       0.17      0.30      0.21        10

    accuracy                           0.41       339
   macro avg       0.32      0.45      0.32       339
weighted avg       0.49      0.41      0.43       339

MLPClassifier Classification Report:
```

	precision	recall	f1-score	support
0	0.69	0.47	0.56	110
1	0.72	0.74	0.73	107
3	1.00	0.92	0.96	12
4	0.23	0.60	0.33	5
5	0.41	0.42	0.42	45
6	0.33	0.46	0.39	39
7	0.73	1.00	0.85	11
8	0.44	0.70	0.54	10
accuracy			0.59	339
macro avg	0.57	0.66	0.60	339
weighted avg	0.62	0.59	0.60	339

In riferimento al miglior classificatore (Random Forest), rispetto alla classificazione effettuata senza alcun bilanciamento (Paragrafo 4.4), la precision fornisce risultati leggermente peggiori. Tuttavia, recall e f1-score sono migliori in questo test. Infatti, mentre nell'altro test i risultati sono più sbilanciati, qui si ottiene più omogeneità.

4.6 SMOTE senza Sampling di No_Fault

In questo test, viene applicata la tecnica SMOTE senza effettuare l'undersampling della classe No Fault. Tutte le classi, quindi, vengono portate ad una numerosità pari a quella di No Fault. Il training set risulta avere una dimensione di 51399 campioni, mentre il test set possiede 1240 campioni.

```
NO FAULT:    1010
ENGINE:     101
AILERON RIGHT:      50
AILERON LEFT:      38
RUDDER ZERO:      11
BOTH AILERON:       9
RUDDER RIGHT:       8
ELEVATOR:       8
RUDDER LEFT:       5
```

In questo test sono stati addestrati meno classificatori, in quanto, essendo il training set molto ampio, i costi computazionali sono cresciuti di molto rispetto agli altri test e non è stato possibile addestrare altri classificatori in tempi ragionevoli.

```
Accuracy: 0.91   ---> RandomForestClassifier
Accuracy: 0.77   ---> KNeighborsClassifier
Accuracy: 0.84   ---> DecisionTreeClassifier
```

I risultati di accuracy sono molto buoni. Come sempre, il Random Forest è il classificatore che ottiene il punteggio migliore di tutti.

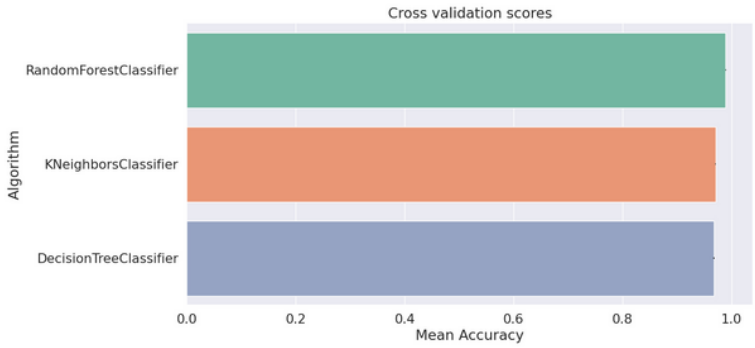

Figura 4.11: Cross Validation Scores

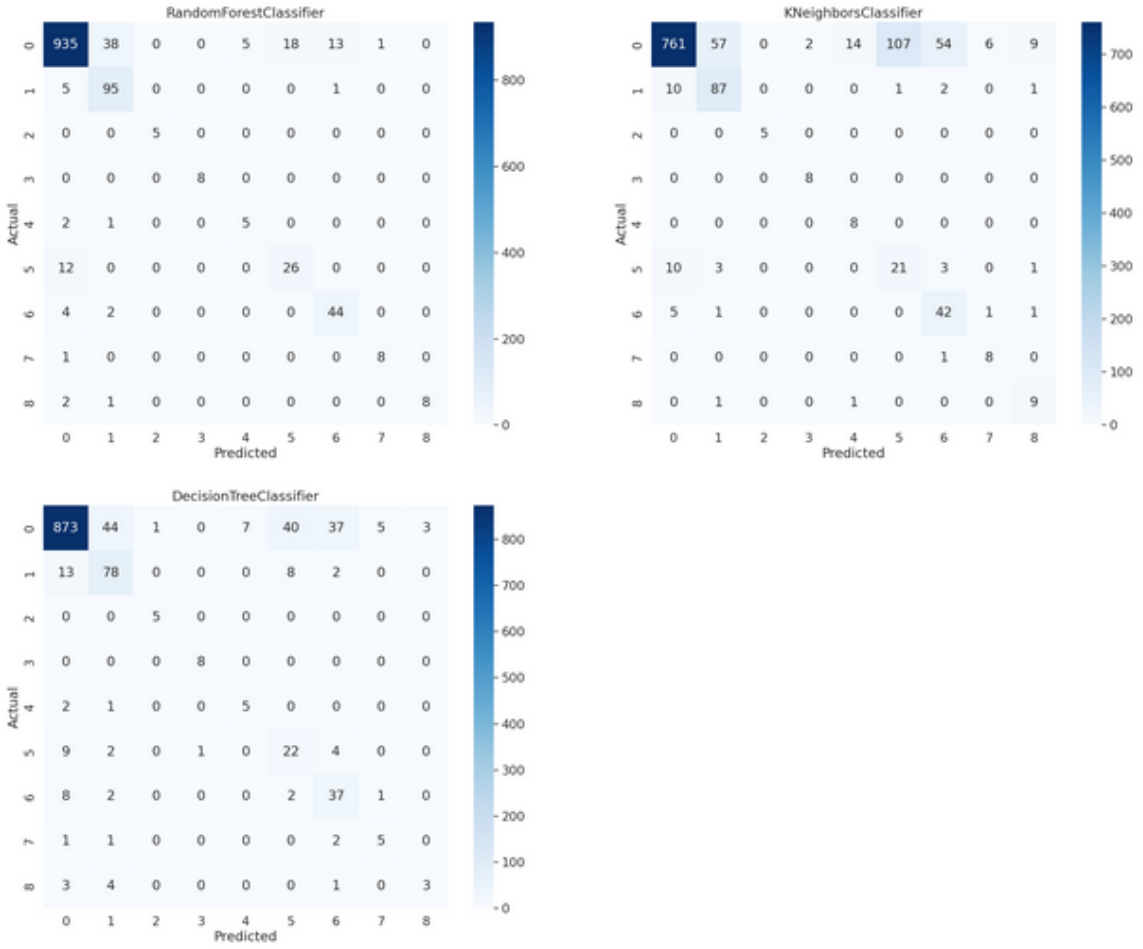

Figura 4.12: Matrix Confusion

RandomForestClassifier Classification Report:

	precision	recall	f1-score	support
0	0.97	0.93	0.95	1010
1	0.69	0.94	0.80	101
2	1.00	1.00	1.00	5
3	1.00	1.00	1.00	8
4	0.50	0.62	0.56	8
5	0.59	0.68	0.63	38
6	0.76	0.88	0.81	50
7	0.89	0.89	0.89	9
8	1.00	0.73	0.84	11

```
    accuracy                           0.91      1240
   macro avg       0.82      0.85      0.83      1240
weighted avg       0.93      0.91      0.92      1240

KNeighborsClassifier Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.75      0.85      1010
           1       0.58      0.86      0.70       101
           2       1.00      1.00      1.00         5
           3       0.80      1.00      0.89         8
           4       0.35      1.00      0.52         8
           5       0.16      0.55      0.25        38
           6       0.41      0.84      0.55        50
           7       0.53      0.89      0.67         9
           8       0.43      0.82      0.56        11

    accuracy                           0.77      1240
   macro avg       0.58      0.86      0.66      1240
weighted avg       0.88      0.77      0.80      1240

DecisionTreeClassifier Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.86      0.91      1010
           1       0.59      0.77      0.67       101
           2       0.83      1.00      0.91         5
           3       0.89      1.00      0.94         8
           4       0.42      0.62      0.50         8
           5       0.31      0.58      0.40        38
           6       0.45      0.74      0.56        50
           7       0.45      0.56      0.50         9
           8       0.50      0.27      0.35        11

    accuracy                           0.84      1240
   macro avg       0.60      0.71      0.64      1240
weighted avg       0.88      0.84      0.85      1240
```

Dai risultati, sempre in riferimento al Random Forest,

Capitolo 5

Problematiche riscontrate

Durante la fase di sviluppo del modello, nello specifico nell'operazione di ranking per la selezione delle features più significative ai fini della classificazione, sono stati scontrati problemi riguardanti i valori 'NaN', infatti la funzione `selectKbest` non può essere applicata a dataframe che contengono valori 'NaN'. Questi valori sono stati il risultato dell'esecuzione di ???

Questo ci ha costretti ad introdurre l'istruzione `dropna()` che è volta ad eliminare le righe con tali valori.

Durante la fase di addestramento del modello di classificazione, il problema principale è lo sbilanciamento nella cardinalità delle classi dei guasti. Però, questo problema trova origine proprio nella struttura del dataset di partenza e nella natura del problema in quanto i guasti vengono iniettati durante la fase di volo del dispositivo.

...

Capitolo 6

Conclusioni e sviluppi futuri

Bibliografia

[1] Azarakhsh Keipour, Mohammadreza Mousaei e Sebastian Scherer. «ALFA: A dataset for UAV fault and anomaly detection». In: *The International Journal of Robotics Research* 40.2-3 (2021), pp. 515–520. DOI: 10.1177/0278364920966642.