Rilevamento dell'usura pala a partire dall'analisi del log di volo di un drone esarotore

Nicolò Bartolini
Matr. 1118768
s1118768@studenti.univpm.it

Nicola Picciafuoco
Matr. 1118755
s1118755@studenti.univpm.it

1. Introduzione

L'avvento dei droni multirotori ha rivoluzionato numerosi settori, dall'agricoltura alla logistica, offrendo soluzioni innovative per operazioni complesse in ambienti dinamici. Tuttavia, l'affidabilità e la sicurezza di tali sistemi dipendono fortemente dalla capacità di rilevare e diagnosticare tempestivamente eventuali guasti agli attuatori, in particolare ai componenti critici come le pale delle eliche.

Il presente progetto si propone di sviluppare un sistema di rilevamento dell'usura delle pale a partire dall'analisi dei log di volo di un drone esarotore. Utilizzando un dataset realizzato appositamente, l'obiettivo è discriminare tra voli effettuati con pale nuove, con usura del 5% e con usura del 10%. In questo contesto, sono stati sviluppati diversi modelli di classificazione per affrontare in maniera completa il problema diagnostico. In particolare, sono stati realizzati:

- Un classificatore binario (guasto/no-guasto) a 2 classi.
- Un classificatore multiclasse a 3 classi per tipo di guasto (no-guasto, 5%, 10%).
- Un classificatore multiclasse a 7 classi per isolamento (no-guasto e guasto localizzato al motore i-esimo).
- Per estensione, un classificatore multiclasse a 13 classi (no-guasto e guasto specifico, differenziando tra usura del 5% e del 10% per ciascun motore).

L'approccio seguito si basa su una pipeline integrata che comprende la sincronizzazione e il pre-processing dei dati grezzi, l'estrazione di feature diagnostiche mediante il Diagnostic Feature Designer (DFD) di MATLAB e l'addestramento di modelli di classificazione. L'impiego di tecniche di feature selection, come l'ANOVA, ha consentito di ridurre lo spazio delle feature, ottenendo prestazioni elevate (fino al 96.9% di accuratezza) anche con un sottoinsieme ottimale delle feature.

La struttura della presente relazione è organizzata come segue: la Sezione 2 descrive il dataset e le procedure di pre-elaborazione, la Sezione 3 illustra il processo di estrazione delle feature con il DFD, la Sezione 4 presenta i risultati ottenuti dai modelli di classificazione, e infine la Sezione 5 offre una discussione critica dei risultati e le conclusioni finali.

2. Descrizione del dataset e pre-processamento

In questa sezione si descrive nel dettaglio il dataset utilizzato e le procedure di pre-elaborazione adottate per ottenere un insieme di segnali uniformati e pronti per l'estrazione delle feature diagnostiche.

2.1. Descrizione del dataset

Il dataset impiegato per il progetto è stato acquisito mediante registrazione dei log di volo di un drone esarotore e scaricato dal repository Zenodo consultabile al link https://zenodo.org/records/7648996. Esso è composto da 18 file con estensione `.mat`, ciascuno corrispondente a un volo registrato in condizioni operative differenti:

- 6 voli con condizioni normali (no guasto).
- 6 voli in cui una pala presenta un'usura del 5%.
- 6 voli in cui una pala presenta un'usura del 10%.

Ognuno dei file relativi ai voli con pala usurata era indicativo di un guasto presente in un motore specifico (es. il file `FAULT_M4_5.mat` è relativo a una pala usurata al 5% applicata al motore 4).

I dati grezzi acquisiti comprendono informazioni provenienti da diversi sensori e sottosistemi del drone, quali:

- **IMU:** dati di accelerazione e velocità angolare provenienti da tre unità (IMU_0, IMU_1, IMU_2).
- **ESC:** informazioni relative alla velocità (RPM) e alla corrente assorbita da ciascuno dei 6 motori (ESC_0 – ESC_5).
- **PWM:** segnali di controllo inviati ai motori, registrati nelle colonne 11–16 della matrice RCOU.
- **Attitude:** dati relativi all'assetto (roll, pitch, yaw) e agli angoli desiderati, contenuti nella matrice ATT.
- **VIBE:** informazioni sulle vibrazioni, fornite da tre set di dati (VIBE_0, VIBE_1, VIBE_2).
- **RATE**: il segnale rate rappresenta i tassi di variazione degli assetti o, più in generale, la velocità di variazione dell'assetto (RATE).

Questa molteplicità di fonti consente di avere una visione completa del comportamento dinamico del drone, elemento essenziale per rilevare eventuali anomalie dovute all'usura delle pale.

2.2. Acquisizione e sincronizzazione dei dati

Il primo step della pipeline ha riguardato la sincronizzazione dei dati, necessaria per poter confrontare e analizzare i segnali provenienti da diverse fonti con frequenze di campionamento differenti. I principali passaggi sono stati:

- **Allineamento dei dati IMU e VIBE:** utilizzando una funzione apposita (`mergeSignals`), sono stati confrontati e allineati i dati provenienti dai tre sensori IMU (IMU_0, IMU_1 e IMU_2) e dai tre set di dati VIBE (VIBE_0, VIBE_1 e VIBE_2). Successivamente, è stata calcolata la media dei segnali di accelerazione (colonne 7–9) e giroscopici (colonne 4–6) per ottenere una rappresentazione unificata del comportamento dinamico del sistema.
- **Estrazione di ulteriori segnali:** oltre ai dati IMU, sono stati estratti e sincronizzati altri segnali critici:
  - **PWM:** i segnali di controllo, ottenuti dalle colonne 11–16 della matrice RCOU.
  - **ESC e CURR:** per ciascuno dei 6 motori, sono stati estratti i valori relativi a RPM (colonna 4) e alla corrente (colonna 7) mediante un ciclo che utilizza la funzione `eval`.
  - **Attitude:** gli angoli desiderati e misurati (roll, pitch, yaw) sono stati estratti dalla matrice ATT.
  - **RATE:** i tassi di variazione degli angoli sono stati estratti a partire dalla matrice RATE (escludendo le colonne non rilevanti).
- **Estrazione e normalizzazione dei timestamp:** per ciascun gruppo di sensori (IMU, PWM, ESC, VIBE, RATE, Attitude) sono stati estratti i timestamp, che sono stati convertiti in secondi. Il tempo iniziale t_0 è stato determinato come il minimo tra i primi campioni di tutti i gruppi, e successivamente tutti i vettori temporali sono stati shiftati in modo che il primo campione corrispondesse a $t = 0$. Questa operazione ha garantito un corretto allineamento temporale dei segnali.
- **Creazione di un vettore temporale uniforme:** una volta normalizzati i tempi, è stato definito un vettore temporale uniforme con una frequenza di campionamento di riferimento pari a 350 Hz. Questo vettore, generato tramite la formula $t = 0 : \frac{1}{\mathrm{FsRef}} : \mathrm{tEnd}$, è stato impiegato per uniformare i segnali mediante la tecnica Zero-Order-Hold (ZOH). Tale approccio ha garantito la comparabilità dei dati provenienti da differenti sensori e la coerenza necessaria per le fasi successive di estrazione delle feature.
- **Flessibilità del metodo di sincronizzazione:** il codice implementa attualmente la tecnica ZOH per la sincronizzazione, ma è stato strutturato in modo modulare, permettendo l'eventuale integrazione di altri metodi (ad es. interpolazione lineare) in futuro, a seconda delle necessità sperimentali.

Questi passaggi hanno costituito la base solida per le successive fasi di estrazione delle feature diagnostiche e per l'addestramento dei modelli di classificazione.

2.3. Pre-elaborazione e pulizia dei dati

Per garantire la qualità dei dati in ingresso alle fasi successive (estrazione delle feature e classificazione), sono state eseguite le seguenti operazioni di pre-elaborazione:

- **Rimozione delle fasi di decollo e atterraggio:** utilizzando il segnale PWM, è stata applicata una soglia a 1500 (basata sulla somma dei canali) per eliminare i campioni relativi alle fasi di decollo e atterraggio, durante le quali il comportamento del drone è significativamente diverso e può introdurre distorsioni nelle feature.
- **Taglio delle estremità temporali:** per ogni volo, sono stati rimossi i primi 2 secondi e gli ultimi 2 secondi di registrazione, focalizzandosi così sulla fase di volo stabile e garantendo la coerenza dell'analisi.

2.4. Preparazione per l'estrazione delle feature

Dopo il pre-processing, i segnali sono stati organizzati in strutture dati (timetable) per ciascuna variabile rilevante, quali:

- **ACC**: (`accX`, `accY`, `accZ`)
- **GYR** (`gyrX`, `gyrY`, `gyrZ`)
- **PWM** (`pwm1–pwm6`)
- **ESC e CURR** (`esc1–esc6`, `curr1–curr6`)
- **Attitude** (`roll`, `pitch`, `yaw`, `des_roll`, `des_pitch`, `des_yaw`)
- **VIBE** (`vibe_x`, `vibe_y`, `vibe_z`)
- **RATE** (`rate`)

Queste strutture sono state poi salvate in una tabella (`dataTable`), alla quale è stato associato il relativo faultCode (in base al tipo di classificazione) a seconda della condizione della pala nel volo. Tale tabella rappresenta l'input per il Diagnostic Feature Designer (DFD), consentendo l'estrazione di feature diagnostiche in dominio temporale e frequenziale.

2.5. Considerazioni finali

L'approccio di pre-elaborazione adottato ha permesso di ottenere un dataset pulito e coerente, fondamentale per l'estrazione di feature affidabili. La fase di sincronizzazione, in particolare, ha garantito che tutti i segnali fossero allineati temporalmente, mentre la rimozione delle fasi non rappresentative (decollo e atterraggio) ha migliorato la qualità dei dati in ingresso al DFD.

Questa preparazione accurata ha costituito la base per il successo dei modelli di classificazione, che sono stati successivamente addestrati su feature derivate sia in dominio temporale che frequenziale, raggiungendo elevati livelli di accuratezza nella diagnosi del guasto.

3. Estrazione delle feature attraverso il Diagnostic Feature Designer

In questa sezione viene descritto in maniera approfondita il processo di estrazione delle feature diagnostiche utilizzato per trasformare i segnali sincronizzati e pre-elaborati in un insieme di variabili numeriche informative. Tali feature costituiscono l'input per il successivo addestramento dei modelli di classificazione, che mirano a discriminare tra voli con condizioni normali e quelli con usura delle pale. Di seguito vengono illustrati, passo dopo passo, tutti i passaggi eseguiti all'interno del Diagnostic Feature Designer (DFD).

3.1. Importazione e configurazione iniziale

Il primo passo è stato l'apertura del DFD in ambiente MATLAB, seguito dall'importazione della data table `dataTable` che contiene tutte le timeTable generate durante la fase di sincronizzazione e pre-elaborazione. Questa tabella include, per ciascun volo, i segnali relativi a:

- **IMU:** accelerazioni (`accX`, `accY`, `accZ`) e velocità angolari (`gyrX`, `gyrY`, `gyrZ`).
- **PWM:** segnali di controllo inviati ai motori (`pwm1–pwm6`).
- **ESC e CURR:** dati relativi alle velocità (`esc1–esc6`) e alle correnti (`curr1–curr6`) dei motori.
- **Attitude:** angoli di assetto (`roll`, `pitch`, `yaw`) e gli angoli desiderati (`des_roll`, `des_pitch`, `des_yaw`).
- **VIBE:** informazioni sulle vibrazioni (`vibe_x`, `vibe_y`, `vibe_z`).
- **RATE**: tassi di variazione degli assetti.

Ogni record della tabella è corredato dal relativo `faultCode` che identifica la condizione del volo.

3.1.1. Impostazione del `faultCode`

Il suddetto `faultCode` è stato impostato in base ai diversi scopi della classificazione nel seguente modo:

- Classificazione binaria: il `faultCode` può essere pari a 0 (no-guasto) oppure a 1 (guasto).
- Classificazione multiclasse per tipo di guasto: il `faultCode` può essere pari a 0 (no-guasto), a 5 (usura della pala al 5%) oppure a 10 (usura della pala al 10%).
- Classificazione multiclasse per isolamento: il `faultCode` può essere pari a 0 (no-guasto) oppure a $k \in [1, 6]$ dove k è pari al numero del motore sul quale è installata la pala usurata.
- Classificazione multiclasse con 13 classi: il `faultCode` può essere pari a 0 (no-guasto) oppure a $k \in [1, 12]$ dove

$$\begin{cases} k \leq 6 \implies \text{Guasto al } 5\% \text{ sul motore } k\text{-esimo} \\ k > 6 \implies \text{Guasto al } 10\% \text{ sul motore } (k-6)\text{-esimo} \end{cases}$$

3.2. Impostazione della frame policy

Una volta importata la data table, è stata configurata la frame policy per segmentare i segnali in finestre temporali omogenee. In particolare, sono stati impostati:

- **Frame Size** (**FS**): 1 secondo, che definisce la durata di ciascuna finestra di analisi.
- **Frame Rate** (**FR**): 1 secondo, ossia l'intervallo con cui vengono aggiornate le feature.

Questa scelta permette di ottenere una rappresentazione temporale uniforme e bilanciata del segnale. La segmentazione in finestre da 1 secondo consente di catturare variazioni dinamiche sufficientemente dettagliate senza introdurre troppa variabilità dovuta a rumore.

3.3. Estrazione del contenuto spettrale tramite modello autoregressivo

Per analizzare il contenuto in frequenza dei segnali, è stata applicata la tecnica di stima spettrale basata su un modello autoregressivo (AR). Il modello AR stima lo spettro di potenza rappresentando il segnale come una combinazione lineare dei suoi valori passati e di un termine di errore. Questa tecnica offre diversi vantaggi:

- Permette una stima accurata delle componenti frequenziali anche in presenza di segnali non completamente stazionari.
- Consente di identificare in modo preciso le frequenze dominanti, essenziali per rilevare anomalie quali spostamenti di picco o variazioni nell'energia distribuita su determinate bande.

In DFD, l'opzione «Autoregressive Model» è stata abilitata per tutti i segnali, garantendo così che ogni finestra temporale venga analizzata sia nel dominio del tempo che in quello della frequenza.

3.4. Scelta delle feature: dominio temporale e frequenziale

L'estrazione delle feature è stata divisa in due categorie principali:

3.4.1. Feature in dominio temporale

Per caratterizzare il comportamento dei segnali nel tempo sono state scelte le seguenti metriche:

- **Mean**: calcola il valore medio all'interno della finestra, fornendo una misura della tendenza centrale.
- **RMS** (Root Mean Square): misura il valore efficace del segnale, rappresentando l'energia complessiva.
- **Deviazione standard**: quantifica la dispersione dei valori attorno alla media, utile per identificare variazioni di ampiezza e rumore.
- **Skewness**: valuta l'asimmetria della distribuzione del segnale. Una deviazione dallo zero può indicare la presenza di impulsi o distorsioni.
- **Kurtosis**: misura la «piccolezza» della distribuzione, evidenziando l'esistenza di picchi estremi o outlier, potenzialmente legati a anomalie nel comportamento del sistema.
- **Crest Factor**: rapporto tra il valore di picco e il valore RMS. Elevati valori possono indicare la presenza di impulsi significativi.

- **Peak Value**: valore massimo raggiunto dal segnale, utile per rilevare eventi transitori di alta intensità.

Queste feature sono state scelte perché in grado di descrivere in maniera dettagliata le variazioni e le caratteristiche del segnale che potrebbero essere influenzate dall'usura delle pale.

3.4.2. Feature in dominio frequenziale

Per quanto riguarda il dominio della frequenza, le feature estratte sono state:

- **Peak Frequency**: la frequenza alla quale lo spettro di potenza raggiunge il suo massimo, indicativa della componente dominante del segnale.
- **Band Power**: la potenza totale contenuta all'interno della banda di frequenze definita. Questo parametro è particolarmente utile per rilevare variazioni nell'energia spettrale che possono derivare da anomalie meccaniche.

L'adozione di queste feature spettrali consente di evidenziare eventuali spostamenti o variazioni nell'energia dei segnali, che risultano discriminanti nel riconoscimento del guasto.

3.5. Procedura di calcolo delle feature

L'estrazione delle feature è stata realizzata in due fasi:

- **Signal Feature Extraction**: per ogni finestra di 1 secondo, DFD ha calcolato le feature temporali descritte, producendo per ogni segnale un insieme di metriche che rappresentano la dinamica locale.
- **Spectrum Feature Extraction**: utilizzando la stima spettrale AR, DFD ha elaborato le feature in dominio frequenziale per ogni finestra, determinando la Peak Frequency e la Band Power. Per realizzare tale passaggio, è stata impostata una frequenza minima a 0.1 Hz e un numero di picchi pari a 1.

Il risultato complessivo è stato la generazione di un gran numero di feature (circa 400 complessive, nonostante il numero preciso vari in funzione del tipo di classificazione da implementare). Questo elevato numero di feature ha permesso di avere una rappresentazione estremamente ricca e dettagliata del comportamento del drone in volo.

3.6. Considerazioni sul ranking e la selezione delle feature

Ogni esperimento, per ottimizzare l'addestramento del classificatore e ridurre il rischio di overfitting, ha comportato anche l'applicazione di una procedura di ranking delle feature (la maggior parte delle volte tramite ANOVA). Questo processo ha permesso di identificare e mantenere un sottoinsieme delle feature più discriminanti (ad es. le 51 migliori variabili) per l'addestramento dei modelli. I vari ranking delle feature verranno discussi durante l'esposizione dei risultati dei modelli di classificazione (Sezione 4), così da descrivere l'approccio utilizzato per ogni esperimento.

3.7. Esportazione dei dati per il Classification Learner

Al termine dell'estrazione, la feature table completa, contenente tutte le feature temporali e spettrali associate ad ogni finestra di analisi e corredate dal `faultCode`, è stata esportata. Questa tabella è stata successivamente importata nel Classification Learner di MATLAB, dove è stato possibile addestrare i vari modelli di classificazione utilizzando l'intero set di feature o un sottoinsieme ottimizzato.

3.8. Impatto dell'estrazione delle feature sul modello predittivo

L'approccio adottato per l'estrazione delle feature ha permesso di catturare in modo esaustivo le variazioni nei segnali che derivano da anomalie meccaniche, quali quelle generate da un'usura delle pale. In particolare, la combinazione di metriche temporali (che forniscono informazioni sull'andamento locale del segnale) e spettrali (che evidenziano la distribuzione dell'energia in frequenza) ha reso possibile la costruzione di modelli di classificazione altamente accurati, in linea con quanto riportato in letteratura.

4. Risultati dei modelli di classificazione

In questa sezione vengono presentati e discussi i risultati ottenuti dai modelli di classificazione, analizzati su diverse configurazioni (classificatore binario a 2 classi, multiclasse a 3 classi, multiclasse a 7 classi e

multiclasse a 13 classi). Per garantire la robustezza e la generalizzabilità dei modelli, è stata adottata una strategia di validazione che prevede sia la 10-fold cross validation sia l'utilizzo di un test set indipendente (30% del dataset).

4.1. Strategia di validazione e testing

Per tutti gli esperimenti è stata utilizzata una **10-fold cross validation**. In questo approccio il dataset viene suddiviso in 10 parti; ad ogni iterazione, 9 parti sono impiegate per l'addestramento mentre la restante viene utilizzata per la validazione. Questo metodo riduce la varianza nella stima delle performance e garantisce una valutazione affidabile del modello, minimizzando il rischio di overfitting.

Oltre alla cross validation, è stato predisposto un **test set indipendente** che rappresenta il 30% del dataset e viene tenuto da parte sin dall'inizio. Il vantaggio di questa suddivisione preliminare è duplice:

- Permette di ottenere una stima veritiera delle performance del modello su dati non visti durante l'addestramento e la validazione.
- Consente di verificare la capacità di generalizzazione del modello, offrendo un confronto tra l'accuratezza in fase di validazione e quella sul test set.

4.2. Workflow sperimentale

Il workflow sperimentale adottato per ciascun classificatore ha seguito i seguenti passaggi:

1. **Addestramento con l'intero set di feature:** tutti i modelli forniti dal Classification Learner sono stati addestrati utilizzando l'intero set di feature (circa 400) derivato dal Diagnostic Feature Designer.
2. **Testing sul test set:** i modelli addestrati sono stati valutati sul test set, registrando le metriche di performance.
3. **Selezione del modello migliore:** Sono stati individuati sia il modello con la migliore accuratezza in validazione sia quello con la migliore accuratezza sul test set.

Lo stesso workflow appena descritto è stato ripetuto utilizzando sottoinsiemi di feature selezionati tramite algoritmi di ranking (51 feature e 16 feature).

È stato effettuato un caso sperimentale (8feat) in cui è stato testato un input estremamente ridotto, selezionato con un algoritmo alternativo (MRMR).

Questo approccio ha permesso di valutare l'impatto della riduzione della dimensionalità sulle performance del classificatore e di identificare la configurazione ottimale per ciascun task.

4.3. Descrizione degli esperimenti

Nel prosieguo della sezione verranno riportati i migliori modelli per ognuno degli esperimenti effettuati. Nel dettaglio, verranno mostrate una matrice confusione di test e una tabella riepilogativa per ogni modello caratterizzata dalle seguenti metriche:

- **Accuracy di validation**
- **Accuracy di test**
- **Precision**
- **Recall**
- **F1-Score**

4.4. Breve descrizione dei modelli

In questo paragrafo forniremo una breve descrizione di tutti i modelli che si sono resi protagonisti dei migliori risultati ottenuti negli esperimenti.

4.4.1. Boosted Trees

Il metodo dei **Boosted Trees** è un approccio di *ensemble learning* che combina molti alberi decisionali «deboli» (weak learners) in modo iterativo. A ogni passo, il modello cerca di migliorare gli errori commessi dall'insieme di alberi precedenti, aggiungendo un nuovo albero che pone maggior enfasi sulle istanze mal classificate. In questo modo, l'errore complessivo tende a ridursi progressivamente, fornendo un classificatore finale più robusto di un singolo albero.

4.4.2. Bagging Trees

Il **Bagging** (acronimo di Bootstrap Aggregating) è un metodo di ensemble learning in cui vengono generati più alberi decisionali (in genere indipendenti tra loro) addestrati su sottoinsiemi casuali (bootstrap) del dataset originale. Alla fine, la previsione finale si ottiene mediando (per la regressione) o facendo la maggioranza (per la classificazione) delle singole predizioni dei vari alberi.

4.4.3. SVM (Cubic e Quadratic)

Le **Support Vector Machine** (**SVM**) sono algoritmi di machine learning supervisionati che ricercano l'iperpiano (o insieme di iperpiani) che massimizza la distanza (margine) di separazione tra le diverse classi in uno spazio delle feature.

Quando si parla di **Cubic SVM**, ci si riferisce a una SVM che utilizza un kernel polinomiale di grado 3 (kernel «cubic»).

La **Quadratic SVM**, invece, utilizza un kernel polinomiale di grado 2 (kernel quadratico).

4.4.4. Wide Neural Network

Una **Wide Neural Network** è un'architettura di rete neurale caratterizzata da pochi strati (depth ridotta), ma con un elevato numero di neuroni per ogni strato (larghezza). L'idea alla base è di fornire alla rete la capacità di apprendere pattern complessi grazie all'elevata dimensionalità dei livelli intermedi, pur mantenendo una profondità contenuta.

4.4.5. RUSBoosted Trees

RUS sta per *Random UnderSampling*, ed è una variante del metodo *Boosted Trees* pensata per gestire dataset sbilanciati (dove una o più classi sono in netta minoranza). L'idea è di combinare l'algoritmo di boosting con un meccanismo di undersampling casuale della classe (o delle classi) più numerose a ogni iterazione, così da rendere il training più «equilibrato».

4.4.6. Subspace Discriminant

Il metodo **Subspace Discriminant** è un approccio di ensemble learning che utilizza sottoinsiemi di feature (random subspace) per addestrare classificatori di tipo discriminant (ad esempio, *Linear* o *Quadratic Discriminant Analysis*). Ogni classificatore «debole» viene quindi allenato su un set di feature estratte casualmente, e le previsioni finali si ottengono combinando (ad esempio, con voto di maggioranza) i risultati dei singoli classificatori.

4.5. Classificazione binaria (guasto/no-guasto)

In questa sezione analizzeremo i migliori risultati ottenuti durante l'addestramento del classificatore binario (guasto/no-guasto).

4.5.1. 401 Feature

Il primo esperimento ha riguardato l'utilizzo di tutte e 401 le feature estratte attraverso il DFD.

In questo esperimento un unico modello ha sovrastato gli altri sia in termini di accuratezza di validation che in termini di accuratezza di test.

4.5.1.1. Modello Boosted Trees (1)

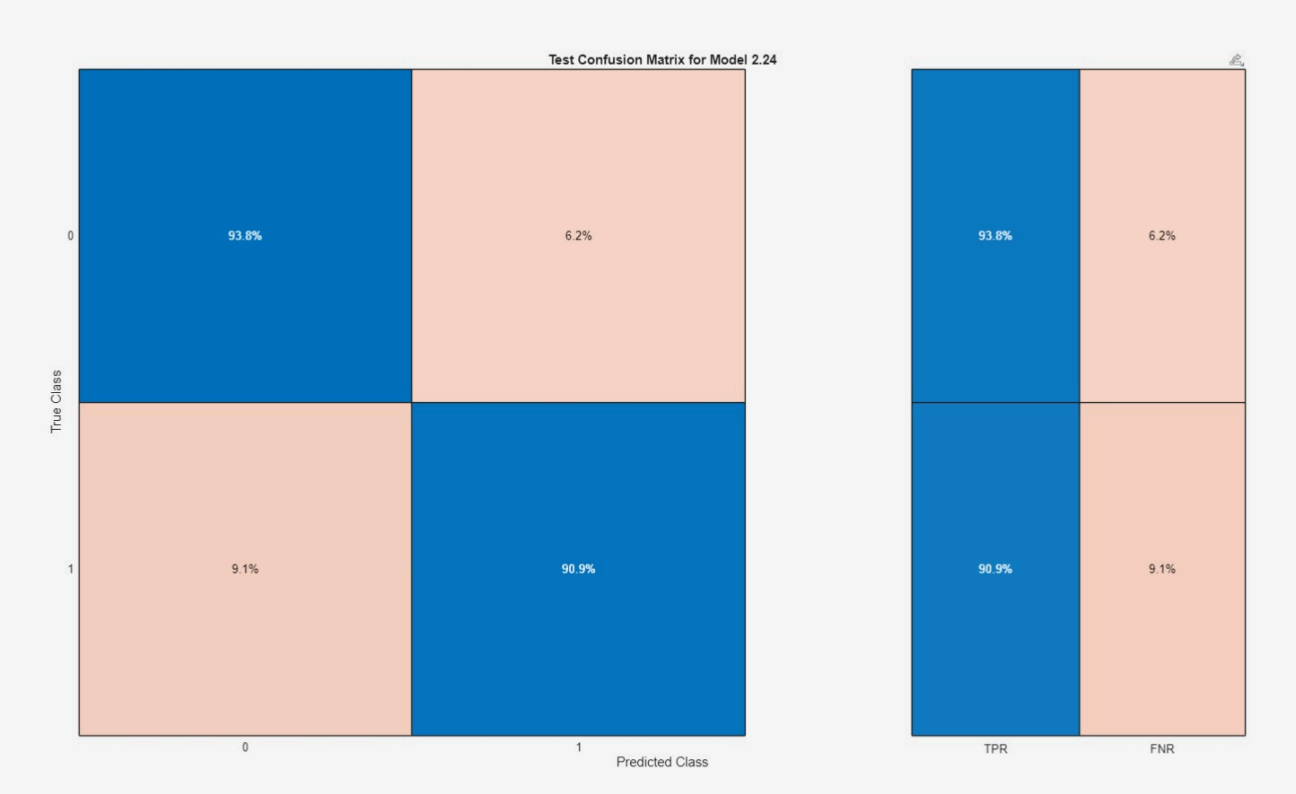
Metrica	Valore
Accuracy (val)	95.0%
Accuracy (test)	96.4%
Precision	96.4%
Recall	96.4%
F1-Score	96.4%

4.5.2. 51 Feature selezionate tramite ranking ROC

Il secondo esperimento del classificatore binario ha riguardato l'utilizzo di 51 feature selezionate attraverso l'algoritmo di ranking ROC. Tale algoritmo è stato scelto sia per il fatto che è utilizzabile in ambiti di classificazione binaria, ma anche perché risulta essere il metodo più «oggettivo» per il ranking delle feature dal momento che rappresenta quanto bene una feature separa le due classi senza dipendere da ipotesi sulla distribuzione dei dati.

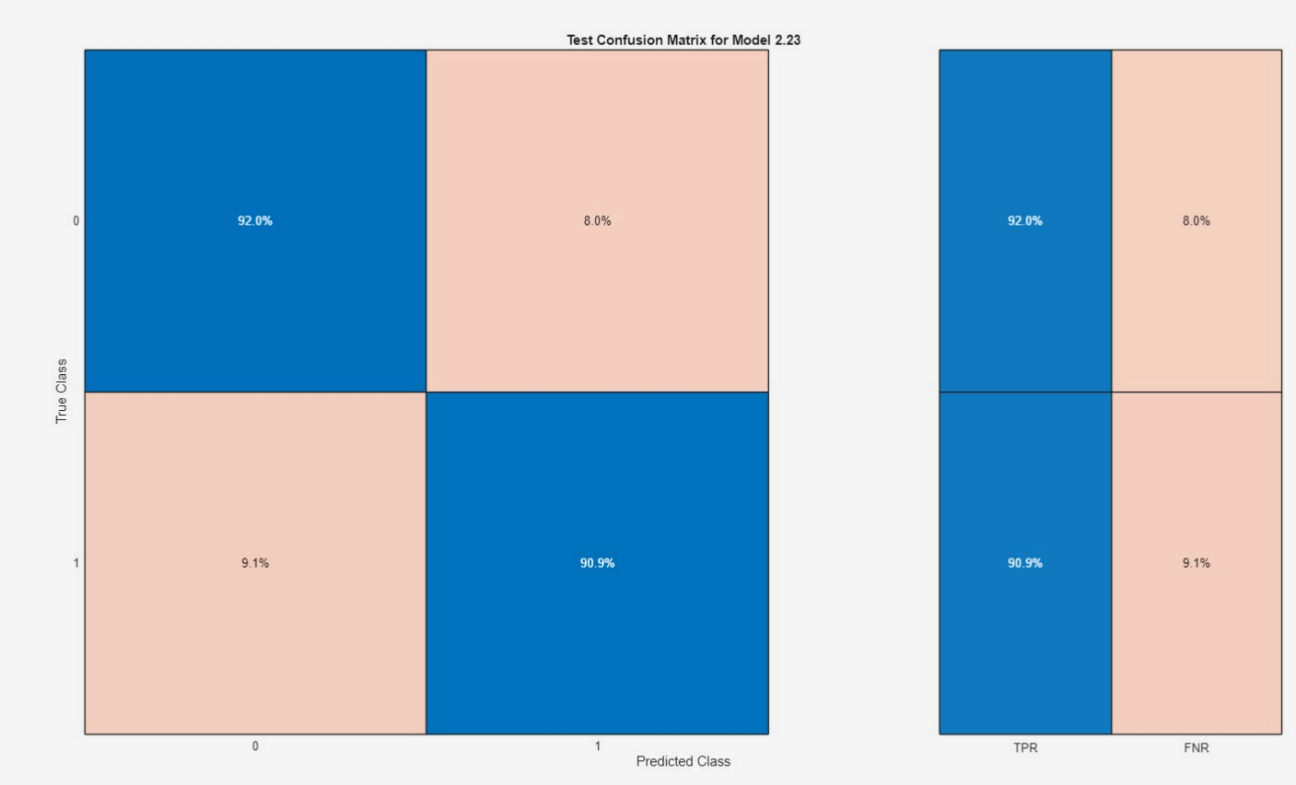
In questo esperimento sono due i modelli che hanno performato meglio, uno in termini di accuratezza di validation, l'altro in termini di accuratezza di test.

4.5.2.1. Modello Bagged Trees (2)



Metrica	Valore
Accuracy (val)	93.0%
Accuracy (test)	91.8%
Precision	91.5%
Recall	87.2%
F1-Score	89.3%

4.5.2.2. Modello Boosted Trees (3)



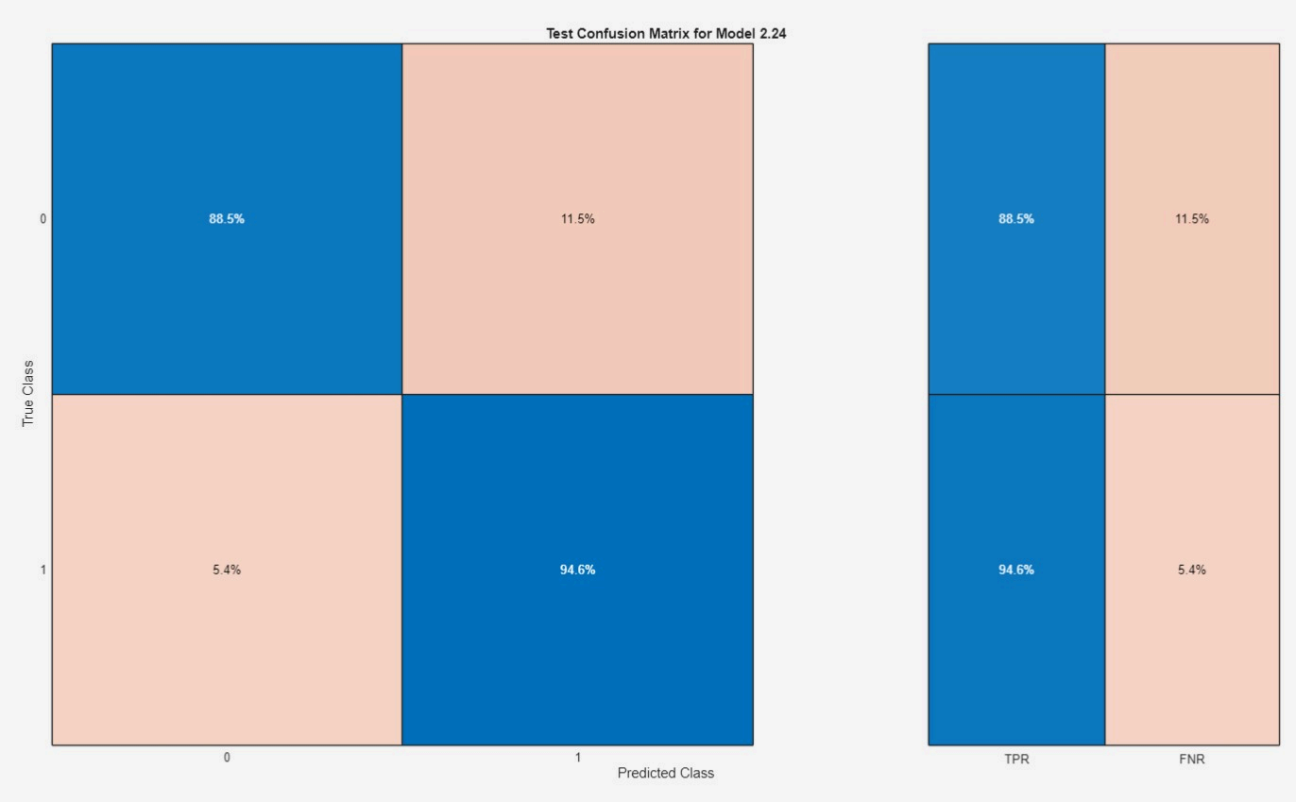
Metrica	Valore
Accuracy (val)	94.7%
Accuracy (test)	91.3%
Precision	92.8%
Recall	86.9%
F1-Score	89.7%

4.5.3. 16 Feature selezionate tramite ranking ROC

Il terzo esperimento del classificatore binario ha riguardato l'utilizzo di 16 feature selezionate attraverso l'algoritmo di ranking ROC.

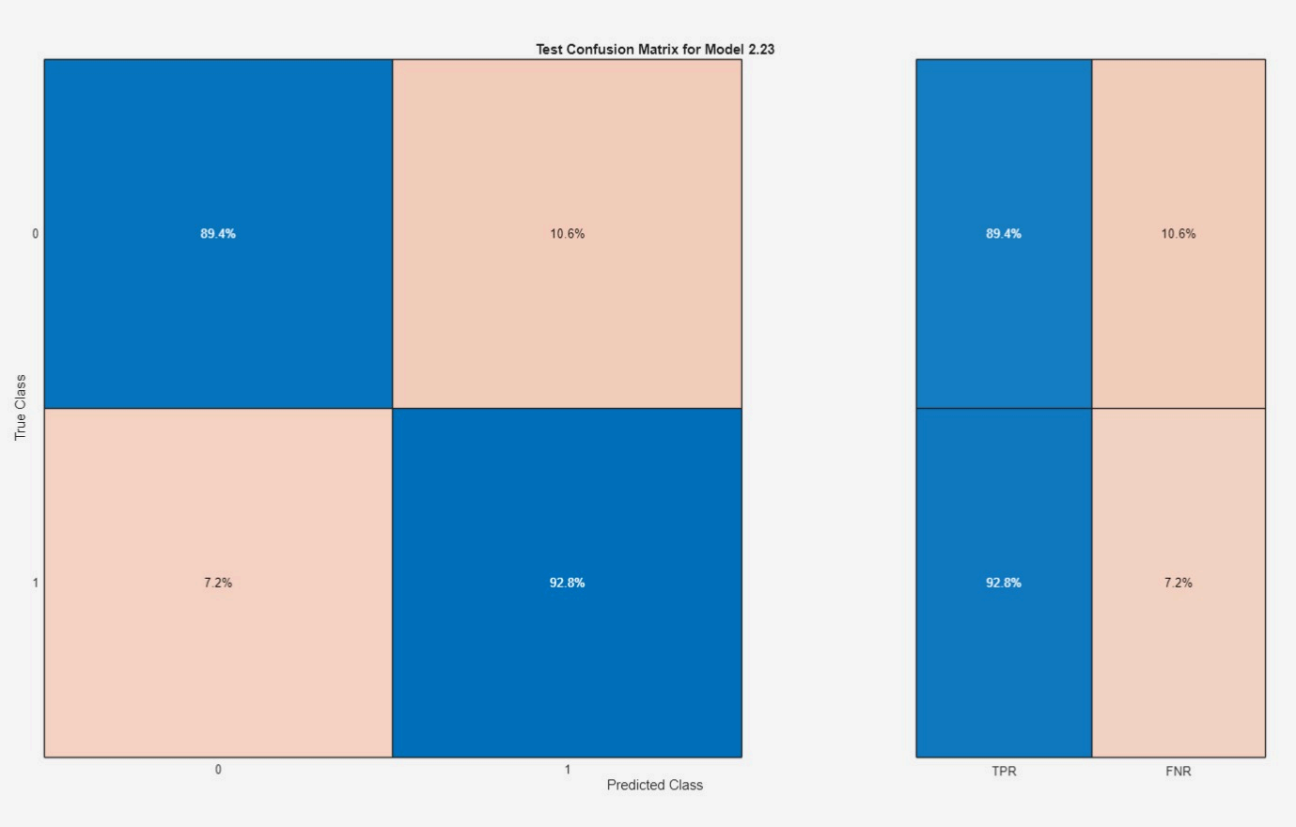
Anche in questo esperimento sono due i modelli che hanno performato meglio.

4.5.3.1. Modello Bagged Trees (4)



Metrica	Valore
Accuracy (val)	93.6%
Accuracy (test)	92.8%
Precision	92.8%
Recall	92.8%
F1-Score	92.8%

4.5.3.2. Modello Boosted Trees (5)



Metrica	Valore
Accuracy (val)	93.8%
Accuracy (test)	91.8%
Precision	92.0%
Recall	91.8%
F1-Score	91.9%

4.6. Classificazione multiclasse a 3 classi per tipo di guasto (no-guasto, 5%, 10%).

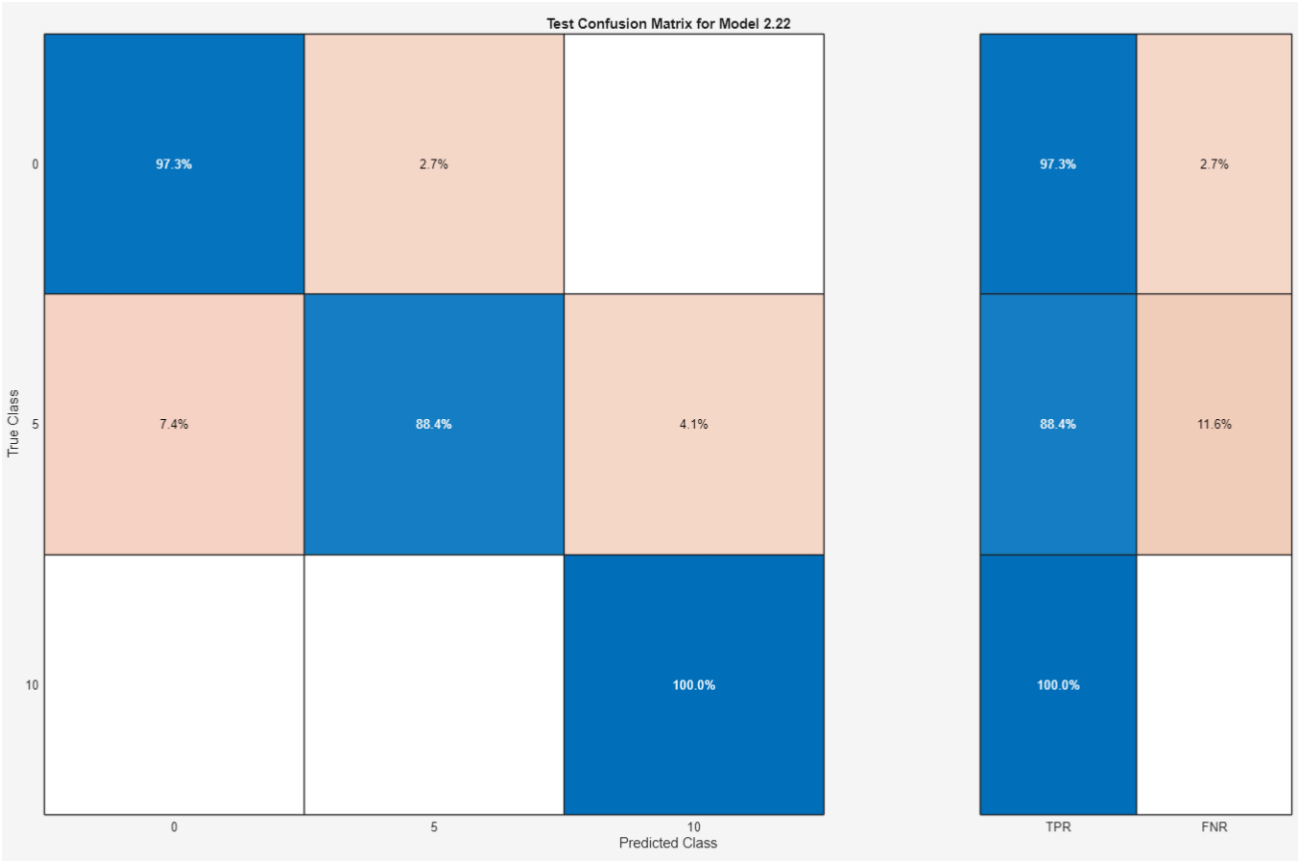
In questa sezione analizzeremo i migliori risultati ottenuti durante l'addestramento del classificatore multiclasse per tipo di guasto (no-guasto, 5%, 10%) a 3 classi.

4.6.1. 415 Feature

Il primo esperimento ha riguardato l'utilizzo di tutte e 415 le feature estratte attraverso il DFD.

In questo esperimento un unico modello ha sovrastato gli altri sia in termini di accuratezza di validation che in termini di accuratezza di test.

4.6.1.1. Modello Boosted Trees (6)

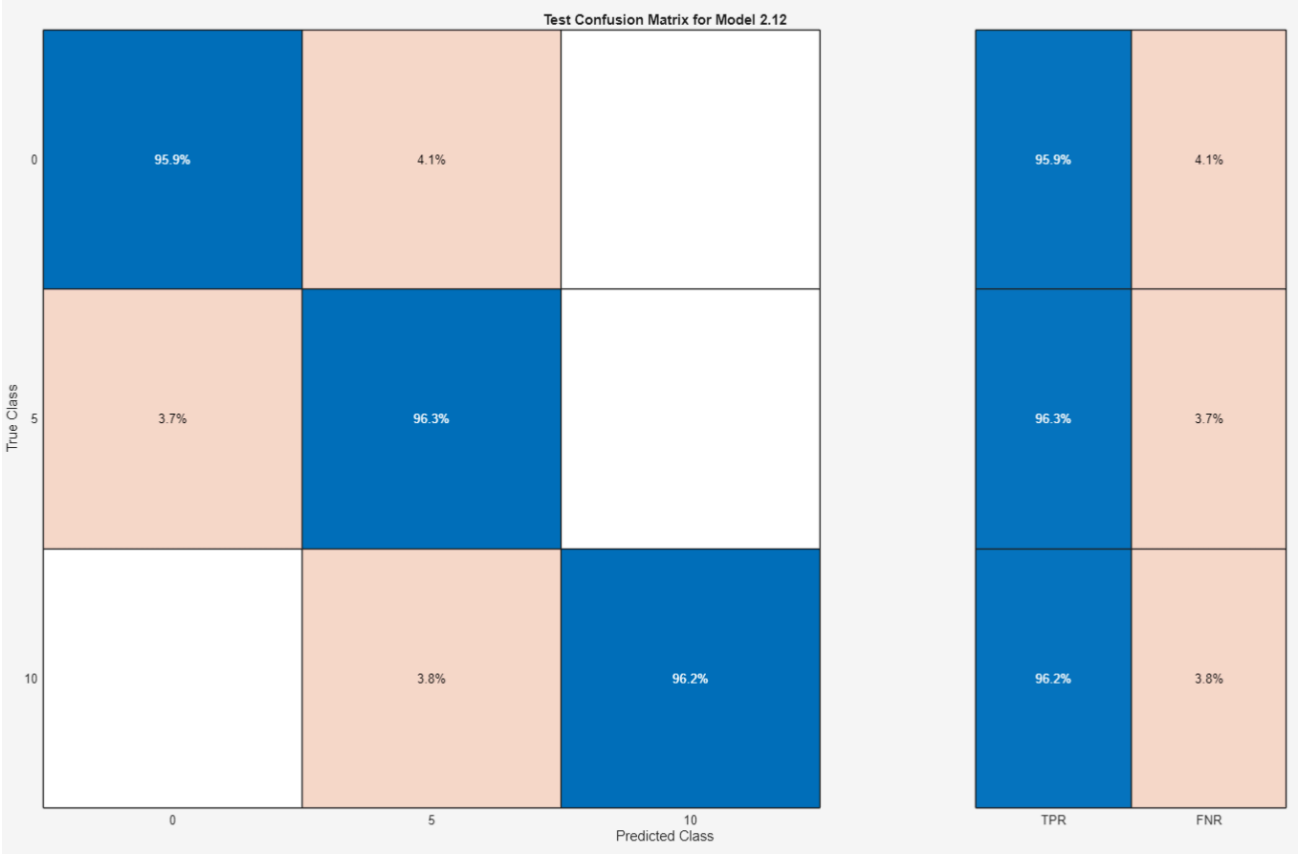


Metrica	Valore
Accuracy (val)	93.6%
Accuracy (test)	95.6%
Precision	95.7%
Recall	95.6%
F1-Score	95.6%

4.6.2. 51 Feature selezionate tramite ANOVA

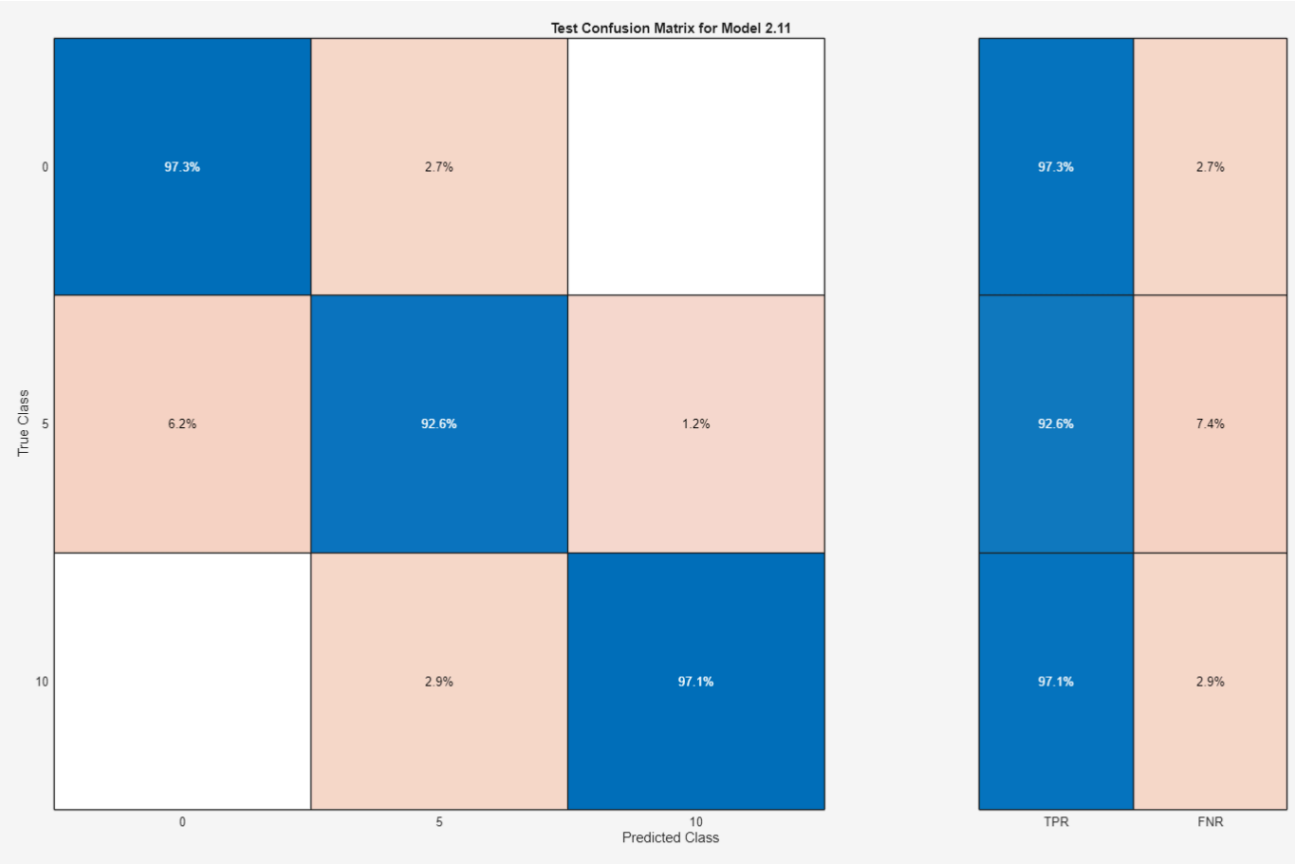
Il secondo esperimento ha riguardato l'utilizzo di 51 feature selezioante mediante l'algoritmo di ranking One-way ANOVA. Tale algoritmo è stato scelto per la sua ottima capacità di discriminare le feature migliori in contesti multiclasse.

4.6.2.1. Modello Cubic SVM (7)



Metrica	Valore
Accuracy (val)	96.2%
Accuracy (test)	96.1%
Precision	96.3%
Recall	96.1%
F1-Score	96.2%

4.6.2.2. Modello Quadratic SVM (8)

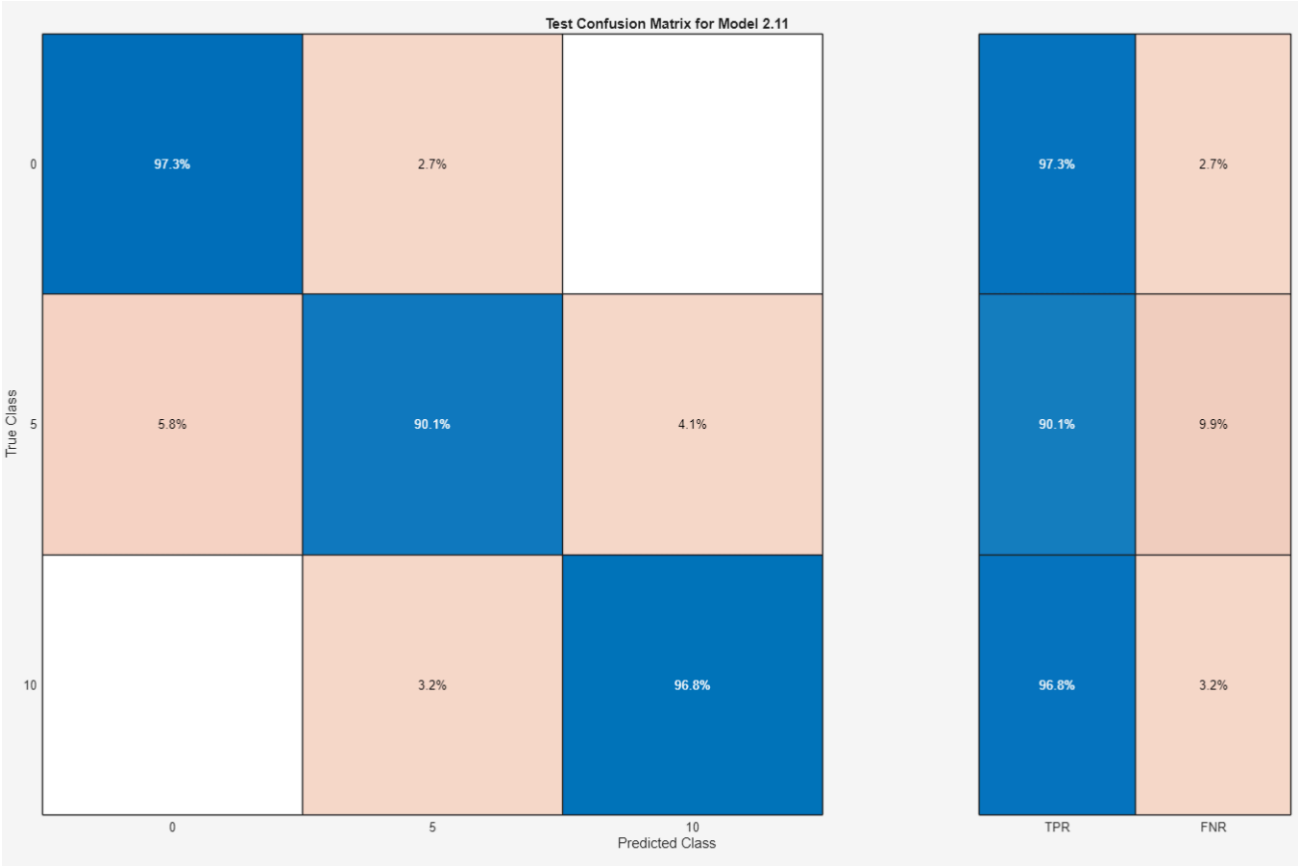


Metrica	Valore
Accuracy (val)	97.2%
Accuracy (test)	95.8%
Precision	95.8%
Recall	95.8%
F1-Score	95.8%

4.6.3. 16 Feature selezionate tramite ANOVA

Il terzo esperimento ha riguardato l'utilizzo di 16 feature selezioante mediante l'algoritmo di ranking One-way ANOVA.

4.6.3.1. Modello Quadratic SVM (9)

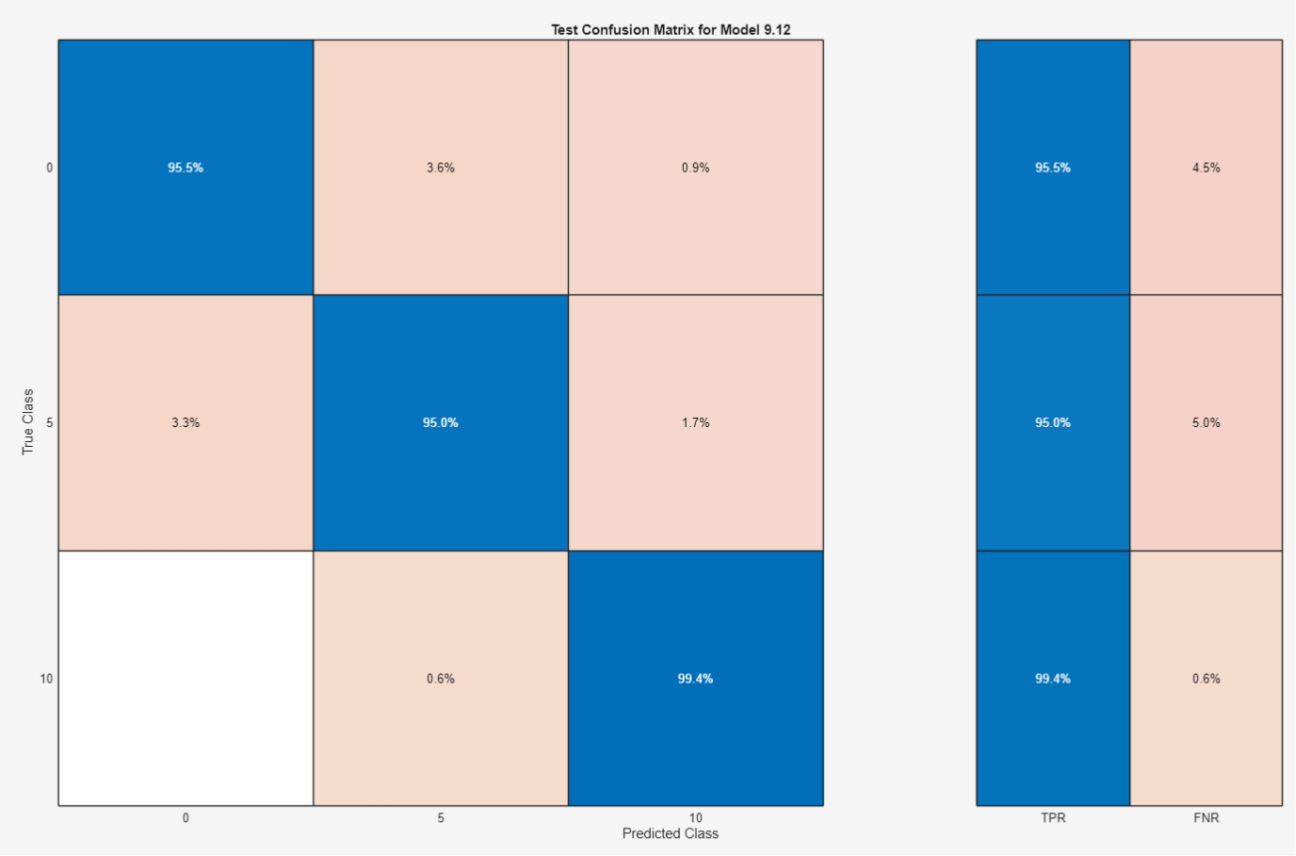


Metrica	Valore
Accuracy (val)	94.5%
Accuracy (test)	94.9%
Precision	94.9%
Recall	94.9%
F1-Score	94.8%

4.6.4. 51 Feature selezionate tramite χ^2

Il quarto esperimento ha riguardato l'utilizzo di 51 feature selezionate mediante l'algoritmo di ranking χ^2. Tale algoritmo è stato scelto per svolgere un ulteriore esperimento che utilizzasse un algoritmo prettamente statistico e per valutare le differenze ottenute sulla base di un altro algoritmo.

4.6.4.1. Modello Cubic SVM (10)

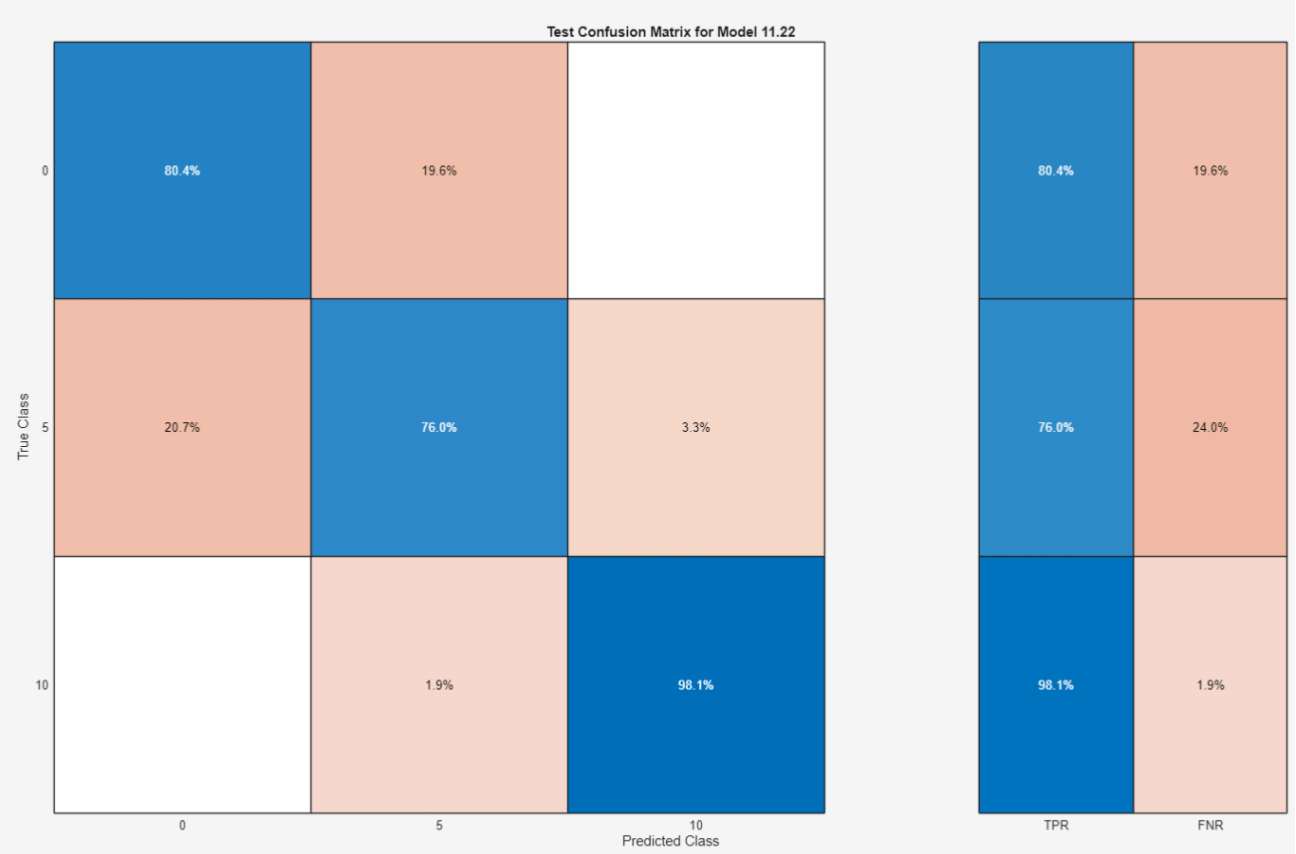


Metrica	Valore
Accuracy (val)	95.4%
Accuracy (test)	96.9%
Precision	96.9%
Recall	96.9%
F1-Score	96.9%

4.6.5. 8 Feature selezionate tramite MRMR

Il quinto esperimento ha riguardato l'utilizzo di 8 feature selezionate mediante l'algoritmo di ranking MRMR. Tale algoritmo è stato scelto come ulteriore test per valutare l'andamento dei modelli in contesti estremamente ristretti, situazione che potrebbe risultare vincente in scenari con vincoli di memoria.

4.6.5.1. Boosted Trees (11)



Metrica	Valore
Accuracy (val)	85.1%
Accuracy (test)	86.1%
Precision	86.1%
Recall	86.1%
F1-Score	86.1%

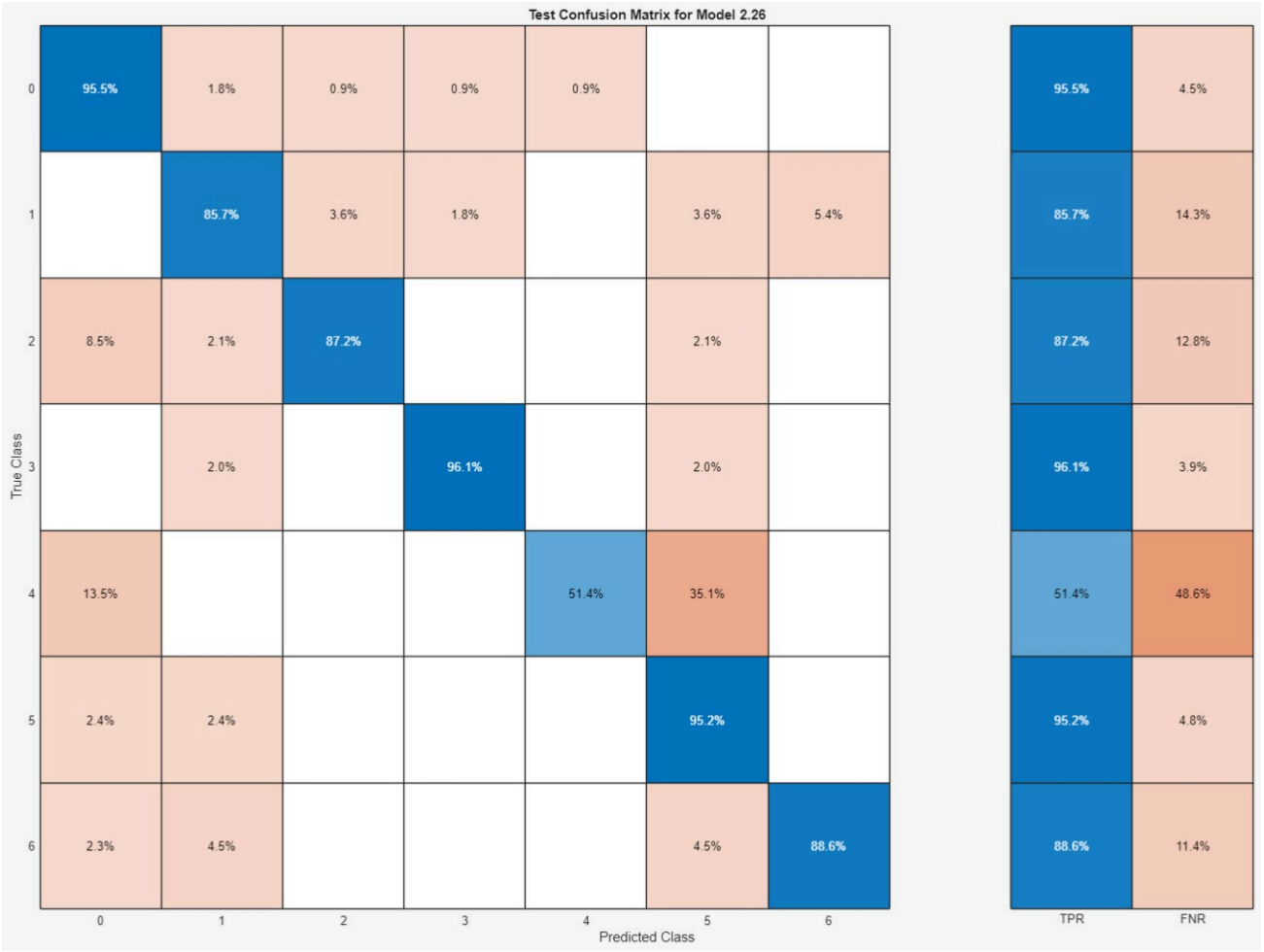
4.7. Classificatore multiclasse a 7 classi per isolamento (no-guasto e guasto localizzato al motore i-esimo).

In questa sezione analizzeremo i migliori risultati ottenuti durante l'addestramento del classificatore multiclasse per isolamento (no-guasto, guasto localizzato al motore i-esimo) a 7 classi.

4.7.1. 401 Feature

Il primo esperimento ha riguardato l'utilizzo di tutte e 401 le feature estratte attraverso il DFD.

4.7.1.1. Modello RUSBoosted Trees (12)



Metrica	Valore
Accuracy (val)	83.5%
Accuracy (test)	88.2%
Precision	89.4%
Recall	88.2%
F1-Score	87.9%

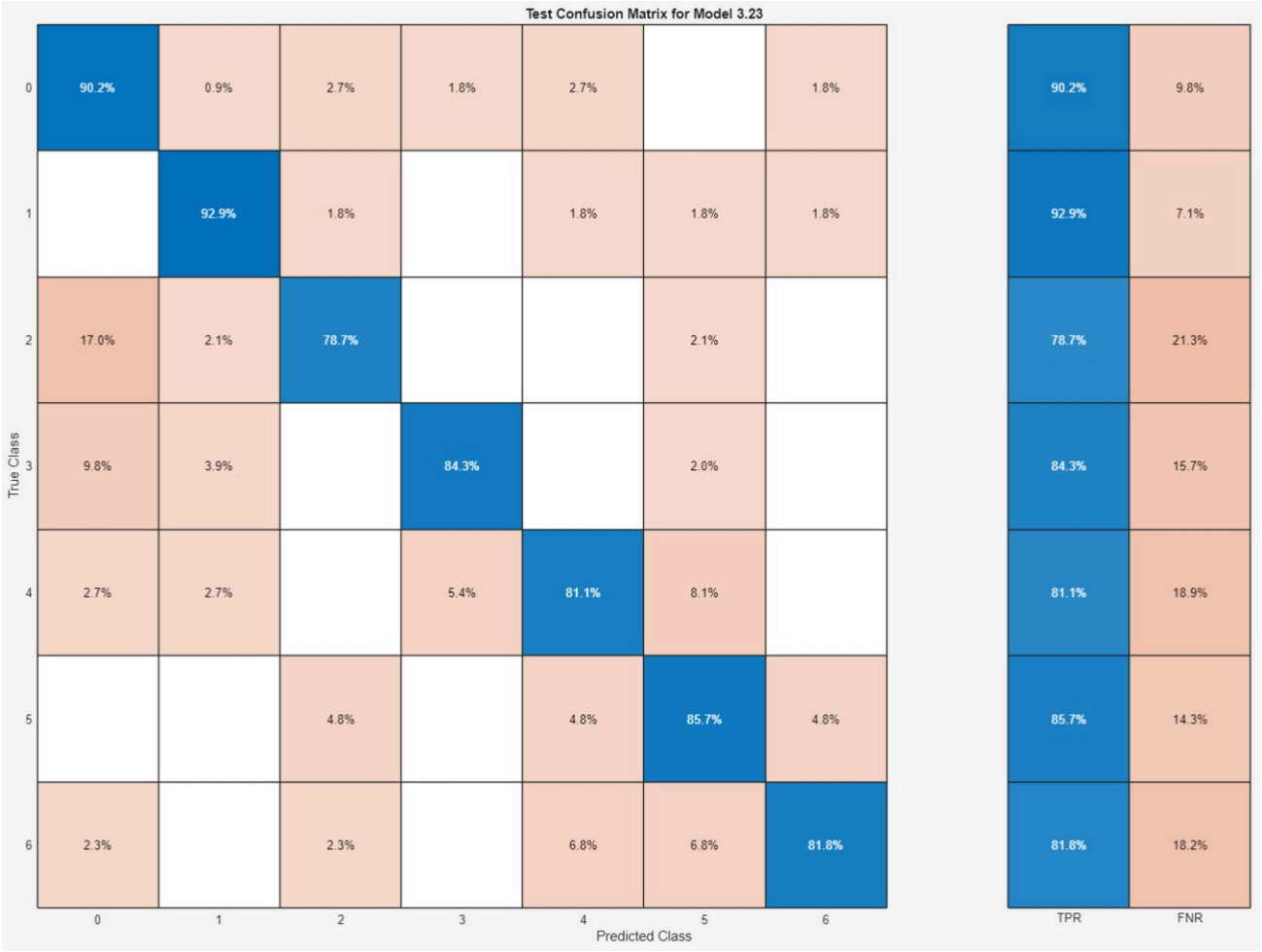
4.7.1.2. Modello Bagged Trees (13)

Metrica	Valore
Accuracy (val)	86.2%
Accuracy (test)	86.1%
Precision	87.4%
Recall	86.1%
F1-Score	86.0%

4.7.2. 51 Feature selezionate tramite ANOVA

Il secondo esperimento ha riguardato l'utilizzo di 51 feature selezionate mediante l'algoritmo di ranking One-way ANOVA.

4.7.2.1. Modello Bagged Trees (14)



Metrica	Valore
Accuracy (val)	87.8%
Accuracy (test)	86.1%
Precision	86.2%
Recall	86.1%
F1-Score	86.1%

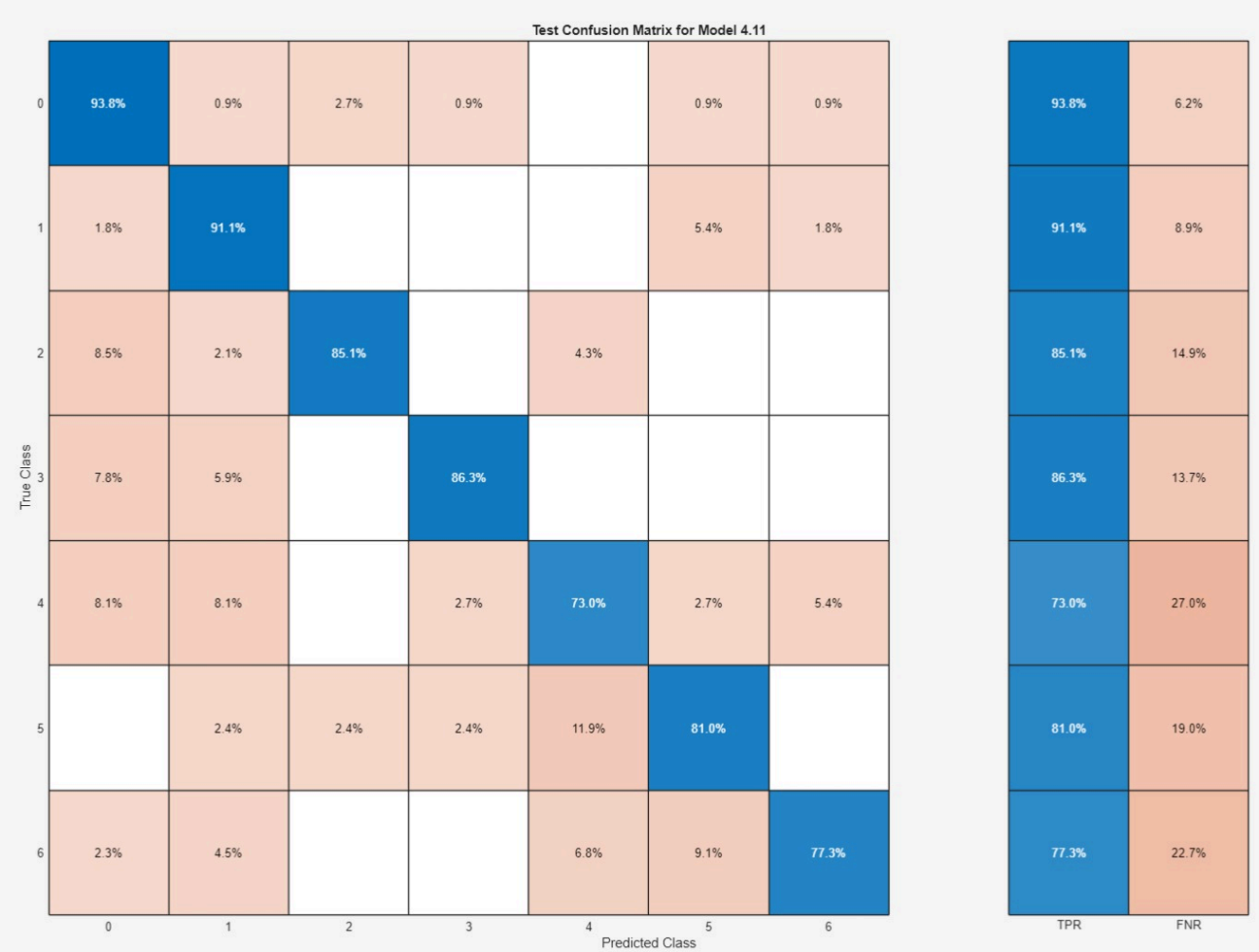
4.7.3. 16 Feature selezionate tramite ANOVA

Il terzo esperimento ha riguardato l'utilizzo di 16 feature selezionate mediante l'algoritmo di ranking One-way ANOVA.

4.7.3.1. Modello Wide Neural Network (15)

Metrica	Valore
Accuracy (val)	86.7%
Accuracy (test)	87.7%
Precision	88.3%
Recall	87.9%
F1-Score	88.0%

4.7.3.2. Modello Quadratic SVM (16)



Metrica	Valore
Accuracy (val)	88.1%
Accuracy (test)	86.1%
Precision	86.5%
Recall	86.3%
F1-Score	86.3%

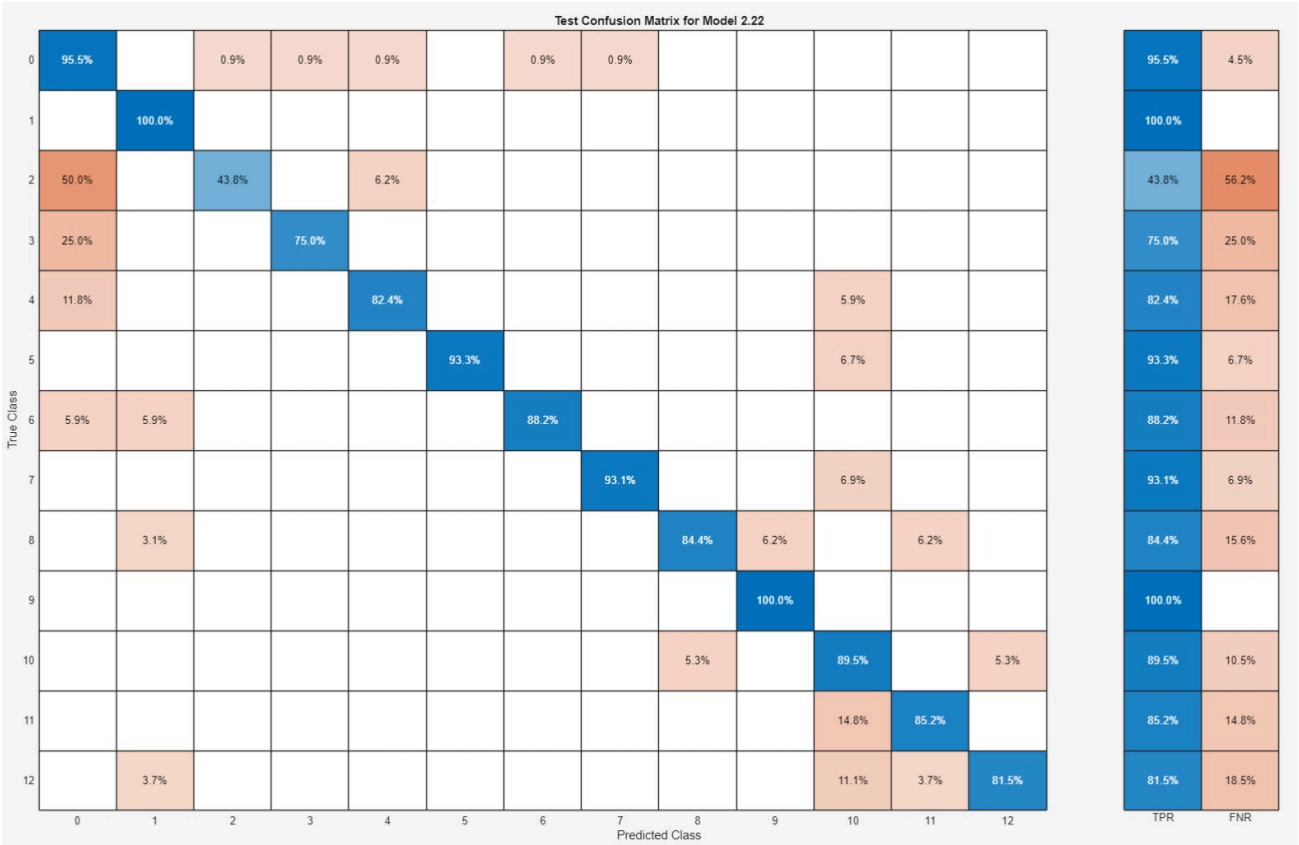
4.8. Classificatore multiclasse a 13 classi (no-guasto e guasto specifico, differenziando tra usura del 5% e del 10% per ciascun motore).

In questa sezione analizzeremo i migliori risultati ottenuti durante l'addestramento del classificatore multiclasse a 13 classi (no-guasto e guasto specifico, differenziando tra usura del 5% e del 10% per ciascun motore).

4.8.1. 429 Feature

Il primo esperimento ha riguardato l'utilizzo di tutte e 429 le feature estratte attraverso il DFD.

4.8.1.1. Modello Boosted Trees (17)

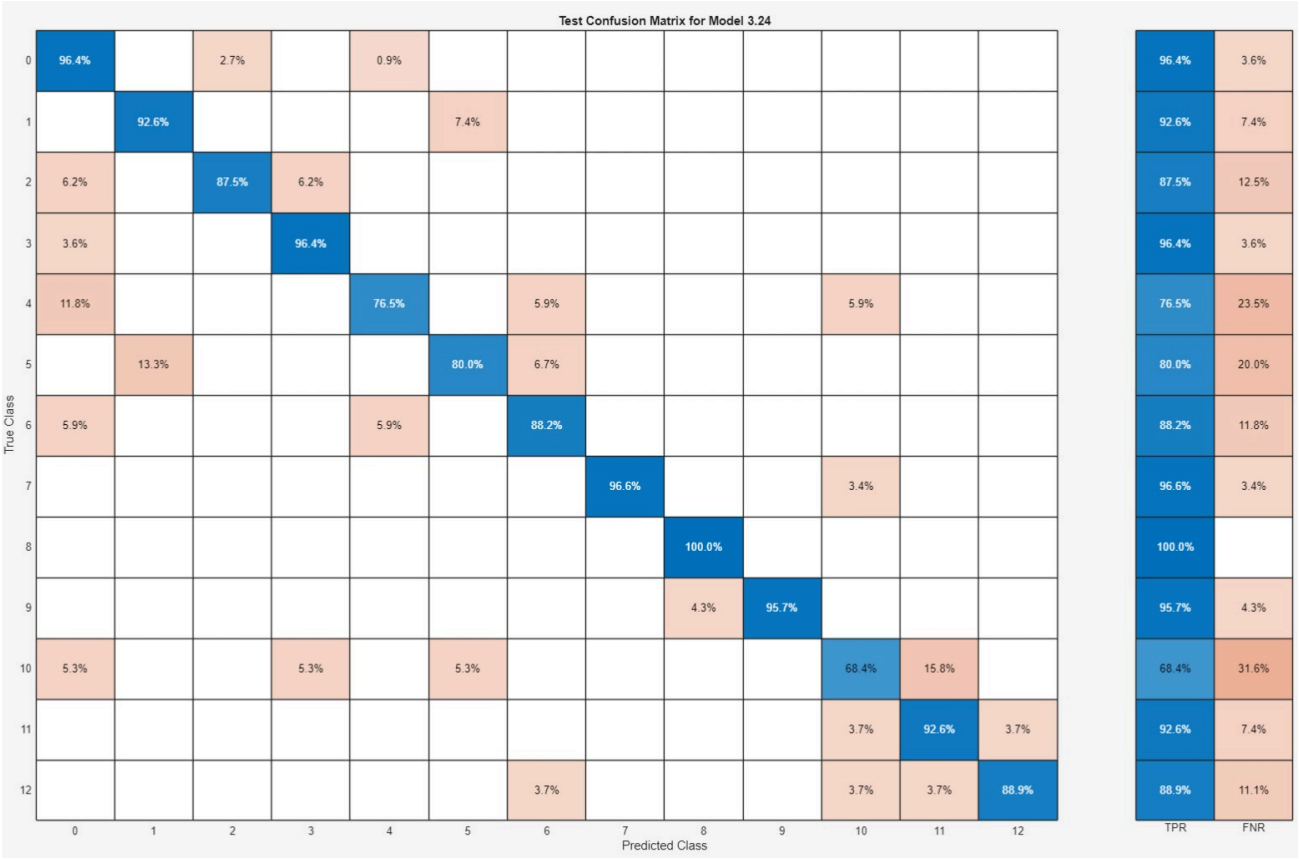


Metrica	Valore
Accuracy (val)	89.6%
Accuracy (test)	89.3%
Precision	89.4%
Recall	88.4%
F1-Score	88.2%

4.8.2. 51 Feature selezionate tramite ANOVA

Il secondo esperimento ha riguardato l'utilizzo di 51 feature selezionate mediante l'algoritmo di ranking One-way ANOVA.

4.8.2.1. Modello Subspace Discriminant (18)

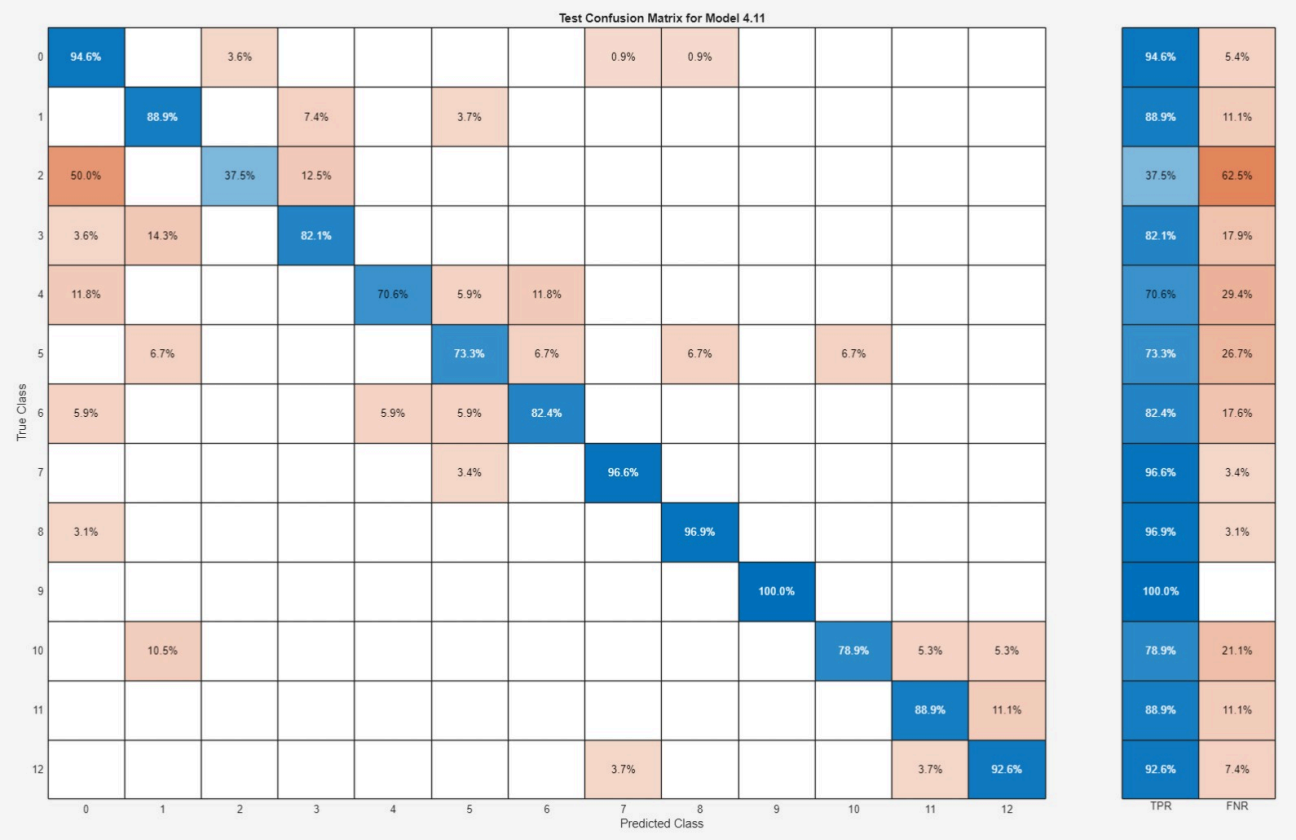


Metrica	Valore
Accuracy (val)	90.7%
Accuracy (test)	92.0%
Precision	92.0%
Recall	92.0%
F1-Score	92.0%

4.8.3. 16 Feature selezionate tramite ANOVA

Il terzo esperimento ha riguardato l'utilizzo di 16 feature selezionate mediante l'algoritmo di ranking One-way ANOVA.

4.8.3.1. Modello Quadratic SVM (19)



Metrica	Valore
Accuracy (val)	87.4%
Accuracy (test)	87.9%
Precision	87.6%
Recall	87.9%
F1-Score	87.5%

4.9. Tabella riepilogativa finale

Di seguito viene riportata una tabella riassuntiva che sintetizza le performance dei modelli testati. Le metriche riportate includono le accuratezze in fase di validation e in fase di test, ma anche il tempo di training e le dimensioni del modello, due metriche non menzionate precedentemente.

Inoltre, per ciascun esperimento sono indicati il numero di classi, il numero di feature utilizzate, l'algoritmo di ranking impiegato.

È importante considerare che gli esperimenti, e quindi i tempi di training risultanti, sono stati effettuati su un PC dotato di CPU Intel Core i7-11700KF, RAM Corsair CMK32GX4M2E3200C16 (2x16GB DDR4) e CMK16GX4M2D3600C18 (2x8GB DDR3), GPU Gigabyte NVIDIA GeForce RTX 3070 Ti 8 GB GDDR6X e SSD Crucial P5 Plus.

	Modello	Acc. (val)	Acc. (test)	T. training	Dim.	# classi	# feature	Alg. rank
1	*Boosted Trees*	95.0%	96.4%	9.26*s*	2 MB	2	401	—
2	*Bagged Trees*	93.0%	91.8%	7.46*s*	510 kB	2	51	ROC
3	*Boosted Trees*	94.7%	91.3%	5.26*s*	521 kB	2	51	ROC
4	*Bagged Trees*	93.6%	92.8%	6.6*s*	443 kB	2	16	ROC
5	*Boosted Trees*	93.8%	91.8%	6.2*s*	339 kB	2	16	ROC
6	*Boosted Trees*	93.6%	95.6%	20.9*s*	3 MB	3	415	—
7	*Cubic SVM*	96.2%	96.1%	**1.6*s***	187 kB	3	51	ANOVA
8	*Quadratic SVM*	**97.2%**	95.8%	2.3*s*	171 kB	3	51	ANOVA
9	*Quadratic SVM*	94.5%	94.9%	2.2*s*	51 kB	3	16	ANOVA
10	*Cubic SVM*	95.4%	**96.9%**	7.2*s*	174 kB	3	51	χ^2
11	*Boosted Trees*	85.1%	86.1%	18.2*s*	321 kB	3	8	MRMR
12	*RUSBoosted Trees*	83.5%	88.2%	35.3*s*	3 MB	7	401	—
13	*Bagged Trees*	86.2%	86.1%	39.2*s*	3 MB	7	401	—
14	*Bagged Trees*	87.8%	86.1%	17*s*	1 MB	7	51	ANOVA
15	*Wide Neural Network*	86.7%	87.7%	4.4*s*	**28 kB**	7	16	ANOVA
16	*Quadratic SVM*	88.1%	86.1%	4.4*s*	259 kB	7	16	ANOVA
17	*Boosted Trees*	89.6%	89.3%	44.5*s*	3 MB	13	429	—
18	*Subspace Discriminant*	90.7%	92.0%	20*s*	912 kB	13	51	ANOVA
19	*Quadratic SVM*	87.4%	87.9%	29.9*s*	742 kB	13	16	ANOVA

5. Conclusioni

5.1. Classificatore binario (2 classi)

Per il classificatore binario (guasto vs. no-guasto), i risultati migliori sono stati raggiunti con i Boosted Trees che, in una configurazione (con 401 feature), hanno mostrato un'accuracy sul test set pari al 96.4%. Le metriche precision e recall, in questi casi, si sono attestate entrambe su valori superiori al 90–95%, con il conseguente F1-score in linea. Ciò significa che il modello non soltanto distingue in modo efficace la presenza di un guasto, ma riduce il rischio di falsi allarmi (elevata precision) e di mancate rilevazioni (elevata recall).

Anche riducendo il set di feature (ad esempio a 16 feature), alcuni modelli (Bagged Trees) hanno mantenuto accuracy di 92.8% con precision e recall intorno al 92–93%, confermando un F1-score elevato. Questa stabilità testimonia la validità dell'approccio di feature selection, che permette di semplificare i modelli senza compromettere la capacità di discriminazione tra voli guasti e non guasti.

5.2. Classificatore multiclasse per tipo di guasto (3 classi)

5.2.1. Massima accuratezza con 51 feature

L'utilizzo di 51 feature opportunamente selezionate (ANOVA o χ^2) ha portato i modelli SVM, in particolare la Cubic, a superare persino le performance ottenute con oltre 400 feature, toccando il picco 96.9% di accuracy (con algoritmo χ^2). Le metriche precision, recall e F1-score, in questi casi, rimangono altissime, confermando un buon equilibrio fra l'identificazione corretta dei voli difettosi e la minimizzazione dei falsi allarmi.

5.2.2. Trade-off tra numero di feature e performance

Man mano che il set di feature si riduce (16 o addirittura 8 variabili), l'accuratezza subisce un calo più o meno marcato (fino all'86.1% con 8 feature). Tuttavia, precision e recall rimangono coerenti fra loro, indicando che il modello non penalizza in modo particolare una classe a scapito delle altre.

Se si punta a una soluzione più «leggera» e si accetta un decremento di performance, un set minimalista (es. 8 feature) può comunque fornire risultati dignitosi, con F1-score ben oltre l'80%.

5.2.3. Equilibrio nelle metriche (precision, recall, F1-score)

In quasi tutti gli esperimenti, precision e recall restano ad alti livelli e sostanzialmente bilanciate, generando valori di F1-score prossimi all'accuratezza. Questo riflette un bilanciamento tra falsi positivi e falsi negativi, cruciale in ambiti diagnostici dove sia le mancate rilevazioni (falsi negativi) sia gli allarmi inutili (falsi positivi) hanno impatti significativi.

5.3. Classificatore multiclasse a 7 classi per isolamento (no-guasto e guasto localizzato al motore i-esimo)

La classificazione a 7 classi, che prevede l'individuazione del motore difettoso, presenta maggiore complessità e uno sbilanciamento più marcato tra le classi (ad esempio, «no-guasto» vs. una singola tipologia di guasto per motore). In questi casi, non solo l'accuracy scende in un range 86–88%, ma metriche come precision e recall mostrano una variabilità leggermente più ampia tra le diverse classi.

- I RUSBoosted Trees, grazie al bilanciamento interno tramite Random UnderSampling, hanno ottenuto un'accuracy di test fino a 88.2% con valori medi di precision e recall attorno all'87–89% (e F1-score corrispondenti). L'uso di RUS aiuta soprattutto a ridurre il problema di classi poco rappresentate, migliorando la recall di alcune categorie di guasto.
- I Bagged Trees si assestano su un'accuracy di circa 86–87%, ma con un equilibrio tra precision e recall analogo. Di conseguenza, anche il F1-score riflette questa stabilità, rendendo il modello affidabile nell'individuare il motore in avaria, pur senza raggiungere le stesse vette del caso binario o a 3 classi.

5.4. Classificatore multiclasse a 13 classi (no-guasto e guasto con differenziazione 5% o 10% per ogni motore)

L'ultimo scenario, con ben 13 classi (no-guasto, guasto al 5% o al 10% per ciascuno dei 6 motori), rappresenta il problema più complesso. Con un numero così elevato di classi, l'accuracy si colloca attorno all'88–92%. In particolare, il Subspace Discriminant ha toccato un'accuracy di 92.0% in test, con valori di precision e recall e F1 score fissi al 92%.

In quest'ottica, si nota come, pur mantenendo una buona capacità di discriminazione, emerga una maggiore difficoltà nel distinguere guasti con percentuali di usura differenti sul medesimo motore. Le metriche precision e recall risultano però abbastanza equilibrate, indice del fatto che il modello non tende a confondere eccessivamente le classi più simili tra loro (5% vs 10%), ma inevitabilmente subisce un leggero calo dovuto all'aumento della complessità.

5.5. Considerazioni generali

5.5.1. Robustezza della pipeline

I passi di sincronizzazione, pre-elaborazione e l'estrazione di feature (sia temporali sia spettrali) hanno permesso di raggiungere elevati valori di accuracy, precision, recall e F1-score. L'uso congiunto di metriche multiple conferma che il sistema non si limita a riconoscere correttamente un alto numero di campioni, ma mantiene un buon equilibrio tra veri positivi e veri negativi.

5.5.2. Impatto della riduzione di dimensionalità

Ridurre il numero di feature (da centinaia a poche decine) non ha penalizzato, anzi ha spesso favorito, l'ottenimento di performance comparabili (in termini di accuracy, precision, recall, F1-score) a quelle del set

completo. Il ranking (ANOVA, ROC, χ^2, MRMR) dimostra come alcune feature siano altamente informative per la diagnosi dei guasti, mentre altre introducono ridondanza.

5.5.3. Distribuzione non bilanciata

In classificazioni complesse (7 o 13 classi) è emerso il ruolo importante di tecniche come il RUSBoosting, utili per gestire eventuali classi sotto-rappresentate. Questo si riflette positivamente su precision e recall di classi minori, incrementando l'F1-score e rendendo il modello globalmente più equo.

5.5.4. Ensemble e SVM come modelli chiave

I metodi di ensemble learning (Bagging, Boosting, RUSBoost) e le SVM (Quadratic/Cubic) risultano tra i più efficaci nel preservare un buon trade-off fra semplicità di addestramento e performance elevate su tutte le metriche considerate. La scelta fra un ensemble e una SVM può dipendere in parte da considerazioni sui tempi di training, sulla complessità e sulla dimensione finale del modello.