

Progetto di Machine Learning

1. Analisi dati
 - a. 1.1 Conteggio delle specie di pesci
 - b. 1.2 Analisi valori degli attributi nel dataset
 - c. 1.3 Matrice di correlazione
 - d. 1.4 Distribuzione degli attributi
2. Classificatori utilizzati
 - a. 2.1 Confronto tra tecniche di classificazione (senza scalatura dati)
 - b. 2.2 grafici di confronto
3. Confronto con e senza scaling dei dati
4. Confronto con e senza undersampling e oversampling
5. Confronto con e senza feature selection

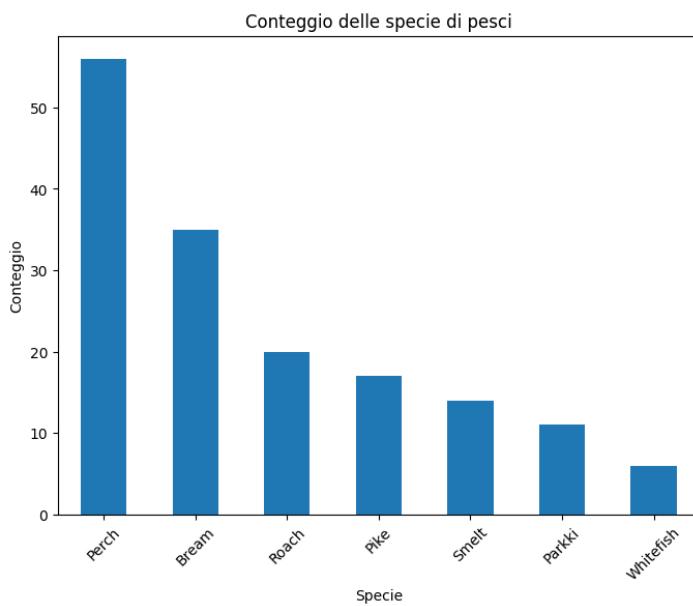
1. Analisi dei dati

Nel nostro progetto di Machine learning siamo partiti da un dataset per fare la classificazione di alcune specie di pesci (<https://www.kaggle.com/datasets/aungpyaeap/fish-market>).

Inizialmente come prima cosa abbiamo fatto l'analisi dell'intero dataset. Il nostro dataset è composto da 159 elementi con 6 attributi e 7 classi dove sono presenti i nomi della specie e sono presenti le caratteristiche del pesce come la lunghezza, la larghezza, l'altezza e il peso

Analisi statistica del dataset:						
# Analisi statistica degli attributi numerici						
	Weight	Length1	Length2	Length3	Height	Width
count	159.000000	159.000000	159.000000	159.000000	159.000000	159.000000
mean	398.326415	26.247170	28.415723	31.227044	8.970994	4.417486
std	357.978317	9.996441	10.716328	11.610246	4.286208	1.685804
min	0.000000	7.500000	8.400000	8.800000	1.728400	1.047600
25%	120.000000	19.050000	21.000000	23.150000	5.944800	3.385650
50%	273.000000	25.200000	27.300000	29.400000	7.786000	4.248500
75%	650.000000	32.700000	35.500000	39.650000	12.365900	5.584500
max	1650.000000	59.000000	63.400000	68.000000	18.957000	8.142000

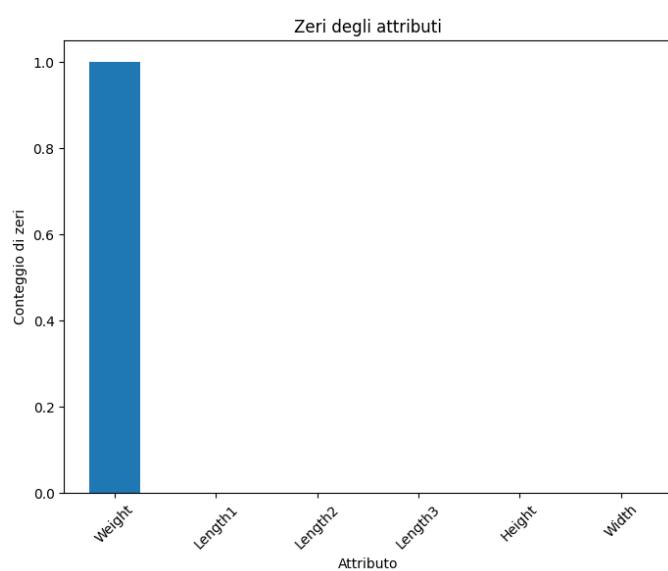
1.1 Conteggio delle specie di pesci:



Perch	56
Bream	35
Roach	20
Pike	17
Smelt	14
Parkki	11
Whitefish	6

Inizialmente abbiamo fatto un boxplot per vedere la distribuzione della specie, notiamo fin da subito che il dataset è un pò sbilanciato infatti i pesci presenti maggiormente sono il persico (perch) e l'orata (bream)

1.2 Analisi valori degli attributi nel dataset



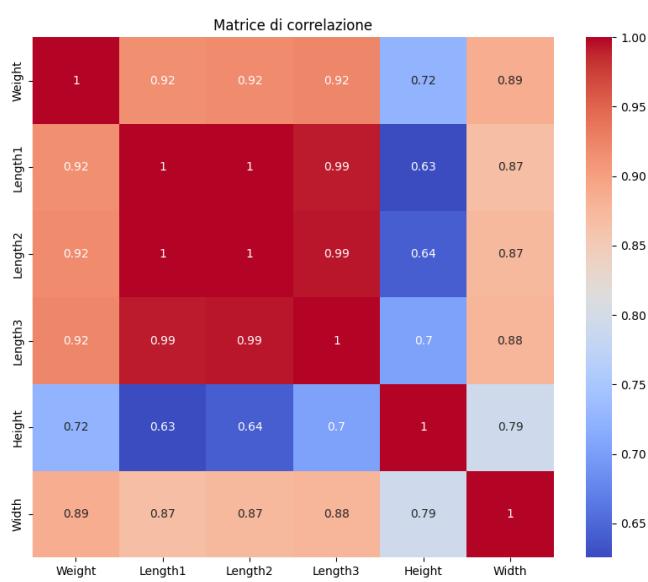
Attributi a zero: **Valori mancanti:**

Weight	1	Species	0
Length1	0	Weight	0
Length2	0	Length1	0
Length3	0	Length2	0
Height	0	Length3	0
Width	0	Height	0
		Width	0

Abbiamo poi fatto l'analisi dei valori mancanti e notiamo che non è presente nessun valore mancante tranne che per un pesce il valore del suo peso è a 0.

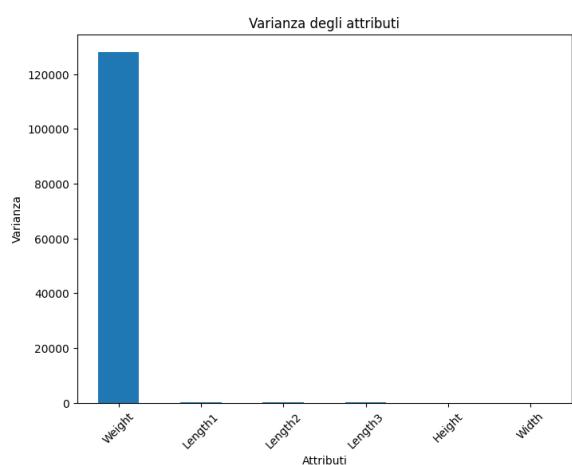
Il record in questione è il numero 41 [Roach, 0, 19, 20.5, 22.8, 6.4752, 3.3516], potrebbe essere un errore nella raccolta dei dati, dato che il peso è un attributo float, dunque si sarebbero potuti indicare anche i grammi.

1.3 Matrice di correlazione



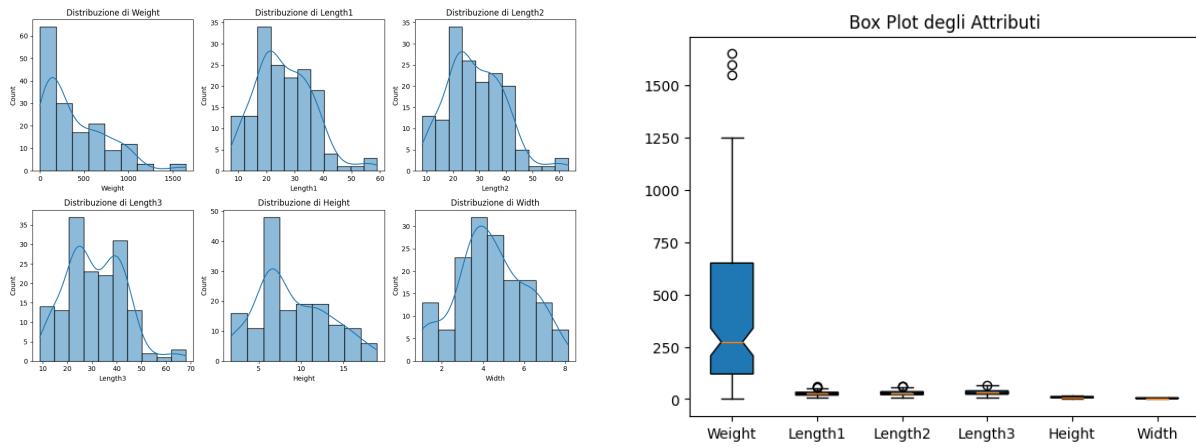
Nella matrice di correlazione viene misurata la relazione tra le varie variabili, notiamo che comunque le varie variabili tra loro hanno tutte una buona correlazione infatti non sono presenti valori molto vicino allo 0 o zero stesso, stanno tutte su un valore minimo di 0,6.

1.4 Distribuzione degli attributi



Abbiamo cercato di mostrare la distribuzione degli attributi, tramite il calcolo della varianza dei valori assunti dai singoli attributi, dal grafico a sinistra e dal BoxPlot notiamo che il peso è l'attributo che varia di più, e nasconde la scala di variazione delle altre variabili.

Gli altri attributi assumono valori tra 0 e un massimo di 70.



2. Classificatori utilizzati

Come prima cosa dobbiamo dire che il nostro task di identificazione delle specie di pesci a partire dai suoi attributi è un task di classificazione.

Questo è il codice della funzione che rende i classificatori

```
def get_classifiers():
    return [
        DecisionTreeClassifier(criterion="gini", max_depth=None,
                               min_samples_split=3, min_samples_leaf=1, max_features=None,
                               random_state=42),
        RandomForestClassifier(criterion='gini',max_depth=53),#prestazioni
        simili a quello sopra
        SVC(kernel = 'linear',C =3.0,probability=True),#acc 0.96875

        ensemble_hard,
        ensemble_soft,
        NaiveBayesClassifier(),
        MLPClassifier(activation= 'identity',early_stopping=
        False,hidden_layer_sizes= (50,),max_iter= 10000,momentum=
        0.9,nesterovs_momentum= True,random_state= 2,tol= 1e-06)]
```

Abbiamo utilizzato inizialmente tre classificatori base, un classificatore custom basato sul naive bayes e un classificatore ensemble personalizzato, che utilizza il soft voting in **ensemble_soft** e hard voting nel **ensemble_hard**.

Abbiamo poi utilizzato un'aggiunta extra di un altro classificatore MLP che abbiamo notato essere più che abbastanza preciso per il nostro task infatti ha ottenuto i valori massimi in accuracy,F1 score e precision, maggiori considerazioni di questo classificatore nel grafico di comparazione.

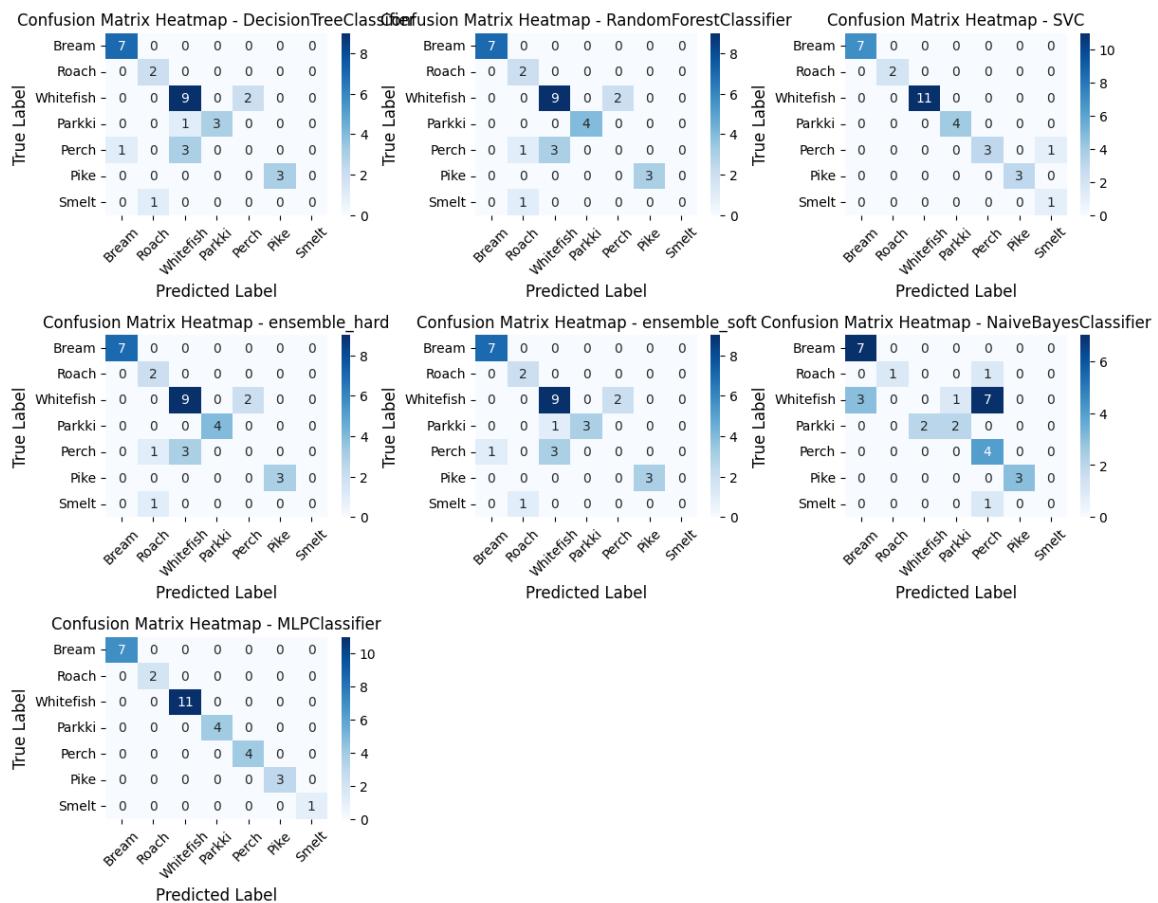
2.1 Confronto prestazioni classificatori SENZA standardizzazione dei dati:

Qui abbiamo confrontato i vari risultati ottenuti dai classificatori e notiamo che i classificatori base ottengono buoni risultati soprattutto l'svm che notiamo essere il migliore tra i classificatori base che ha ottenuto un'accuratezza di 0.96. Il peggiore risulta essere il classificatore custom basato su naive bayes che ha ottenuto solamente un'accuratezza del 0.53.

DecisionTree	ensemble_hard	NaiveBayes	SVC
Accuracy: 0.78	Accuracy: 0.81	Accuracy: 0.625	Accuracy: 0.96875
Precision: 0.85	Precision: 0.86	Precision: 0.636	Precision: 0.85
F1-score: 0.774	F1-score: 0.79	F1-score: 0.50460	F1-score: 0.84
RandomForest	ensemble_soft	MLP	
Accuracy: 0.81	Accuracy: 0.78	<u>Accuracy: 1.0</u>	
Precision: 0.75	Precision: 0.85	<u>Precision: 1.0</u>	
F1-score: 0.68	F1-score: 0.774	<u>F1-score: 1.0</u>	

2.2 Grafici confronto

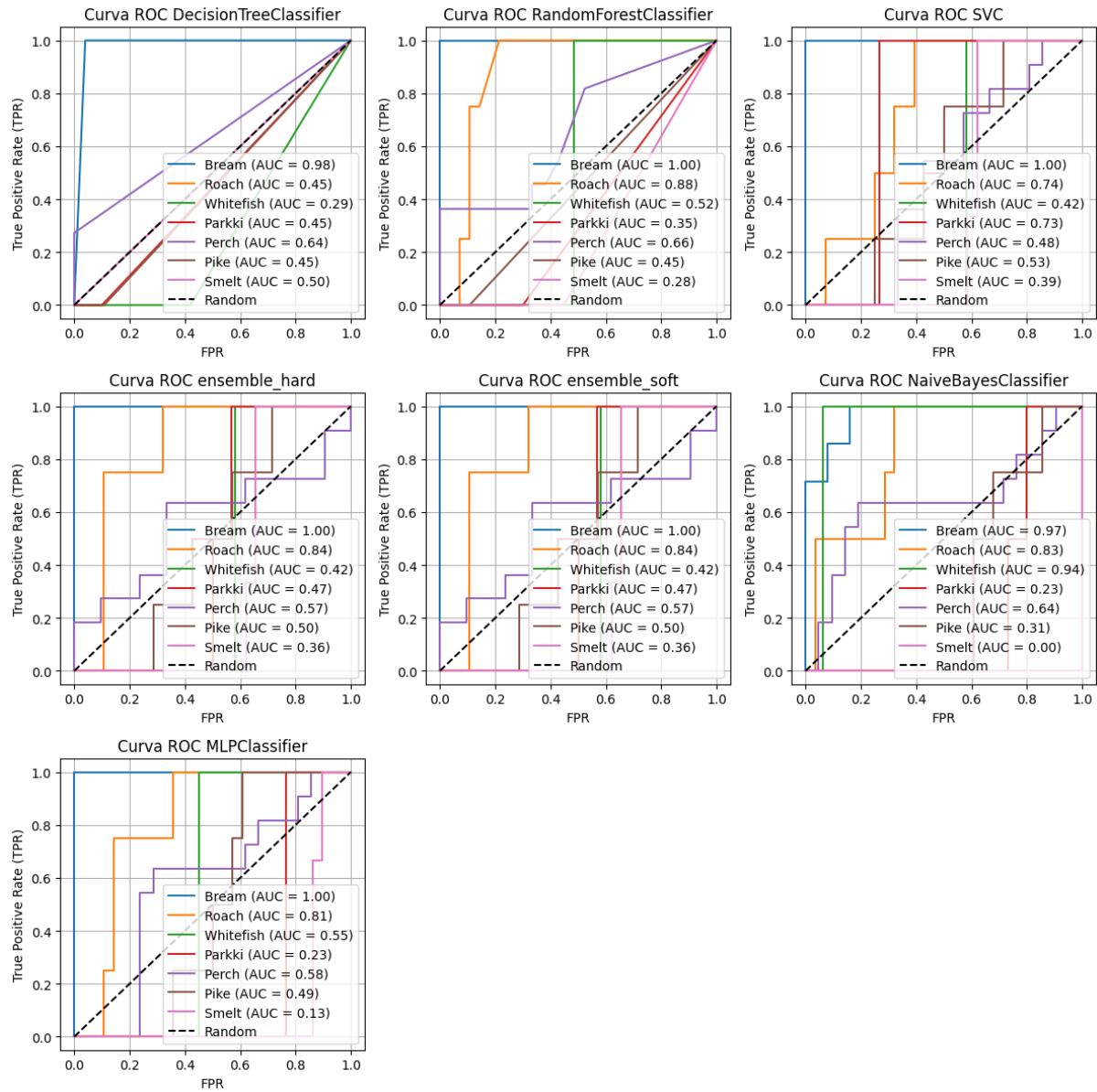
- **Confusion Matrix Heatmap**



Notiamo che con la matrice di confusione per i classificatori base il whitefish tende ad essere confuso con il persico (perch). Il classificatore che commette più errori in assoluto è il classificatore custom naive bayes che confonde il whitefish con il persico per 7 volte e commette degli errori anche su gli altri pesci. I classificatori migliori sono SVC che confonde solamente una volta il pesce persico(perch) con

lo smelt. Basandoci su questo grafico il classificatore migliore come detto in precedenza risulta essere MLP, che non commette nessun errore.

- **Curva Roc**



L'Area Under the ROC Curve (**AUC-ROC**) è un modo per valutare quanto un modello di classificazione binaria sia bravo a fare previsioni. La curva ROC mostra come il modello riesce a distinguere tra due diverse classi a diverse soglie di decisione.

L'AUC-ROC rappresenta un punteggio che va da 0 a 1, dove:

- Se l'AUC-ROC è 1, significa che il modello è perfetto e riesce a fare previsioni corrette in tutte le situazioni.
- Se l'AUC-ROC è 0.5, il modello non è migliore di una previsione casuale, come il lancio di una moneta.

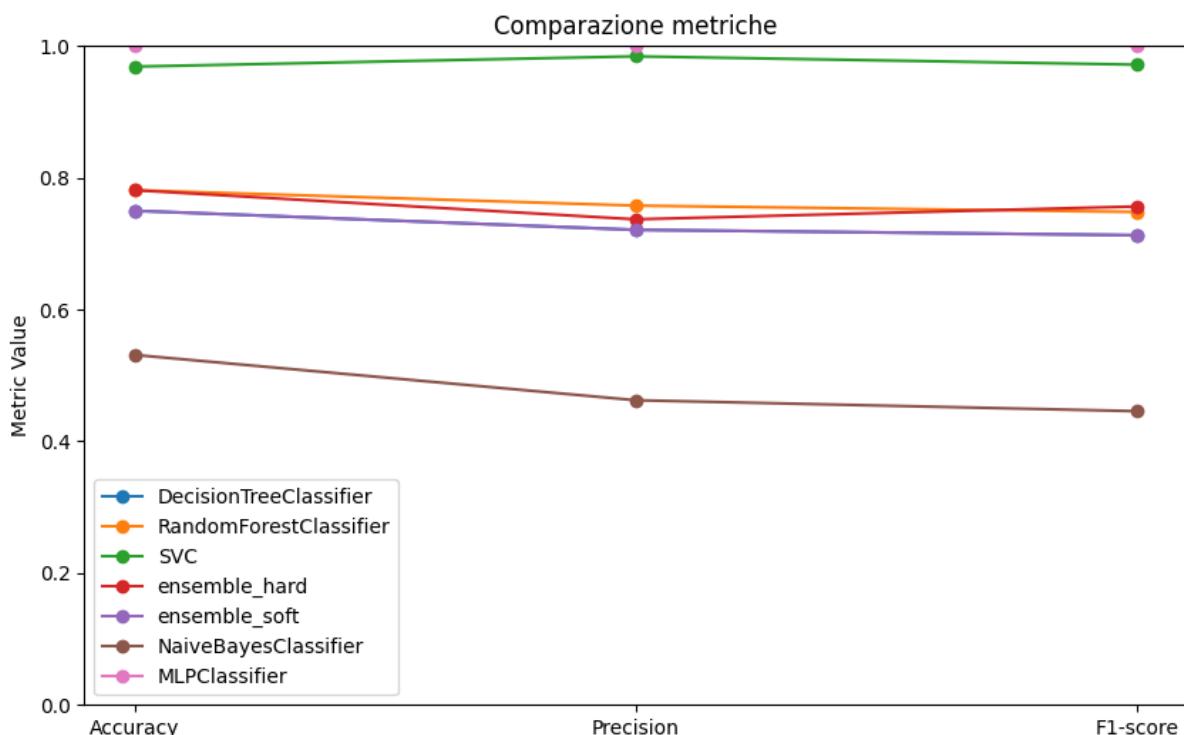
- Se l'AUC-ROC è compreso tra 0.5 e 1, il modello è migliore di una previsione casuale, ma quanto più ci si avvicina a 1, tanto migliore è il modello.

In sostanza, l'AUC-ROC ci dà un'idea di quanto sia buono il nostro modello nel fare previsioni corrette. Un valore vicino a 1 è il massimo obiettivo, mentre un valore vicino a 0.5 significa che il modello potrebbe non essere molto utile.

Per poter rappresentare la curva ROC nel nostro task multiclasse (e non binario) abbiamo utilizzato la conversione in etichette binarie per ciascuna classe con la codifica one-hot.

Detto ciò possiamo dire che tutti i modelli sono molto performanti nel riconoscere la classe maggioritaria del dataset Bream, ciò non accade invece con le altre classi, abbiamo AUC < 0.5 e in alcuni casi addirittura 0.13.

- **Confronto metriche**



Nota: metriche calcolate con un validation set, composto dal 20% del dataset

Come possiamo vedere tutti i modelli, eccetto il NaiveBayesClassifier, sono nella metà alta del grafico.

Il modello che ha performance migliori generali, un po' nascosto dai bordi del grafico, è il **MLP** classifier che raggiunge valore 1.0 in accuracy, Precision e F1-score, qua dobbiamo fare delle considerazioni:

Questi risultati indicano che il modello ha imparato perfettamente dai dati di addestramento e ha ottenuto prestazioni eccellenti sul validation set.

Tuttavia, è importante considerare il fatto che ottenere una precisione del 100% sul validation set potrebbe essere un segnale di overfitting. L'overfitting si verifica quando il modello si adatta troppo ai dati di addestramento, memorizzandoli invece di generalizzare i modelli per nuovi dati, di conseguenza, il modello potrebbe non essere in grado di generalizzare bene su nuovi dati sconosciuti e potrebbe mostrare prestazioni scadenti su un test set o su dati del mondo reale, dato il dataset molto piccolo

potrebbe essere possibile, maggiori considerazioni su questo nella parte di oversampling e cross validation.

SVC è il secondo modello migliore, poi notiamo che **ensemble_hard** e **ensemble_soft** hanno performance molto simili, infine **NaiveBayes** è quello che ha performance minori di predizione.

3. Confronto con e senza scaling dei dati

Abbiamo poi utilizzato la tecnica dello scalamento dei dati questo permette il rimessionamento delle dimensioni dei dati in modo che siano tutti sullo stesso intervallo. Questa tecnica può migliorare l'accuratezza dei modelli.

Noi abbiamo utilizzato 3 diverse tecniche di ridimensionamento: standardscaler, normalizzazione e minMax scaler.

- **DecisionTreeClassifier**

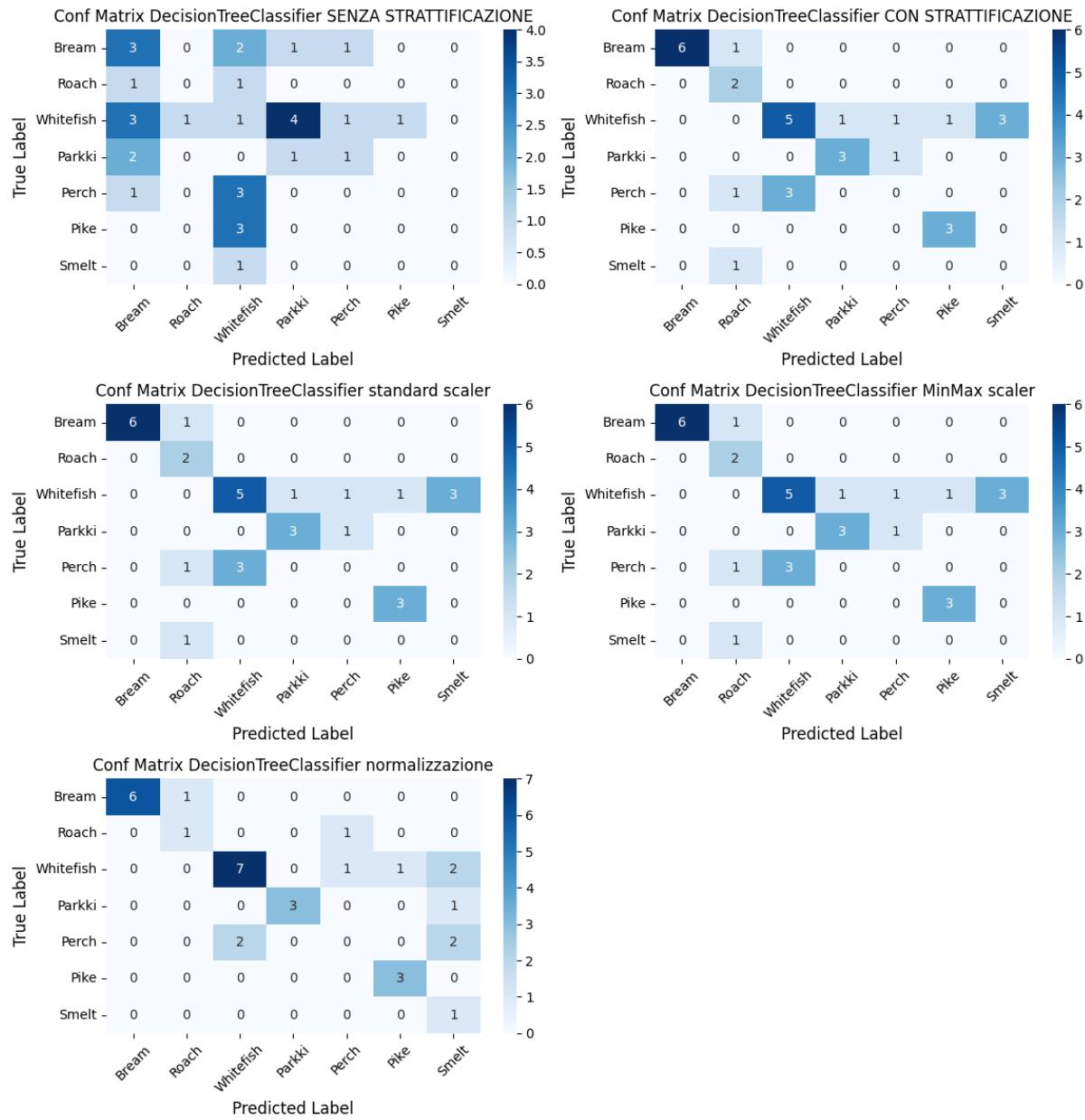
Accuratezza del DecisionTree sul dataset originale SENZA STRATTIFICAZIONE, 0.78125

Accuratezza del DecisionTree sul dataset originale CON STRATTIFICAZIONE, 0.59375

Accuratezza del DecisionTree dopo standard scaler 0.59375

Accuratezza del DecisionTree dopo MinMax scaler 0.59375

Accuratezza del DecisionTree dopo normalizzazione 0.65625



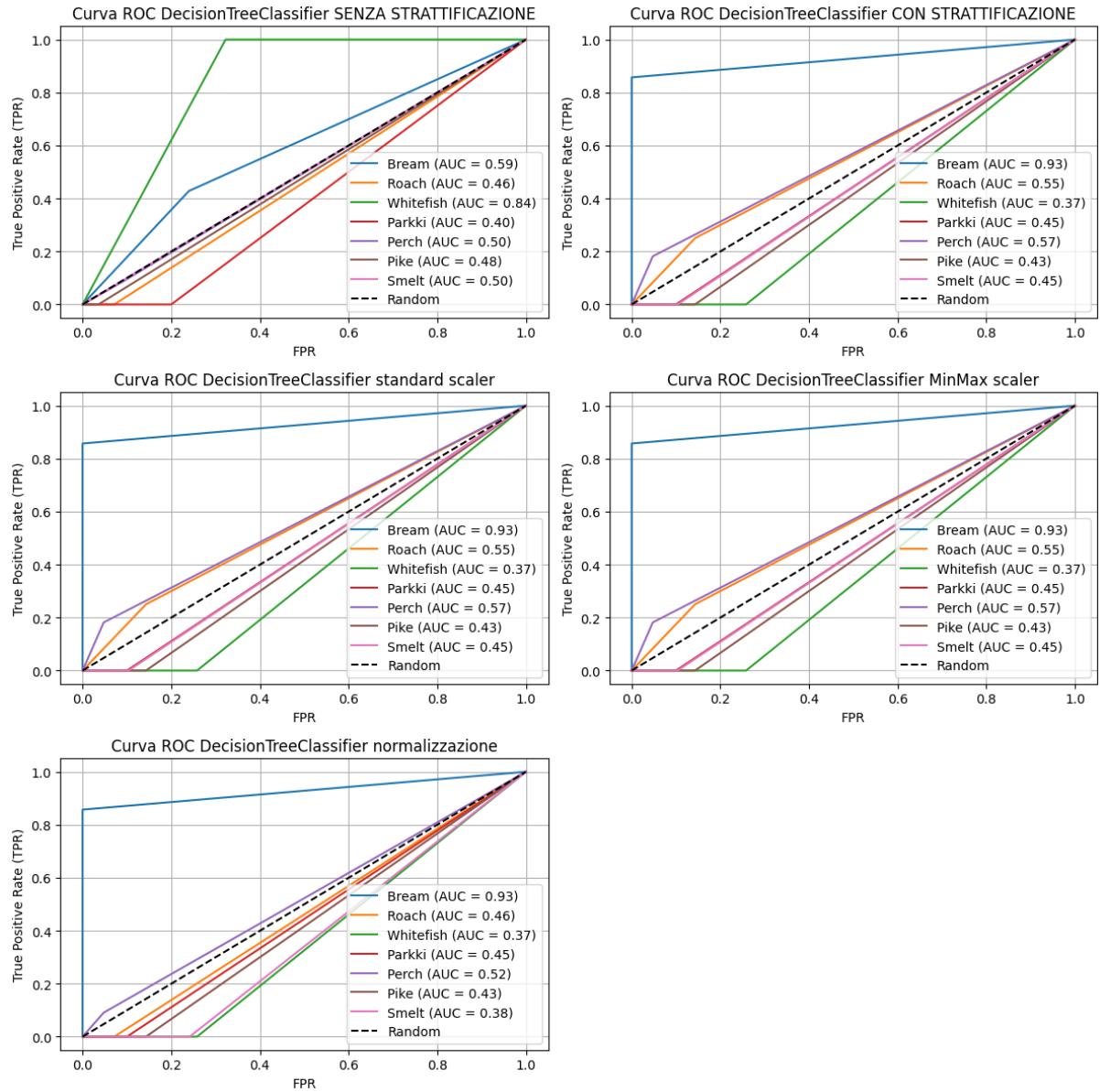
La **stratificazione** viene utilizzata per mantenere una distribuzione bilanciata delle classi nel training set e nel test set, il che potrebbe portare a risultati più affidabili in termini di generalizzazione.

In questo caso, abbiamo un decremento della accuratezza, sembra che la stratificazione non abbia migliorato le prestazioni del modello, invece guardando la matrice di confusione abbiamo un miglioramento, basti guardare le “diagonali” della matriche con TrueLabel = Predicted Label, notiamo anche che la classe minoritaria WhiteFish passa da 1 predizione corretta a 5 .

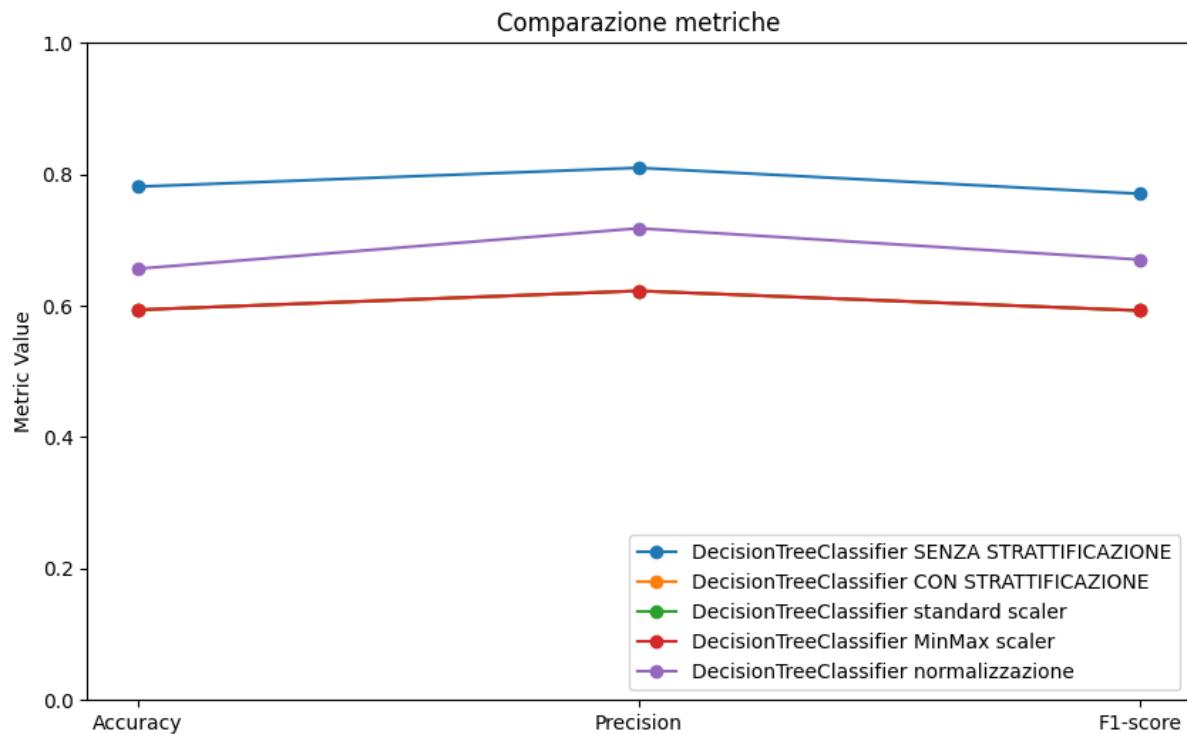
L'applicazione del Standard Scaler ha normalizzato i dati, rendendoli con media zero e varianza unitaria, tuttavia in questo caso, sembra che la normalizzazione non abbia avuto un impatto significativo rispetto alla strattificazione, come anche il MinMaxScaler, hanno esattamente la medesima matrice di confusione.

Invece la normalizzazione ha portato sempre a risultati inferiori rispetto al non strattificato, ma comunque superiori a strattificato, MinMaxScaler e StandardScaler.

- Curva Roc



Notiamo che le classi maggioritarie hanno non hanno un ampia variazione del AUC, invece la classe minoritaria WhiteFish sembra averne risentito di più, probabilmente nel caso senza stratificazione erano presenti la maggior il numero di record nel train set e pochi molto simili nel test set, ed essa risente anche della scalatura dei dati.



Sulla base dei valori mostrati prima sappiamo che lo standard scaler, MinMax scaler e lo strattificato stanno sulla stessa curva, notiamo che abbiamo performance migliori senza strattificazione

- **RandomForest**

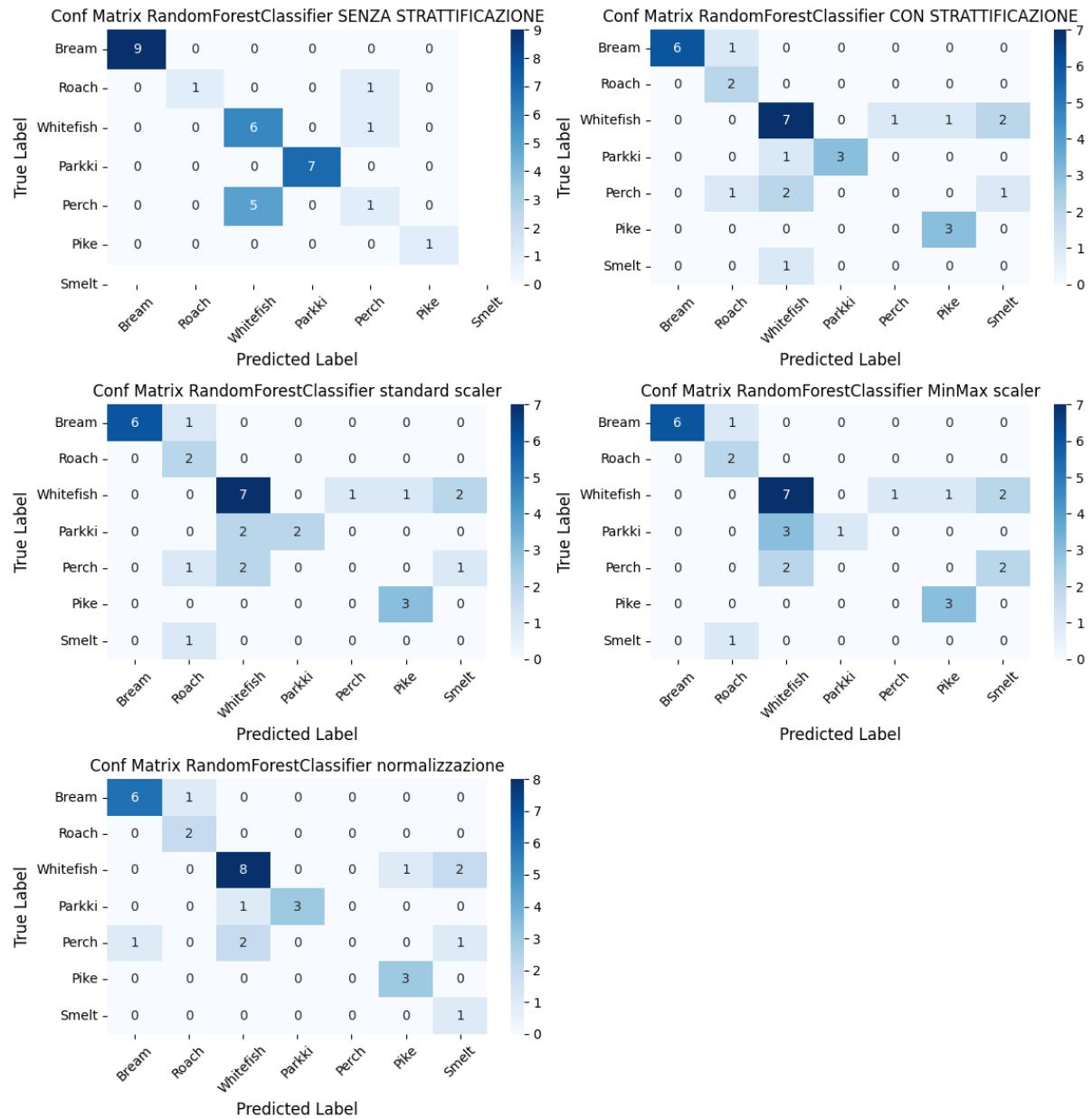
Accuratezza del RandomForest sul dataset originale SENZA STRATTIFICAZIONE, 0.8125

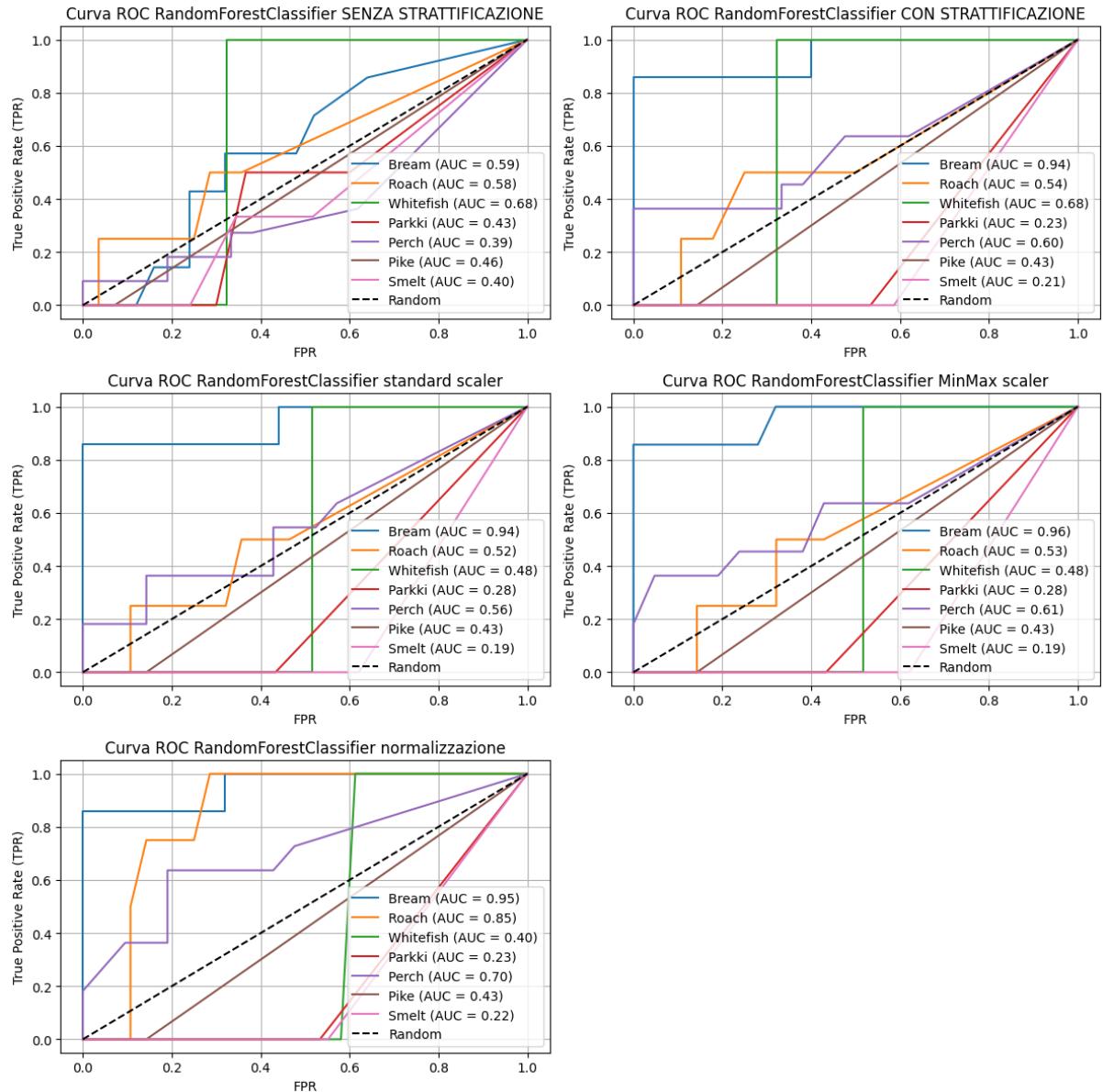
Accuratezza del RandomForest sul dataset originale CON STRATTIFICAZIONE, 0.65625

Accuratezza del RandomForest dopo standard scaler 0.625

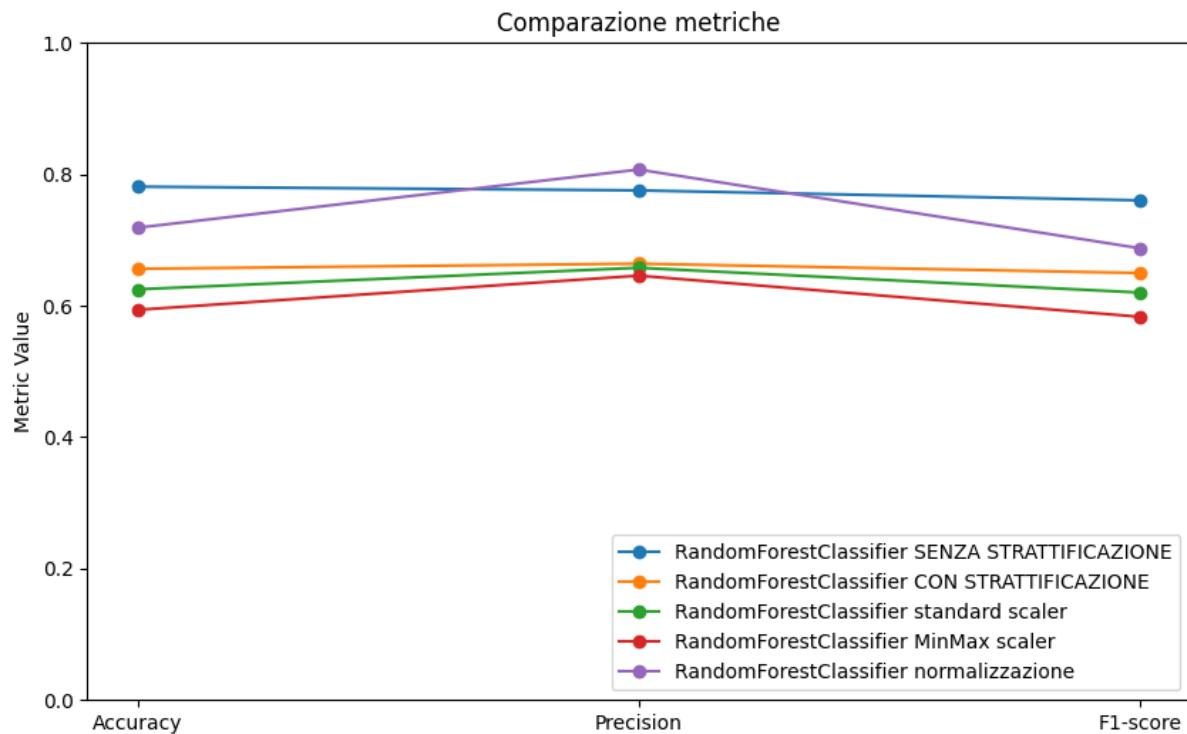
Accuratezza del RandomForest dopo MinMax scaler 0.65625

Accuratezza del RandomForest dopo normalizzazione 0.6875





Notiamo che applicare metodi di scalatura e/o normalizzazione porta a migliori predizioni per la classe Breame Perch, a discapito delle classi Parkki e Smelt, un po' anche Pike.



Il modello senza strattificazione porta risultati bilanciati tra le metriche, è comunque buono anche con normalizzazione.

- **SVC**

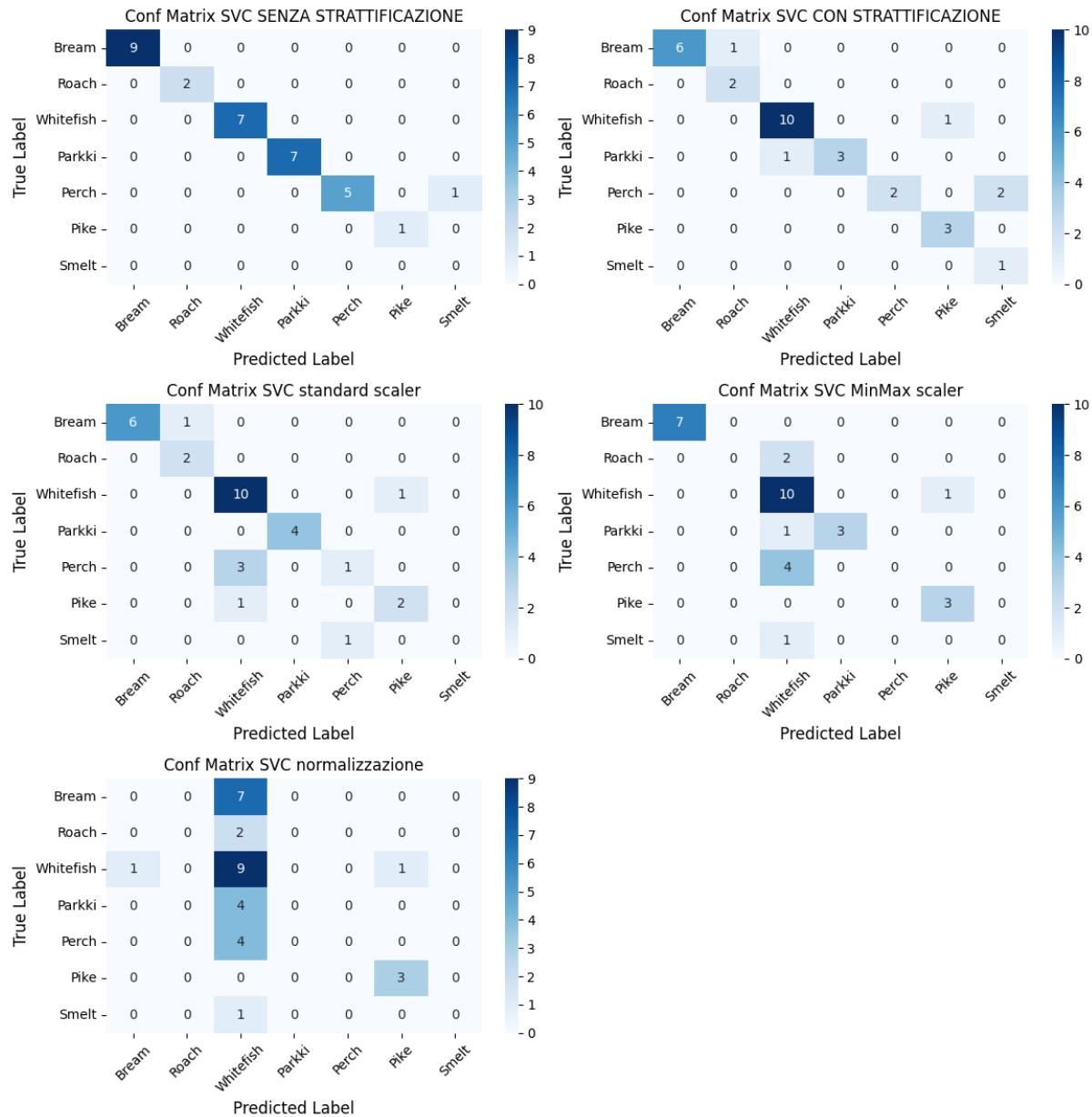
Accuratezza del SVC sul dataset originale SENZA STRATTIFICAZIONE, 0.96875

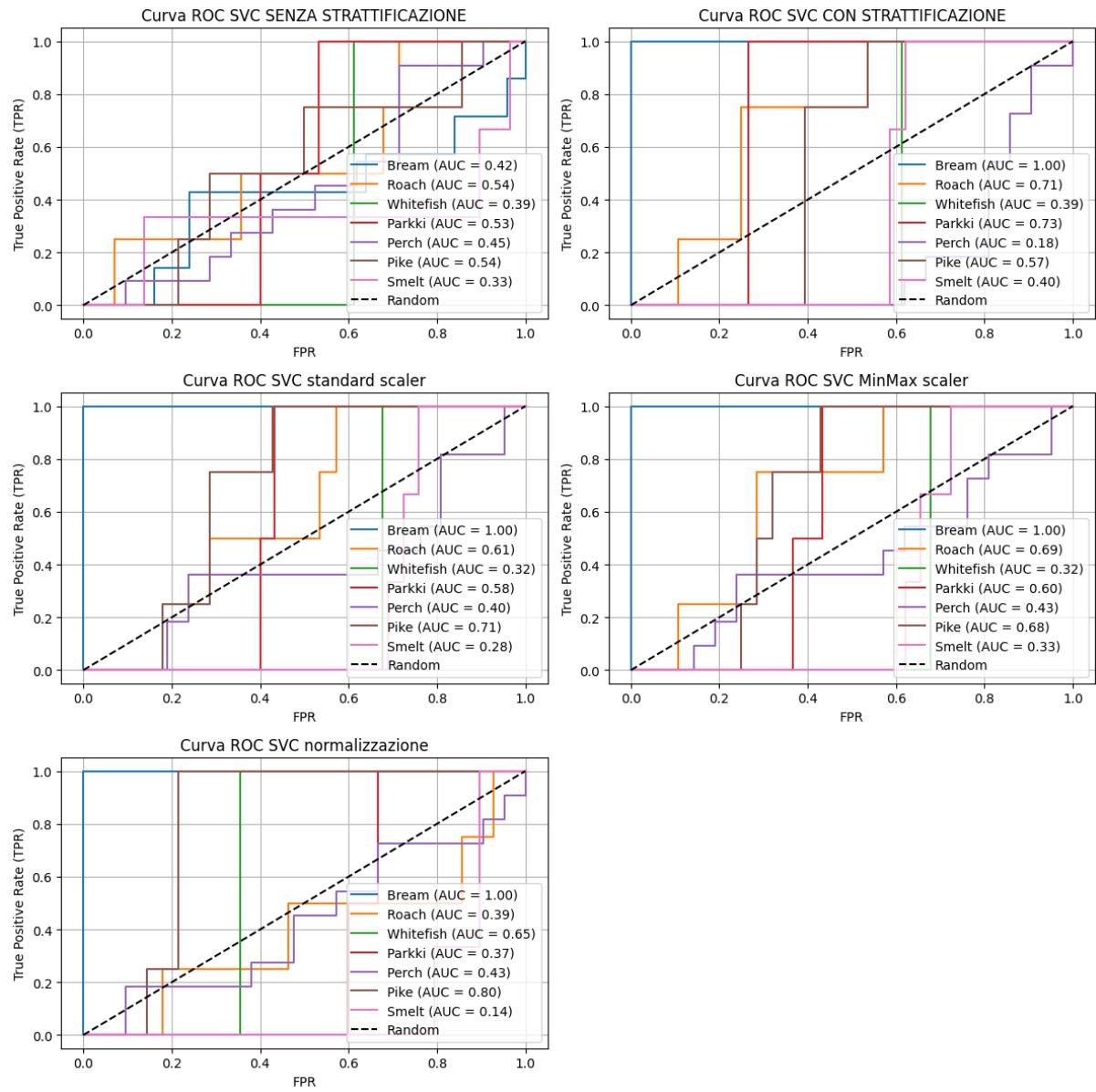
Accuratezza del SVC sul dataset originale CON STRATTIFICAZIONE, 0.84375

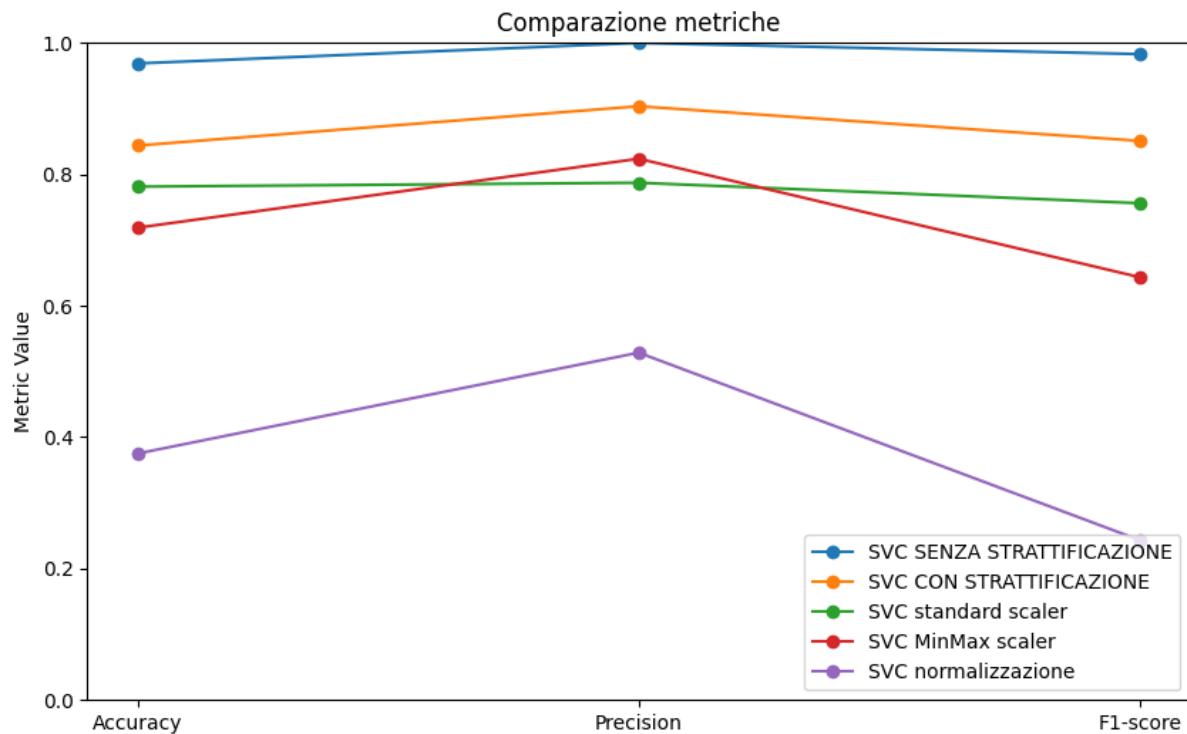
Accuratezza del SVC dopo standard scaler 0.78125

Accuratezza del SVC dopo MinMax scaler 0.71875

Accuratezza del SVC dopo normalizzazione 0.375







Anche qui il modello migliore è senza strettificazione

- **ensemble_hard**

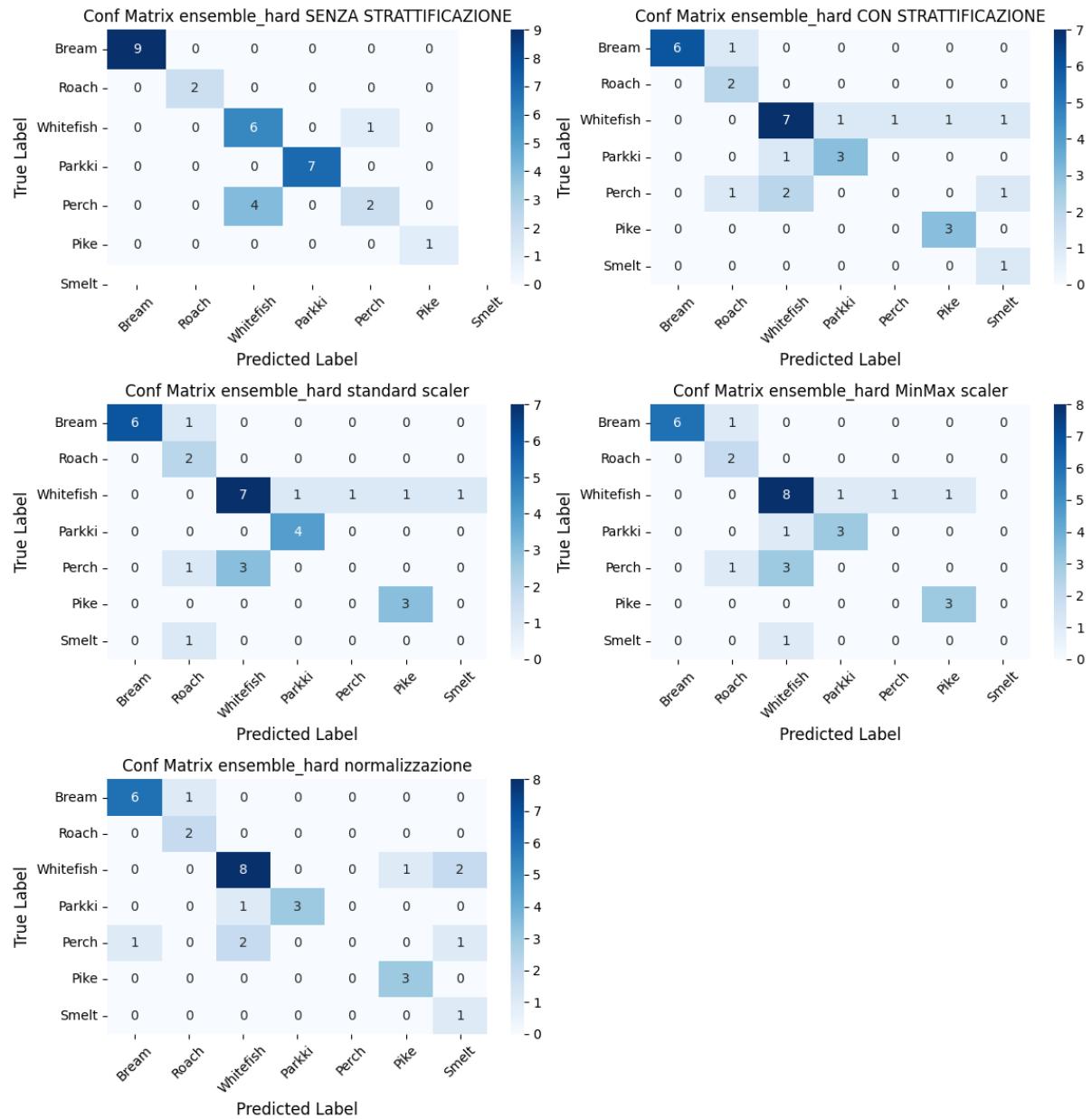
Accuratezza del ensemble_hard sul dataset originale SENZA STRATTIFICAZIONE, 0.8125

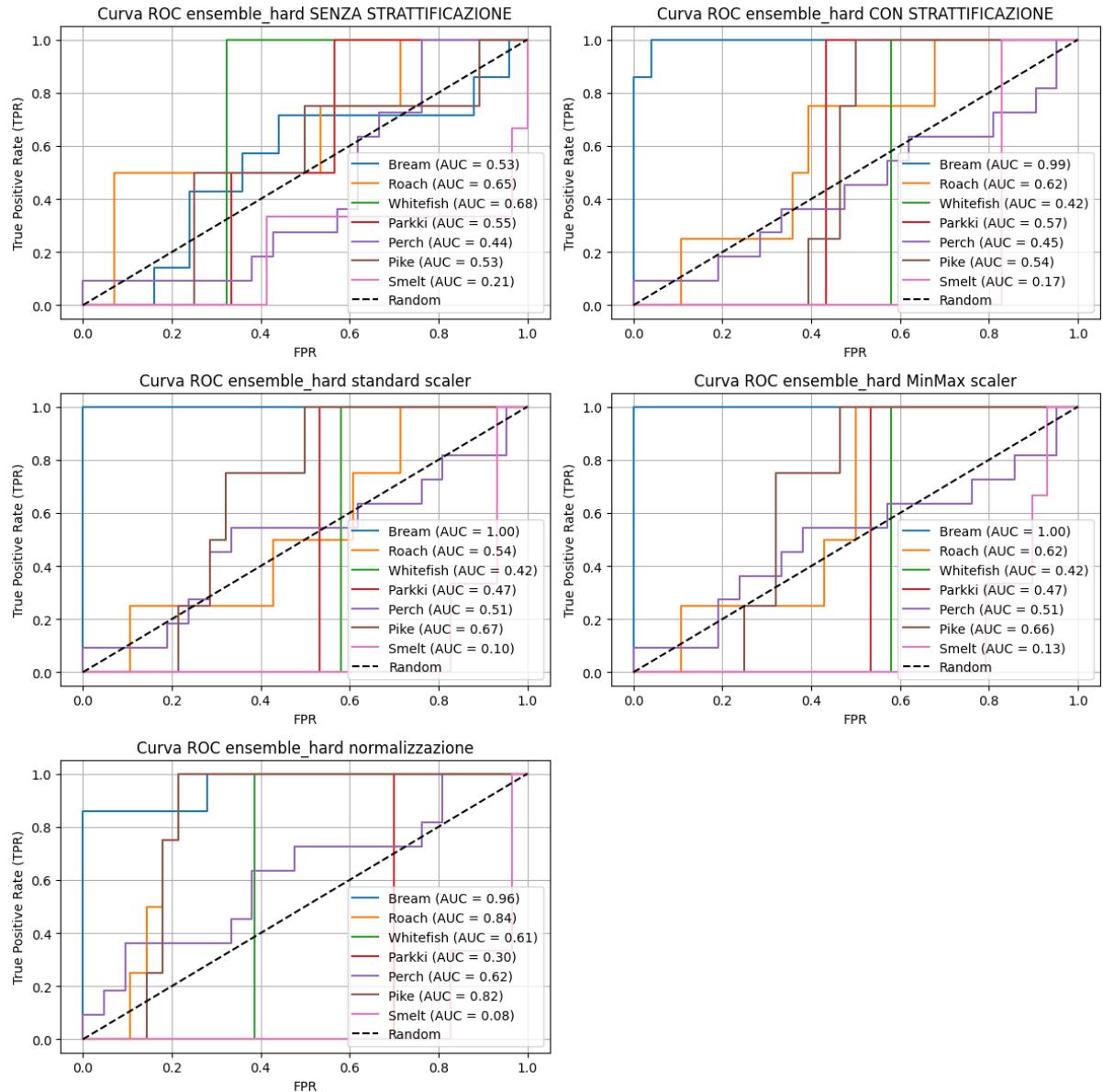
Accuratezza del ensemble_hard sul dataset originale CON STRATTIFICAZIONE, 0.65625

Accuratezza del ensemble_hard dopo standard scaler 0.6875

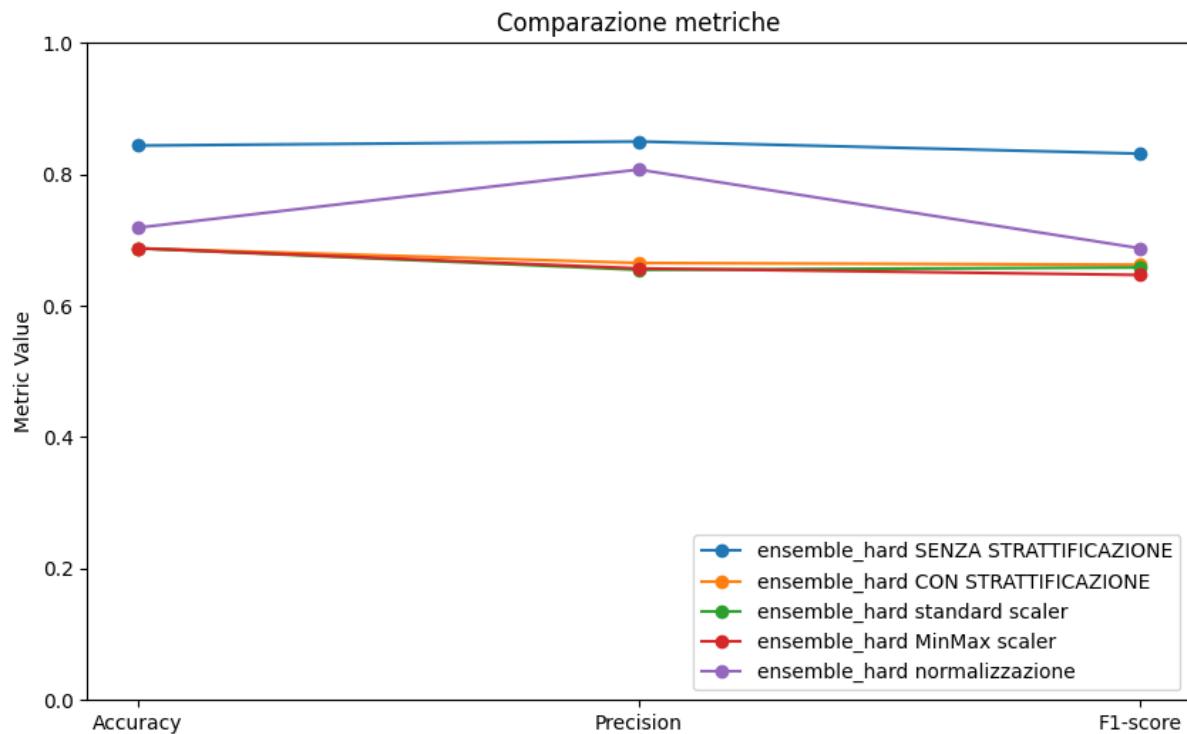
Accuratezza del ensemble_hard dopo MinMax scaler 0.65625

Accuratezza del ensemble_hard dopo normalizzazione 0.6875





Notiamo che applicare metodi di scalatura e/o normalizzazione porta a migliori predizioni per alcune classi, a discapito delle altre, si conferma migliore senza strattificazione.



- **ensemble_soft**

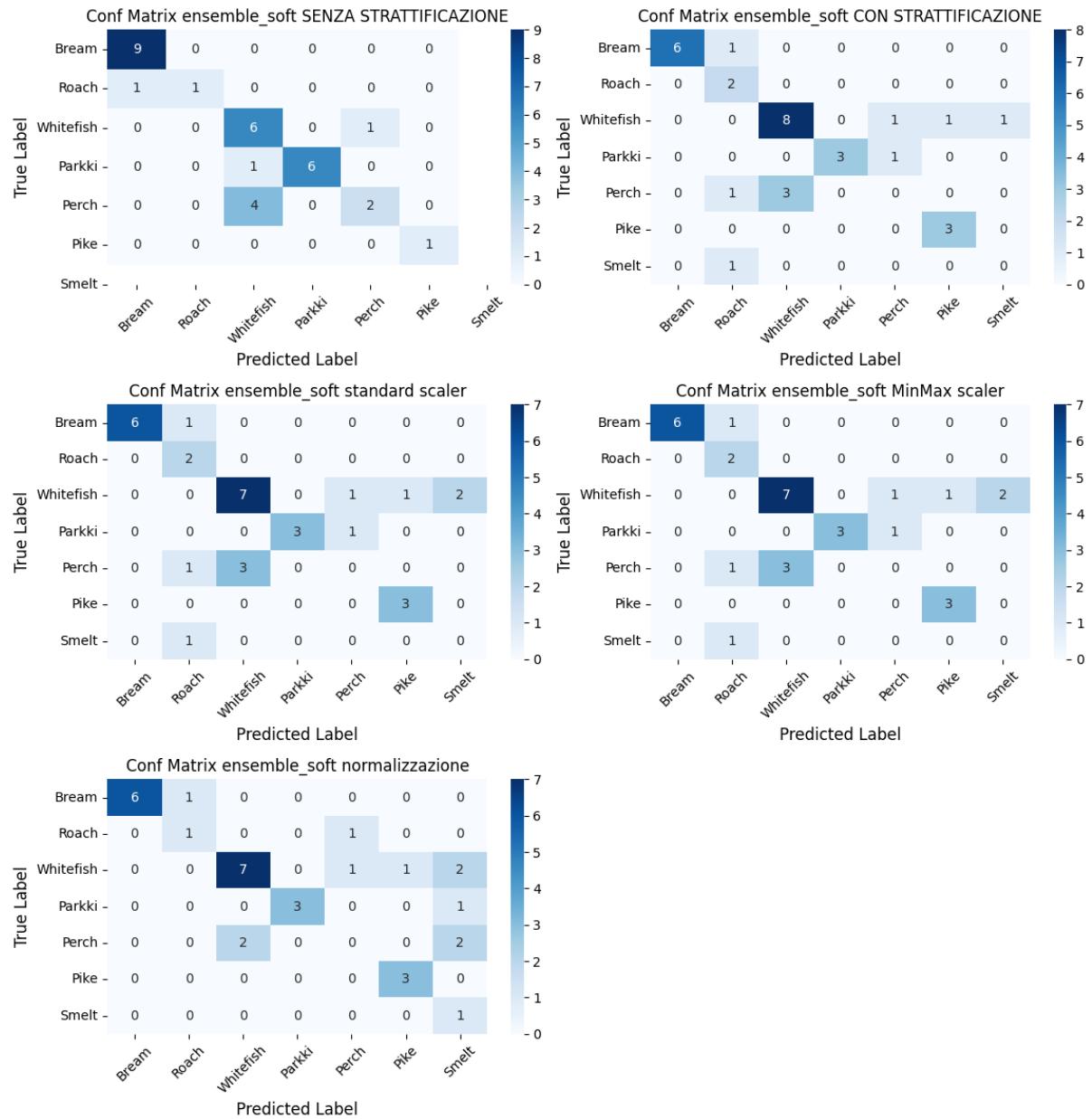
Accuratezza del ensemble_soft sul dataset originale SENZA STRATTIFICAZIONE, 0.78125

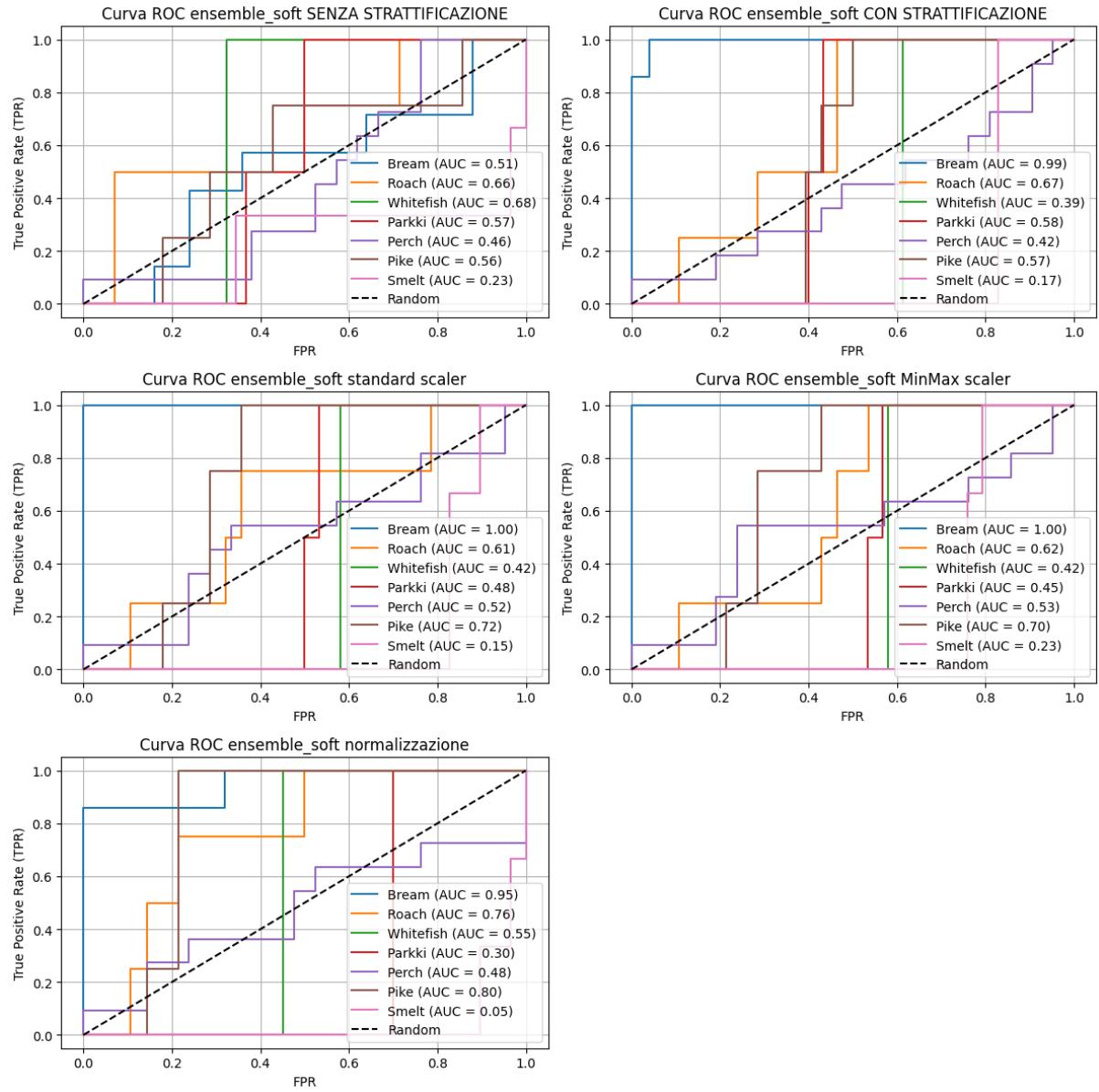
Accuratezza del ensemble_soft sul dataset originale CON STRATTIFICAZIONE, 0.65625

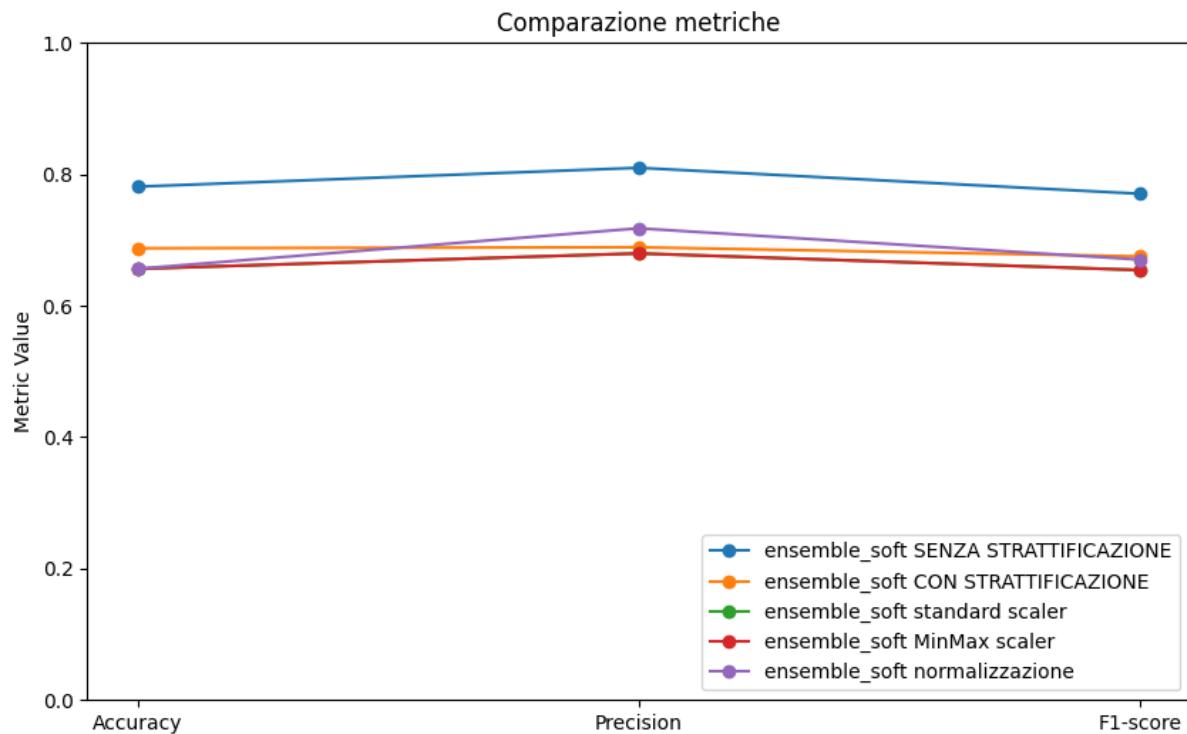
Accuratezza del ensemble_soft dopo standard scaler 0.65625

Accuratezza del ensemble_soft dopo MinMax scaler 0.65625

Accuratezza del ensemble_soft dopo normalizzazione 0.65625







- **NaiveBayesClassifier**

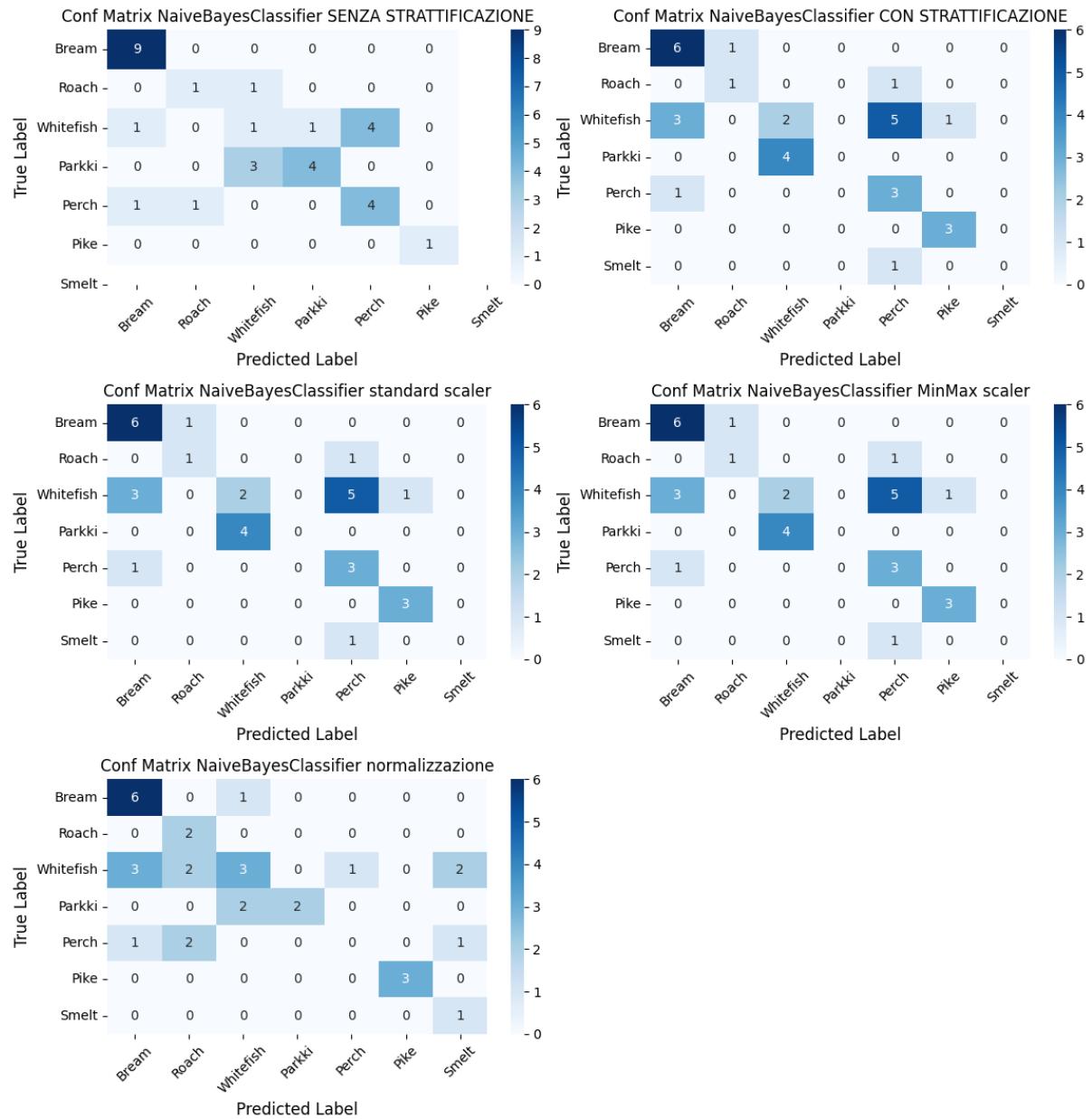
Accuratezza del NaiveBayesClassifier sul dataset originale SENZA STRATTIFICAZIONE, 0.625

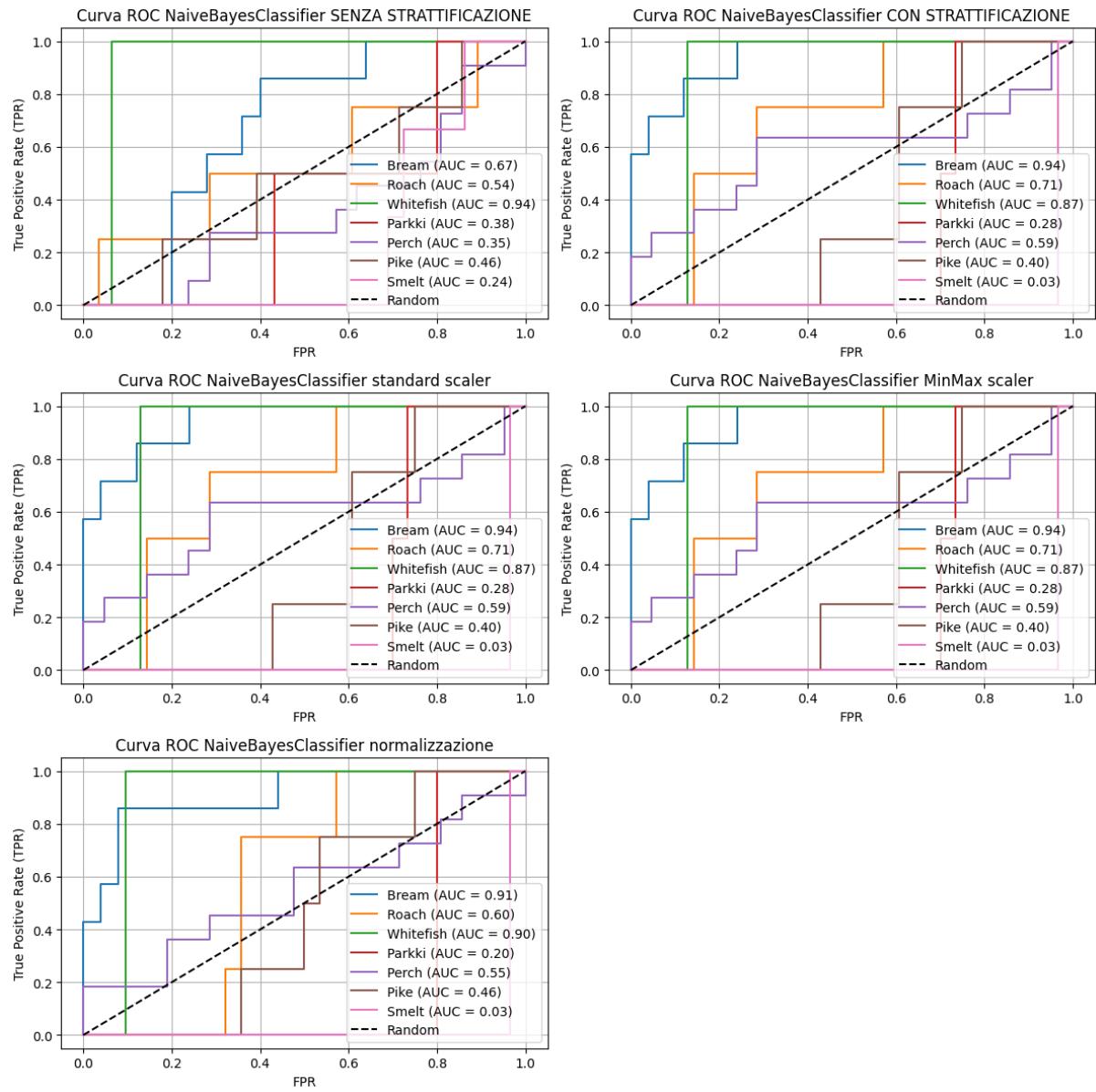
Accuratezza del NaiveBayesClassifier sul dataset originale CON STRATTIFICAZIONE, 0.46875

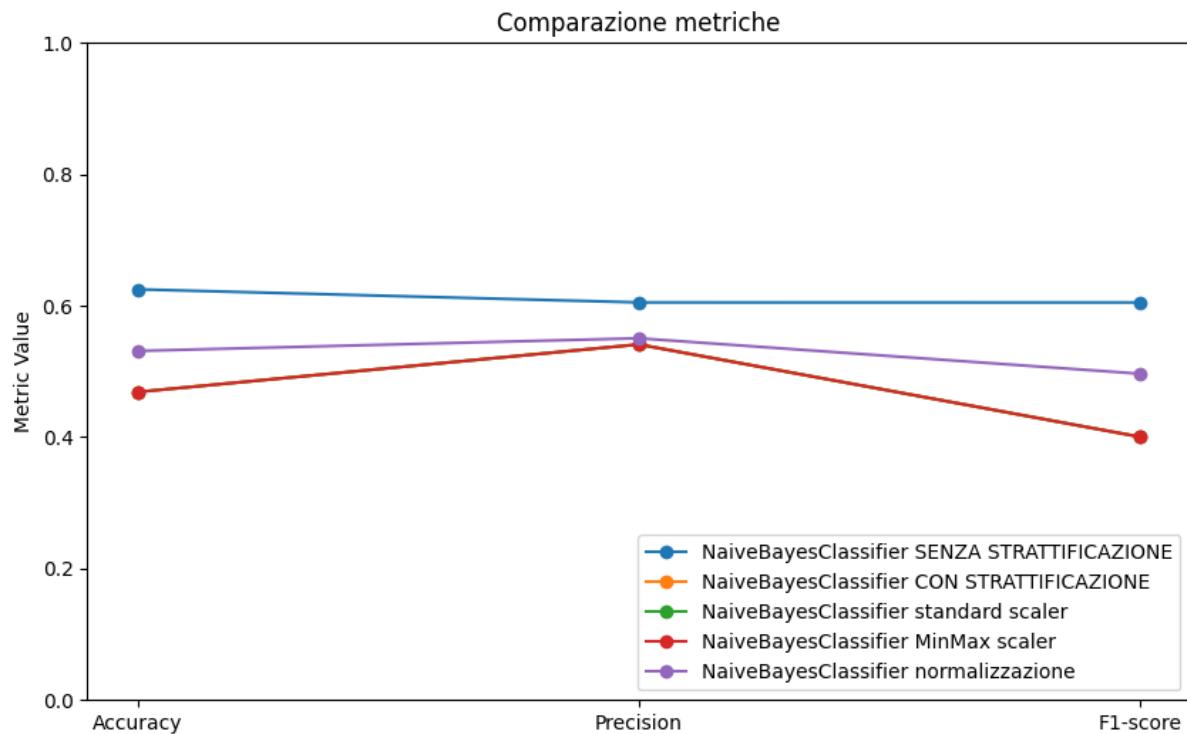
Accuratezza del NaiveBayesClassifier dopo standard scaler 0.46875

Accuratezza del NaiveBayesClassifier dopo MinMax scaler 0.46875

Accuratezza del NaiveBayesClassifier dopo normalizzazione 0.53125







- **MLPClassifier**

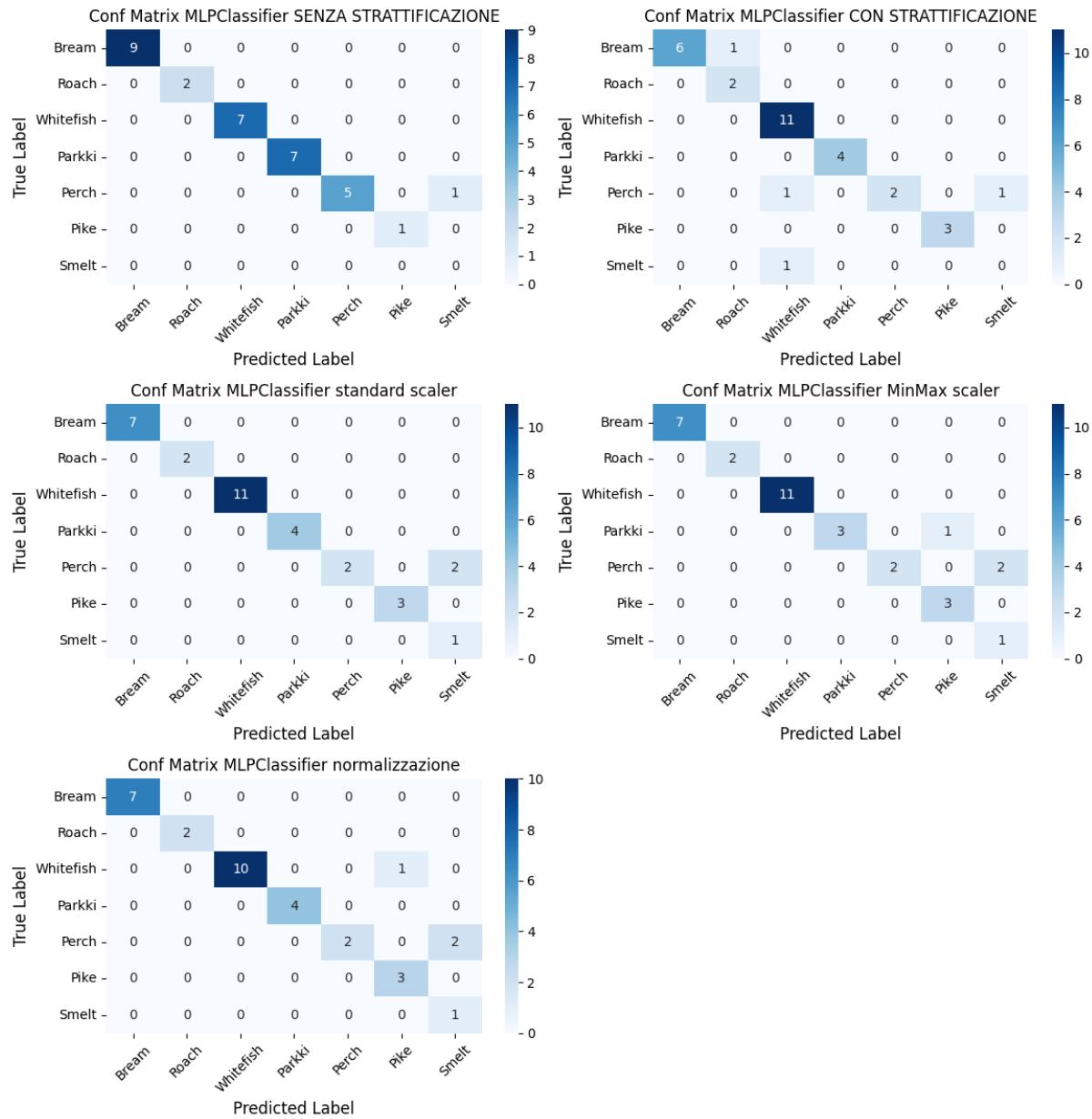
Accuratezza del MLP sul dataset originale SENZA STRATTIFICAZIONE, 0.96875

Accuratezza del MLP sul dataset originale CON STRATTIFICAZIONE, 0.875

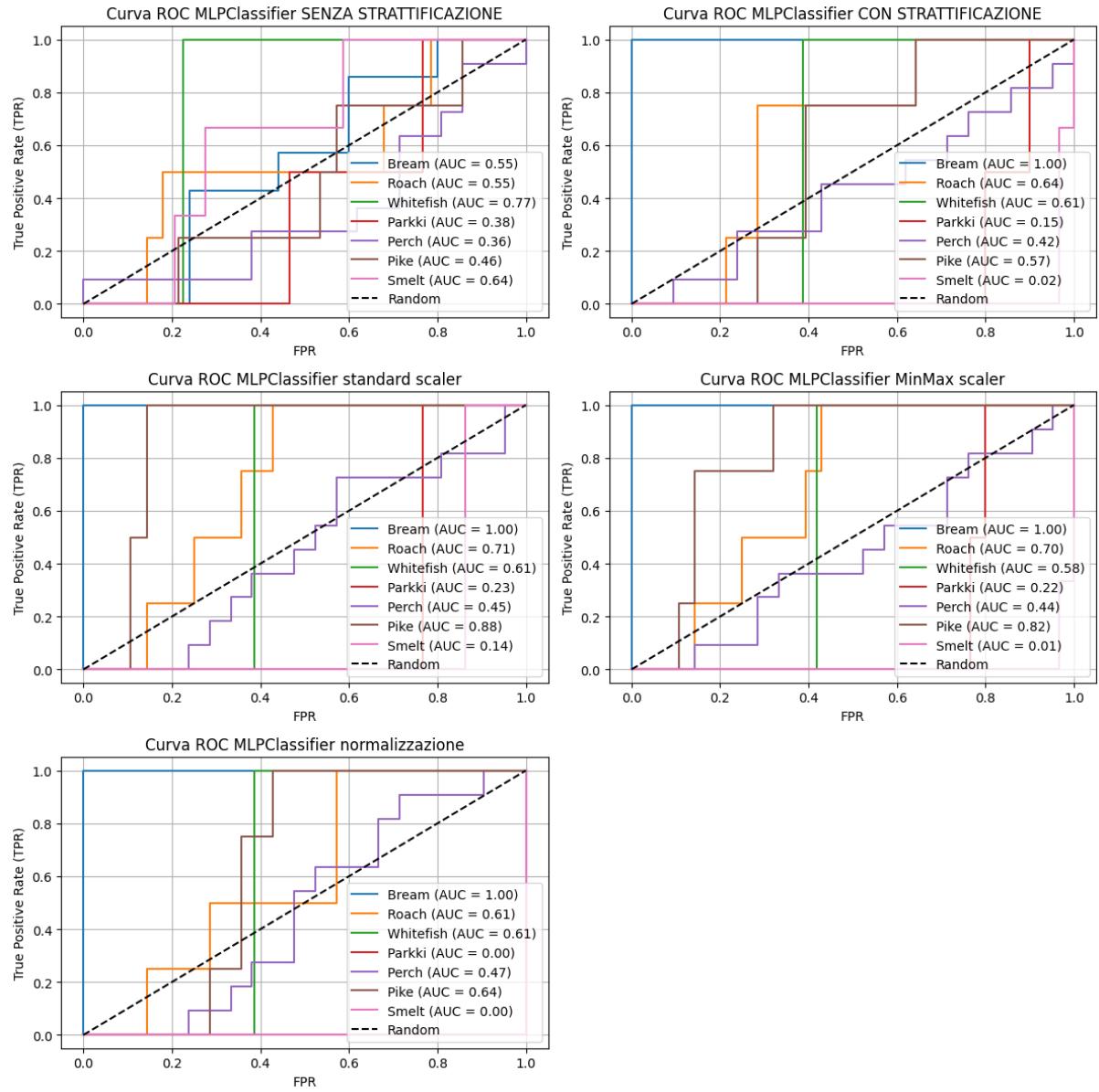
Accuratezza del MLP dopo standard scaler 0.9375

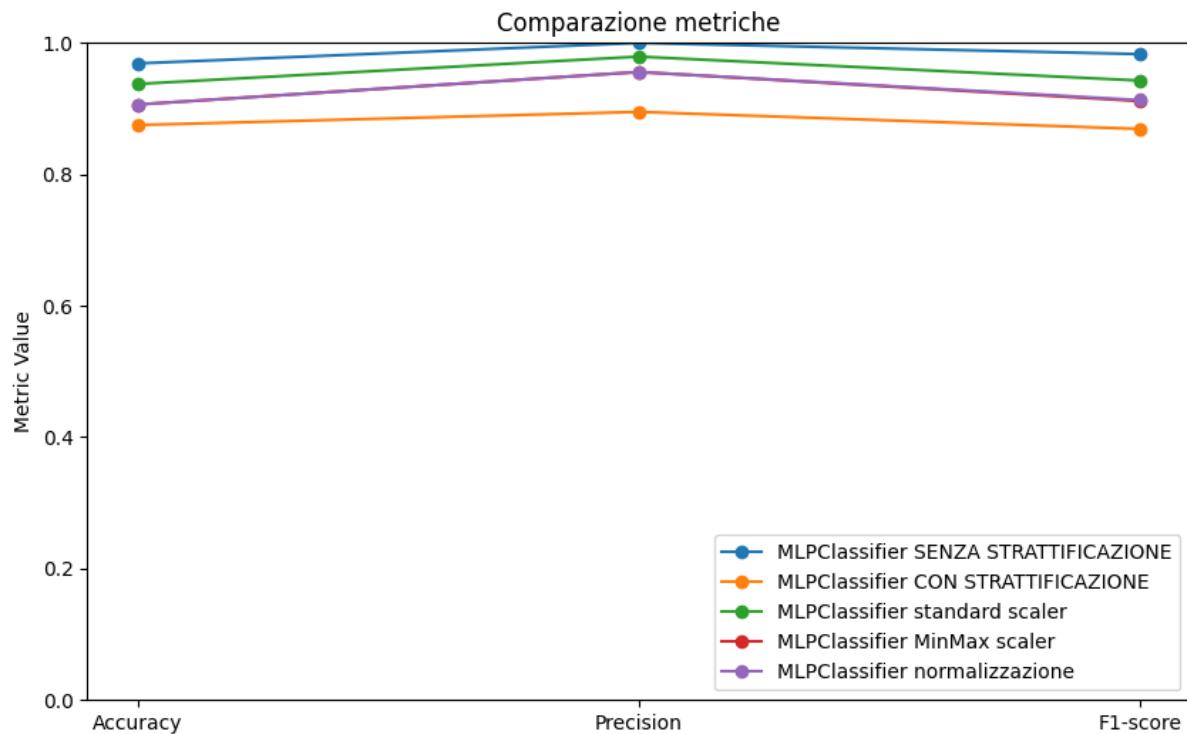
Accuratezza del MLP dopo MinMax scaler 0.90625

Accuratezza del MLP dopo normalizzazione 0.90625



Qua notiamo una cosa interessante, avevamo tutte le metriche a 1 prima su questo modello, (era strattificato come parametro della `train_test_split`) abbiamo scoperto che dividendo il dataset con un `random_state=2` porta ad avere accuracy, F1 score e Precision a 1, con `random_state=0` abbiamo questi risultati, abbiamo comunque deciso di mostrare con `random_state=0` perché mostra una variazione dei risultati.





In generale su tutti i modelli notiamo che applicare metodi di scalatura e/o normalizzazione porta a migliori predizioni per alcune classi, a discapito delle altre, si conferma migliore sempre il modello senza strattificazione.

4. Confronto con e senza oversampling

Dataset bilanciato

```
['Bream', 'Parkki', 'Perch', 'Pike', 'Roach', 'Smelt',
'Whitefish'],
array([45, 45, 45, 45, 45, 45, 45]))
```

Dataset bilanciato Smote

```
['Bream', 'Parkki', 'Perch', 'Pike', 'Roach', 'Smelt',
'Whitefish'],
array([56, 56, 56, 56, 56, 56, 56]))
```

DecisionTree

Accuracy: 0.75
Score Cross Val: 0.930
Score: 0.75
Accuracy: 0.75
score Smote + Cross Val: 0.80357

RandomForest

Accuracy: 0.78125
Score Cross Val:
0.91427
Score: 0.781
Accuracy: 0.78125
score Smote + Cross Val: 0.821

SVC

Accuracy: 0.96875
Score Cross Val:
0.9873
Score: 0.96875
Accuracy: 0.96875
score Smote + Cross Val: 0.969

ensemble_hard

Accuracy: 0.78125
Score Cross Val: 0.955
Score: 0.78125
Accuracy: 0.78125
score Smote + Cross Val: 0.852

ensemble_soft

Accuracy: 0.75
Score with Cross Val: 0.93971
Score: 0.75

NaiveBayes

Accuracy: 0.53125
Score Cross Val: 0.5616
Score: 0.53125

MLP

Accuracy: 1.0
Cross Val: 0.5141
Score: 1.0
Accuracy: 1.0

Accuracy: 0.75

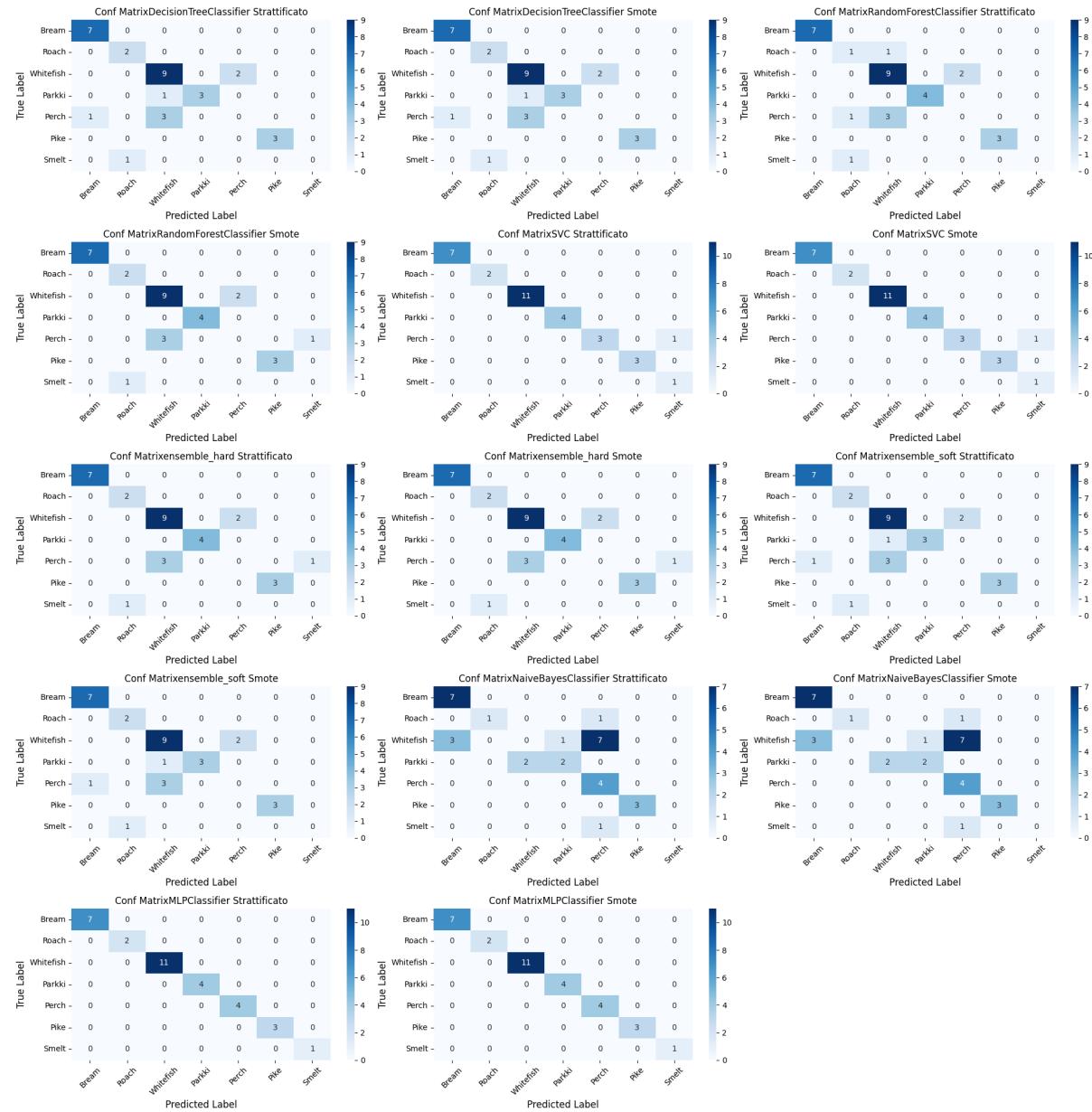
score Smote + Cross Val: 0.852

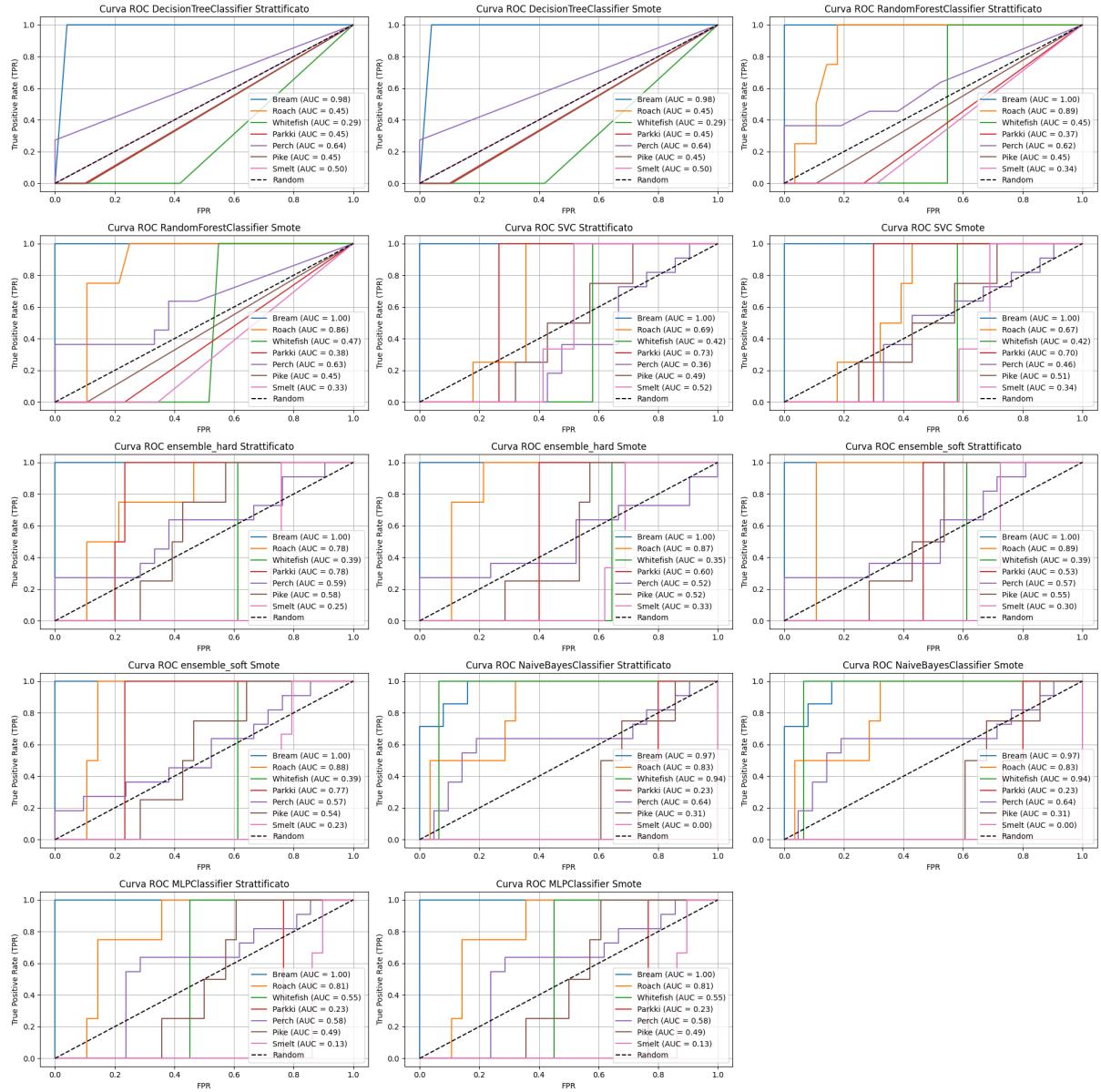
Accuracy: 0.53125

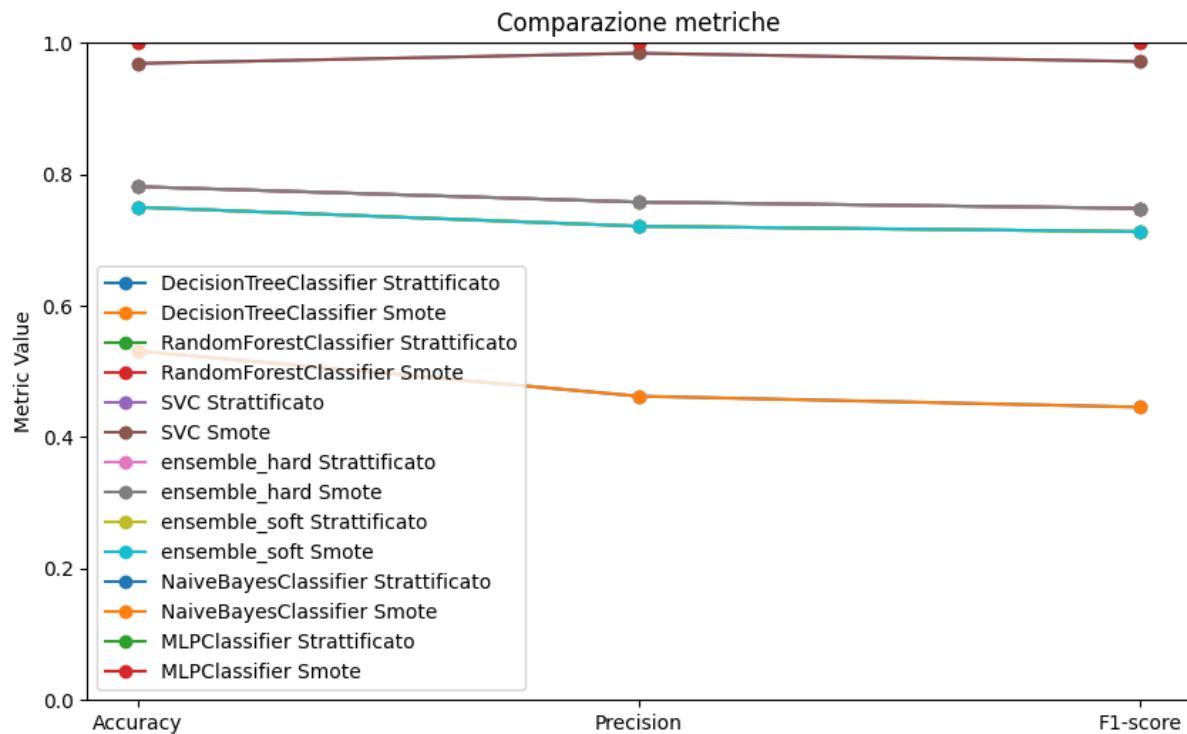
score Smote + Cross Val: 0.592

score Smote + Cross

Val: 0.7985







Notiamo che l'oversampling porta a risultati superiori, abbiamo anche fatto il calcolo della cross validation per vedere se c'erano dei problemi col modello, vediamo che MLP in effetti non era il migliore(potrebbero essere necessarie maggiori verifiche con un altro test set per confermare), perché con molta probabilità va in overfitting.

5. Confronto con e senza feature selection

Classificatore: DecisionTreeClassifier

Numero di feature selezionate: 1

Feature selezionate: ['Height']

Accuratezza: 0.375

Classificatore: DecisionTreeClassifier

Numero di feature selezionate: 2

Feature selezionate: ['Length2', 'Height']

Accuratezza: 0.625

Classificatore: DecisionTreeClassifier

Numero di feature selezionate: 3

Feature selezionate: ['Length1', 'Length2', 'Height']

Accuratezza: 0.6875

Classificatore: DecisionTreeClassifier

Numero di feature selezionate: 4

Feature selezionate: ['Length1', 'Length2', 'Length3', 'Height']

Accuratezza: 0.71875

Classificatore: DecisionTreeClassifier

Numero di feature selezionate: 5

Feature selezionate: ['Weight', 'Length1', 'Length2', 'Length3', 'Height']

Accuratezza: 0.6875

Migliori valori per il classificatore: DecisionTreeClassifier

Miglior n_features_to_select: 4, Migliore accuratezza: 0.71875

Feature selezionate migliori: ['Length1', 'Length2', 'Length3', 'Height']

Classificatore: RandomForestClassifier

Numero di feature selezionate: 1

Feature selezionate: ['Height']

Accuratezza: 0.4375

Classificatore: RandomForestClassifier

Numero di feature selezionate: 2

Feature selezionate: ['Length1', 'Height']

Accuratezza: 0.75

Classificatore: RandomForestClassifier

Numero di feature selezionate: 3

Feature selezionate: ['Length2', 'Length3', 'Height']

Accuratezza: 0.78125

Classificatore: RandomForestClassifier

Numero di feature selezionate: 4

Feature selezionate: ['Length1', 'Length2', 'Length3', 'Height']

Accuratezza: 0.75

Classificatore: RandomForestClassifier

Numero di feature selezionate: 5

Feature selezionate: ['Length1', 'Length2', 'Length3', 'Height', 'Width']

Accuratezza: 0.78125

Migliori valori per il classificatore: RandomForestClassifier

Miglior n_features_to_select: 3, Migliore accuratezza: 0.78125

Feature selezionate migliori: ['Length2', 'Length3', 'Height']

Classificatore: SVC

Numero di feature selezionate: 1

Feature selezionate: ['Height']

Accuratezza: 0.46875

Classificatore: SVC

Numero di feature selezionate: 2

Feature selezionate: ['Length2', 'Height']

Accuratezza: 0.8125

Classificatore: SVC

Numero di feature selezionate: 3

Feature selezionate: ['Length2', 'Length3', 'Height']

Accuratezza: 0.9375

Classificatore: SVC

Numero di feature selezionate: 4

Feature selezionate: ['Length1', 'Length2', 'Length3', 'Height']

Accuratezza: 0.9375

Classificatore: SVC

Numero di feature selezionate: 5

Feature selezionate: ['Length1', 'Length2', 'Length3', 'Height', 'Width']

Accuratezza: 0.9375

Migliori valori per il classificatore: SVC

Miglior n_features_to_select: 3, Migliore accuratezza: 0.9375

Feature selezionate migliori: ['Length2', 'Length3', 'Height']

Classificatore: ensemble_hard

Numero di feature selezionate: 1

Feature selezionate: ['Height']

Accuratezza: 0.4375

Classificatore: ensemble_hard

Numero di feature selezionate: 2

Feature selezionate: ['Length2', 'Height']

Accuratezza: 0.75

Classificatore: ensemble_hard

Numero di feature selezionate: 3

Feature selezionate: ['Length1', 'Length2', 'Height']

Accuratezza: 0.78125

Classificatore: ensemble_hard

Numero di feature selezionate: 4

Feature selezionate: ['Length1', 'Length2', 'Height', 'Width']

Accuratezza: 0.75

Classificatore: ensemble_hard

Numero di feature selezionate: 5

Feature selezionate: ['Length1', 'Length2', 'Length3', 'Height', 'Width']

Accuratezza: 0.8125

Migliori valori per il classificatore: ensemble_hard

Miglior n_features_to_select: 5, Migliore accuratezza: 0.8125

Feature selezionate migliori: ['Length1', 'Length2', 'Length3', 'Height', 'Width']

Classificatore: ensemble_soft

Numero di feature selezionate: 1

Feature selezionate: ['Length2']
Accuratezza: 0.3125

Classificatore: ensemble_soft
Numero di feature selezionate: 2
Feature selezionate: ['Length2', 'Height']
Accuratezza: 0.65625

Classificatore: ensemble_soft
Numero di feature selezionate: 3
Feature selezionate: ['Length2', 'Length3', 'Height']
Accuratezza: 0.71875

Classificatore: ensemble_soft
Numero di feature selezionate: 4
Feature selezionate: ['Length2', 'Length3', 'Height', 'Width']
Accuratezza: 0.78125

Classificatore: ensemble_soft
Numero di feature selezionate: 5
Feature selezionate: ['Length1', 'Length2', 'Length3', 'Height', 'Width']
Accuratezza: 0.75

Migliori valori per il classificatore: ensemble_soft
Miglior n_features_to_select: 4, Migliore accuratezza: 0.78125
Feature selezionate migliori: ['Length2', 'Length3', 'Height', 'Width']

Classificatore: NaiveBayesClassifier
Numero di feature selezionate: 1
Feature selezionate: ['Height']
Accuratezza: 0.46875

Classificatore: NaiveBayesClassifier
Numero di feature selezionate: 2
Feature selezionate: ['Length1', 'Height']
Accuratezza: 0.53125

Classificatore: NaiveBayesClassifier
Numero di feature selezionate: 3
Feature selezionate: ['Length1', 'Length2', 'Height']
Accuratezza: 0.59375

Classificatore: NaiveBayesClassifier
Numero di feature selezionate: 4
Feature selezionate: ['Length1', 'Length2', 'Height', 'Width']
Accuratezza: 0.625

Classificatore: NaiveBayesClassifier
Numero di feature selezionate: 5
Feature selezionate: ['Weight', 'Length1', 'Length2', 'Height', 'Width']
Accuratezza: 0.59375

Migliori valori per il classificatore: NaiveBayesClassifier
Miglior n_features_to_select: 4, Migliore accuratezza: 0.625
Feature selezionate migliori: ['Length1', 'Length2', 'Height', 'Width']

Classificatore: MLPClassifier
Numero di feature selezionate: 1
Feature selezionate: ['Height']
Accuratezza: 0.46875

Classificatore: MLPClassifier
Numero di feature selezionate: 2
Feature selezionate: ['Length2', 'Height']
Accuratezza: 0.875

Classificatore: MLPClassifier
Numero di feature selezionate: 3
Feature selezionate: ['Length2', 'Height', 'Width']
Accuratezza: 0.875

Classificatore: MLPClassifier
Numero di feature selezionate: 4
Feature selezionate: ['Length2', 'Length3', 'Height', 'Width']
Accuratezza: 0.9687

Classificatore: MLPClassifier
Numero di feature selezionate: 5
Feature selezionate: ['Length1', 'Length2', 'Length3', 'Height', 'Width']
Accuratezza: 0.96875

Migliori valori per il classificatore: MLPClassifier
Miglior n_features_to_select: 4, Migliore accuratezza: 0.96875
Feature selezionate migliori: ['Length2', 'Length3', 'Height', 'Width']

Ricapitolando

Classificatore: DecisionTreeClassifier
Miglior n_features_to_select: 4, Migliore accuratezza: 0.71875
Feature selezionate migliori: ['Length1', 'Length2', 'Length3', 'Height']

Performance peggiorate (0.7187 < 0.78)

Classificatore: RandomForestClassifier
Miglior n_features_to_select: 3, Migliore accuratezza: 0.78125
Feature selezionate migliori: ['Length2', 'Length3', 'Height']

Performance peggiorate (0.7187 < 0.8125)

Classificatore: SVC
Miglior n_features_to_select: 3, Migliore accuratezza: 0.9375

Feature selezionate migliori: ['Length2', 'Length3', 'Height']

Performance peggiorate (0.9375 < 0.968)

Classificatore: ensemble_hard

Miglior n_features_to_select: 5, Migliore accuratezza: 0.8125

Feature selezionate migliori: ['Length1', 'Length2', 'Length3', 'Height', 'Width']

Performance UGUALI (0.8125 == 0.8125)

Classificatore: ensemble_soft

Miglior n_features_to_select: 4, Migliore accuratezza: 0.78125

Feature selezionate migliori: ['Length2', 'Length3', 'Height', 'Width']

Performance UGUALI (0.78125 == 0.78125)

Classificatore: NaiveBayesClassifier

Miglior n_features_to_select: 4, Migliore accuratezza: 0.625

Feature selezionate migliori: ['Length1', 'Length2', 'Height', 'Width']

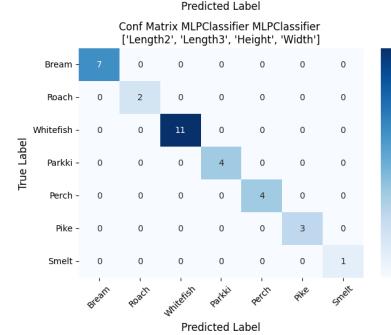
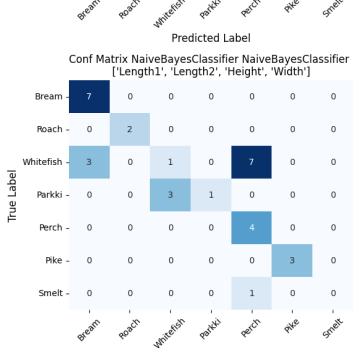
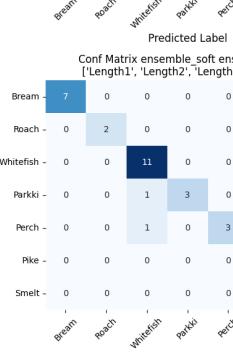
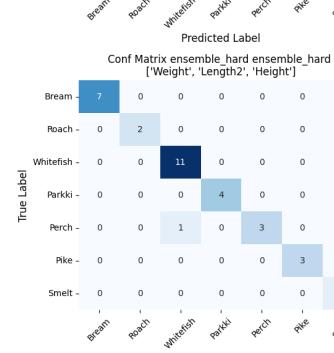
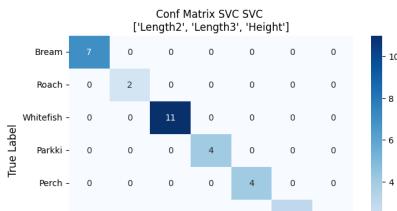
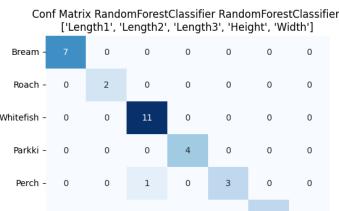
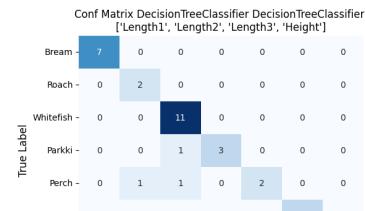
Performance UGUALI (0.625 == 0.625)

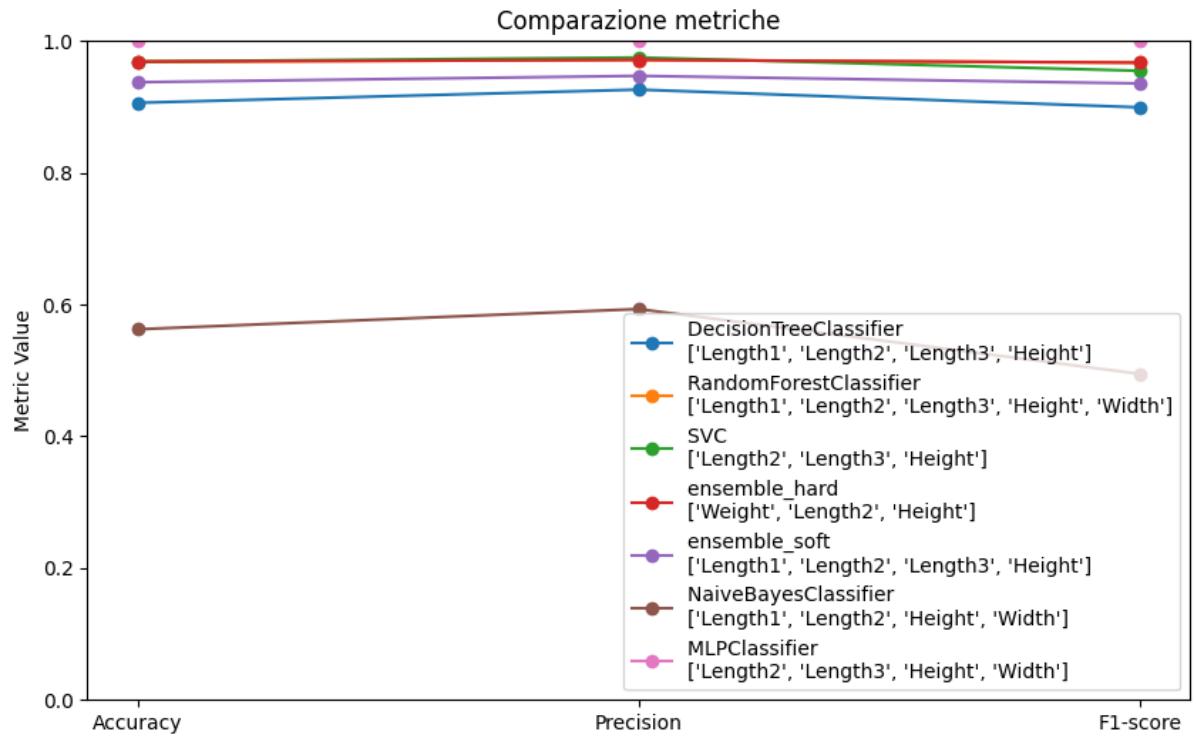
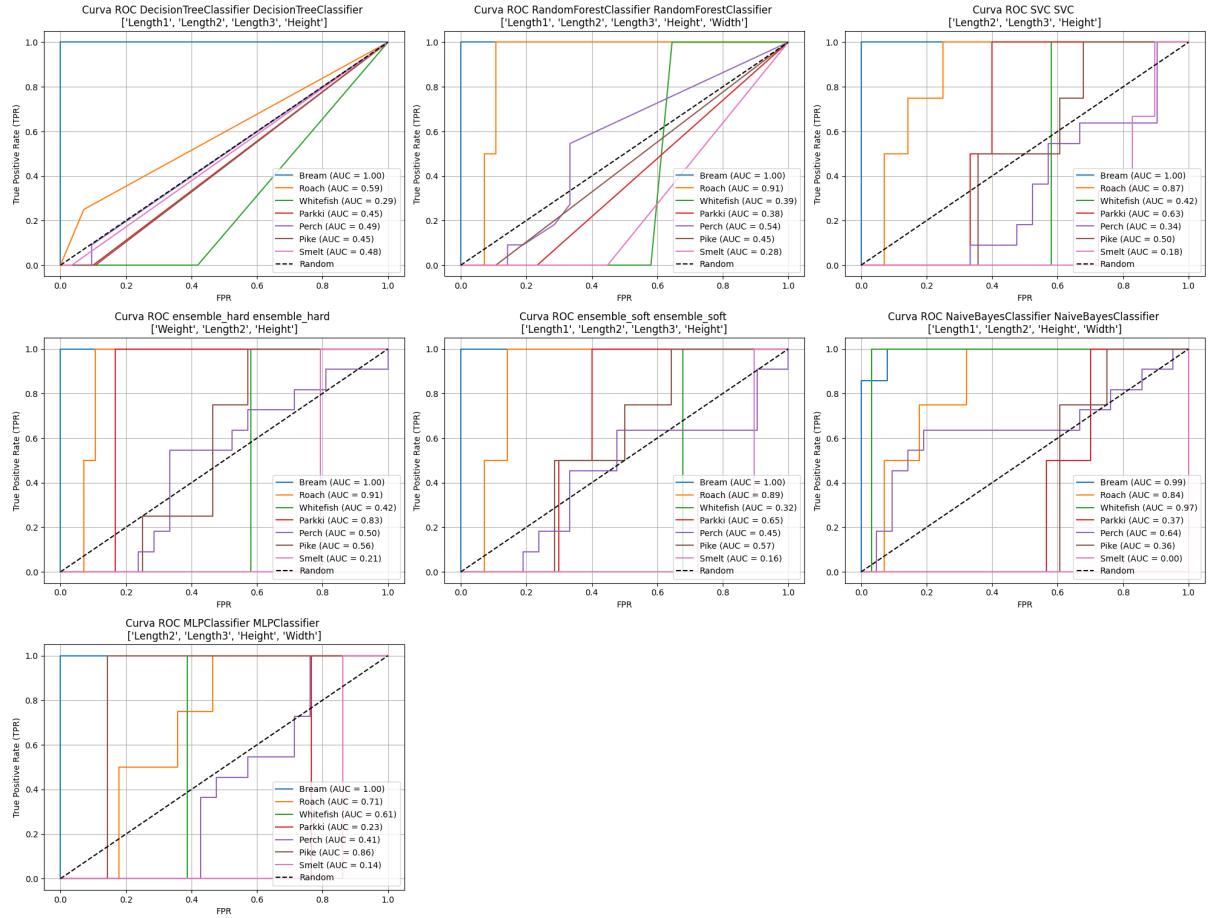
Classificatore: MLPClassifier

Miglior n_features_to_select: 4, Migliore accuratezza: 0.96875

Feature selezionate migliori: ['Length2', 'Length3', 'Height', 'Width']

Performance UGUALI (0.96875 == 0.96875)





In alcuni casi notiamo un miglioramento sulle predizioni dopo aver fatto la selezione delle feature. Questo accade perché vengono scartate eventuali feature ridondanti o poco significative e per questo il

modello può diventare più efficiente. Il numero ottimale di selezione delle feature risulta essere 4. Ora alcuni modelli che prima tendevano a confondere il whitefish con altri ora hanno solamente un errore.

Conclusioni Finali

Il dataset utilizzato conteneva molti pochi record, con numero di classi abbastanza sbilanciato.

I modelli migliori si sono rilevati SVC e MLP, MLP avrebbe bisogno di un analisi dell'andamento del training per capire meglio l'effettiva bontà dei suoi risultati, tutti senza strattificazione.

Progetto svolto da:

Francesco Marotto, Lilliu Stefano, Pili Francesco