UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO | DIPARTIMENTO DI INFORMATICA

Corso di Laurea in Informatica

Ingegneria della Conoscenza

Anno Accademico 2022-2023

CLASSIFICAZIONE E CLUSTERING DI TRACCE AUDIO

Progetto di **Festa Donato** (758339) d.festa5@studenti.uniba.it

Link GitHub: progetto

INDICE

1 Introduzione

L'obiettivo di questo progetto è migliorare la classificazione dei generi musicali basata su caratteristiche audio utilizzando algoritmi di machine learning. Il sistema sviluppato applicherà diverse tecniche per ottimizzare la predizione dei generi musicali di nuove canzoni, contribuendo così a una migliore organizzazione e classificazione delle stesse. Dopo aver valutato le performance iniziali si vanno a rieseguire gli esperimenti con le prime 5 feature per importanza per vedere come e cosa varia.

Il progetto è stato realizzato in Python utilizzando l'IDE Visual Studio Code, una scelta motivata dalla vasta disponibilità di librerie che facilitano la manipolazione e l'analisi dei dati in modo efficiente.

Versione Python: 3.12.2

Librerie utilizzate:

- **matplotlib:** Utilizzata per la visualizzazione dei grafici, incluse le curve di apprendimento e le rappresentazioni grafiche delle feature più importanti.
- **numpy:** Libreria essenziale per la gestione di array e operazioni matematiche complesse.
- **pandas:** Utilizzata per l'importazione e la manipolazione dei dataset in formato CSV.
- **scikit-learn:** Fornisce strumenti per il machine learning, inclusi modelli di classificazione come KNeighborsClassifier e RandomForestClassifier, e la ricerca degli iperparametri con GridSearchCV.
- **seaborn:** Utilizzata per la visualizzazione delle matrici di confusione.

Per avviare il progetto, aprirlo con un IDE ed eseguire nel terminale il comando 'pip install -r requirements.txt' in modo da installare tutte le librerie necessarie. Successivamente avviare il programma dal file main.py per la classificazione oppure dal file clustering.py per il clustering.

2 Preparazione dei Dati

2.1 Caricamento e struttura del dataset

Il dataset utilizzato per questo progetto è il Music Features Dataset (https://www.kaggle.com/datasets/insiyeah/musicfeatures), che contiene 1000 esempi di file audio. Il dataset è suddiviso in 800 esempi per il training e 200 per il test. Le caratteristiche (features) estratte da ogni brano, utilizzate per la classificazione, sono le seguenti:

- **Tempo (0)**: Misura della velocità della musica.
- **Beats (1)**: Unità ritmica della musica.
- **Chroma_stft (2)**: Trasformata di Fourier a breve termine.
- **RMSE (3)**: Root Mean Square Error.
- **Spectral_centroid (4)**: Indica il "centro di massa" dello spettro.

- **Spectral_bandwidth (5)**: Misura dell'ampiezza dello spettro, che non può essere inferiore alla metà del suo massimo.
- **Roll-off (6)**: Attenuazione delle frequenze gravi o acute man mano che ci si allontana dalla gamma centrale.
- **Zero_crossing_rate (7)**: Frequenza con cui il segnale cambia da positivo a negativo, o viceversa.
- **MFCCs (8-27)**: Coefficienti Mel-frequency cepstral (MFCCs), 20 coefficienti utilizzati per identificare l'MFC.
- **Label (Target Feature)**: Nome del genere associato al file audio.

Queste features sono fondamentali per rappresentare una traccia audio. Durante la fase di pre-processing del dataset, si è deciso di mantenere tutte le features per garantire una classificazione accurata dei generi musicali.

2.2 Processing dei dati audio

Il processing del dataset è una fase cruciale per garantire che i dati siano pronti per l'addestramento e la valutazione dei modelli di machine learning. Le principali operazioni di processing del dataset includono la separazione delle feature dalle etichette, la suddivisione del dataset in training e test set, e la normalizzazione delle feature. Il processo seguito per il dataset di questo progetto include i seguenti passaggi chiave:

1. **Caricamento del Dataset**: Utilizzando la libreria pandas, il dataset viene caricato in un DataFrame per una facile manipolazione e analisi. La colonna 'filename', che non contiene informazioni utili per il training del modello, viene rimossa in questa fase.
2. **Separazione delle Features e delle Etichette**: Le feature e le etichette (colonna 'label') vengono separate. Le feature costituiscono i dati di input per i modelli, mentre le etichette sono i target che i modelli devono imparare a predire.
3. **Suddivisione del Dataset**: Il dataset viene suddiviso in set di training e test utilizzando train_test_split di scikit-learn. Questa suddivisione è importante per valutare le performance del modello su dati non visti durante l'addestramento.
4. **Normalizzazione delle Features**: Le feature vengono normalizzate per garantire che tutte abbiano lo stesso range di valori. Questo step è particolarmente importante per algoritmi come KNN, che si basano sulle distanze tra i punti dati.
5. **Ottimizzazione degli Iperparametri**: Per migliorare le prestazioni dei modelli, vengono utilizzate tecniche di Grid Search per trovare i migliori iperparametri. Questo processo viene applicato sia per KNeighborsClassifier che per RandomForestClassifier.
6. **Selezione delle Feature**: Viene utilizzato il Random Forest Classifier per identificare le feature più importanti. Le prime 5 feature più importanti vengono selezionate per una

seconda fase di sperimentazione per valutare l'impatto della riduzione delle feature sulle performance del modello.

Questi passaggi assicurano che i dati siano preprocessati correttamente per l'addestramento e la valutazione dei modelli di machine learning, contribuendo a migliorare la classificazione dei generi musicali.

3 Clustering dei Dati Audio

3.1 Metodo di Clustering Utilizzato e Descrizione

Nel contesto del progetto, è stato utilizzato l'algoritmo di clustering K-means per analizzare e raggruppare i dati audio relativi ai generi musicali. Questo metodo è stato scelto per la sua capacità di gestire grandi volumi di dati senza supervisione e per la sua efficienza nel trovare pattern nei dati.

Prima dell'applicazione di K-means, le feature audio estratte da ciascuna traccia sono state normalizzate per assicurare che avessero lo stesso peso nel calcolo delle distanze euclidee utilizzate dall'algoritmo. Il dataset è stato quindi suddiviso in cluster, con un numero prestabilito di k=5, sulla base di un'analisi preliminare dei dati musicali.

Durante l'esecuzione di K-means, l'algoritmo è stato iterato fino alla convergenza, ricalcolando i centroidi dei cluster sulla base delle assegnazioni dei punti dati ad ogni iterazione. Questo approccio ha permesso di identificare raggruppamenti significativi all'interno dei generi musicali, contribuendo alla comprensione delle relazioni e delle caratteristiche distintive tra di essi.

3.2 Analisi e Interpretazione dei Cluster Ottenuti

L'analisi dei cluster ottenuti dall'algoritmo K-means rappresenta una fase cruciale per comprendere le relazioni e le caratteristiche distintive dei generi musicali nel dataset. Dopo aver applicato K-means con k=5, è stato eseguito un processo di interpretazione dei risultati per valutare la coerenza e la significatività dei raggruppamenti formati.

Durante l'esecuzione dell'algoritmo, sono state registrate varie iterazioni per ottimizzare la posizione dei centroidi dei cluster. L'inertia, che rappresenta la somma delle distanze quadrate tra ciascun punto dati e il centroide del cluster corrispondente, ha mostrato una tendenza decrescente nelle iterazioni successive. L'algoritmo ha raggiunto la convergenza dopo 11 iterazioni, come indicato dall'inertia che ha smesso di diminuire significativamente.

Questo processo ha contribuito a definire cluster di tracce audio basati su similitudini nelle feature estratte, tra cui tempo, beats, chroma_stft, e altre caratteristiche audio descritte in precedenza. Ogni cluster rappresenta un gruppo di tracce con caratteristiche audio simili, suggerendo una suddivisione significativa dei generi musicali in categorie distinte. Per valutare la qualità dei cluster ottenuti, è stato calcolato il Purity Score, che misura quanto i cluster corrispondano

effettivamente alle etichette di genere fornite nel dataset. Il Purity Score ottenuto è stato del 35.70%, indicando una buona coerenza dei raggruppamenti rispetto alle etichette originali.

```
Initialization complete
Iteration 0, inertia 446.44727757119426.
Iteration 1, inertia 320.8284013196756.
Iteration 2, inertia 313.9420753288875.
Iteration 3, inertia 311.32018277577265.
Iteration 4, inertia 310.1440823122196.
Iteration 5, inertia 309.63943648328274.
Iteration 6, inertia 309.49197354512944.
Iteration 7, inertia 309.33639369536274.
Iteration 8, inertia 309.238213253397.
Iteration 9, inertia 309.0916076037716.
Iteration 10, inertia 308.9801227323272.
Iteration 11, inertia 308.9468316979233.
Converged at iteration 11: strict convergence.
Purity Score: 0.3570
```

4 Classificazione Supervisionata

4.1 Modelli di Machine Learning Utilizzati

In questo progetto sono stati implementati e valutati diversi modelli di apprendimento supervisionato per la classificazione dei generi musicali. I modelli selezionati includono K-Nearest Neighbors (KNN), Random Forest Classifier (RFC) e Naive Bayes (NB), ciascuno dei quali offre vantaggi specifici per il tipo di dati e il problema in esame.

K-Nearest Neighbors (KNN) Il modello KNN classifica i dati in base alla vicinanza a esempi nel dataset di addestramento. Per ogni traccia audio da classificare, KNN identifica le K tracce più vicine nel set di addestramento e assegna la classe più comune tra queste. Questo modello è particolarmente utile per dataset con strutture di dati complesse dove la similarità è un buon indicatore di classe.

Naive Bayes Il modello Naive Bayes si basa sul teorema di Bayes con l'assunzione di indipendenza tra le feature. Nonostante questa assunzione sia raramente vera per dati reali, Naive Bayes è noto per essere sorprendentemente efficace per molti problemi di classificazione. Questo modello è particolarmente adatto per dataset di grandi dimensioni e con molte feature.

Random Forest Classifier Random Forest è un ensemble di alberi decisionali, dove ogni albero è costruito utilizzando un sottocampione casuale del dataset e un sottoinsieme casuale delle feature. La predizione finale è ottenuta mediante la media delle predizioni di tutti gli alberi nel forest. Questo metodo aiuta a ridurre l'overfitting e migliora la generalizzazione del modello. La selezione degli iperparametri ottimali per Random Forest è stata eseguita utilizzando Grid Search per garantire le migliori prestazioni possibili.

4.2 Ottimizzazione degli Iperparametri

L'ottimizzazione degli iperparametri è un passaggio cruciale nel processo di machine learning per migliorare le prestazioni dei modelli. In questo progetto, abbiamo utilizzato Grid Search per trovare i migliori iperparametri per ciascun modello di machine learning utilizzato (K-Nearest Neighbors, Naive Bayes e Random Forest Classifier).

Grid Search Grid Search è una tecnica di ricerca esaustiva per l'ottimizzazione degli iperparametri utilizzata per KNN e Random Forest, in cui si esplora un insieme predefinito di iperparametri per trovare la combinazione che produce i migliori risultati. La procedura segue questi passaggi:

1. **Definizione del parametro grid:** Per ciascun modello, è stato definito un insieme di iperparametri da esplorare. Ad esempio, per KNN, il numero di vicini (k) e il tipo di distanza (metrica) sono stati variati. Per Random Forest, il numero di alberi (n_estimators) e la profondità massima degli alberi (max_depth) sono stati considerati.
2. **Esecuzione di Grid Search:** Utilizzando la funzione GridSearchCV di scikit-learn, il modello è stato addestrato e validato per ciascuna combinazione di iperparametri nel grid. Questo processo ha utilizzato la cross-validation per garantire una valutazione accurata delle prestazioni.
3. **Selezione dei migliori iperparametri:** La combinazione di iperparametri che ha prodotto le migliori prestazioni di cross-validation è stata selezionata come configurazione ottimale per il modello.

Per il modello Naive Bayes, non è stata necessaria una ricerca di iperparametri complessa come per KNN e Random Forest, poiché Naive Bayes ha meno iperparametri e spesso funziona bene con i valori di default.

4.3 Valutazione delle Performance dei Modelli

La valutazione delle performance dei modelli è fondamentale per comprendere l'efficacia delle soluzioni di machine learning implementate. In questo progetto, sono state utilizzate diverse metriche di valutazione per fornire una visione completa delle prestazioni dei modelli.

Metriche di Valutazione

1. **Accuracy:** La proporzione di predizioni corrette rispetto al totale delle predizioni effettuate. È una metrica intuitiva e di facile interpretazione, ma potrebbe non essere sufficiente per dataset sbilanciati.
2. **Precision:** La proporzione di predizioni positive corrette rispetto al totale delle predizioni positive.
3. **Recall:** La proporzione di effettivi positivi correttamente identificati.
4. **F1-Score:** La media armonica tra precision e recall, fornendo un equilibrio tra le due metriche.

L'accuratezza generale, insieme a precision, recall e F1-score, permette di comprendere quanto bene i modelli sono in grado di classificare correttamente i generi musicali.

Confusion Matrix

La confusion matrix è uno strumento fondamentale per la valutazione delle performance di un modello di classificazione. Essa rappresenta in modo tabellare il numero di predizioni corrette e incorrette del modello, suddivise per classe. La matrice di confusione è strutturata in modo che ogni riga rappresenti le istanze nella classe reale e ogni colonna rappresenti le istanze nella classe predetta.

In particolare, la confusion matrix permette di:

- **Identificare le classi ben classificate**: Il numero di predizioni corrette per ogni classe si trova sulla diagonale principale della matrice.
- **Rilevare le classi confuse**: Gli elementi fuori dalla diagonale principale mostrano quali classi vengono erroneamente predette come altre classi.
- **Analizzare il pattern di errore**: Fornendo un quadro dettagliato degli errori di classificazione, la matrice permette di capire se ci sono particolari classi che vengono frequentemente confuse tra loro, indicando potenziali problemi nei dati o nel modello.

La confusion matrix, insieme ad altre metriche come precision, recall e F1-score, offre una visione completa delle capacità del modello di classificare correttamente i dati e di identificare le aree che necessitano di miglioramento.

Curve di Apprendimento

Le curve di apprendimento sono uno strumento utile per visualizzare come la performance di un modello di machine learning evolve con il numero di campioni di addestramento. In generale, una curva di apprendimento mostra:

- **Training Score**: La performance del modello sul set di addestramento.
- **Validation Score**: La performance del modello sul set di validazione.

Queste curve aiutano a diagnosticare problemi come overfitting (dove il modello si adatta troppo bene ai dati di addestramento ma non generalizza ai dati di test) e underfitting (dove il modello non cattura la complessità del problema).

5 Valutazione delle prestazioni

5.1 Metriche di Valutazione e Matrici di Confusione

In questa sezione, verranno presentate le metriche di valutazione per i modelli di machine learning utilizzati nel progetto: **K-Nearest Neighbors** (KNN), **Naive Bayes** (NB), e **Random Forest Classifier** (RFC). Le metriche considerate includono l'**accuratezza** (Accuracy), la **precisione** (Precision), il

recall, e il **punteggio F1** (F1 Score). Inoltre, saranno analizzate le **confusion matrix** per ciascun modello, le quali permettono di individuare quali classi vengono maggiormente confuse tra loro.

K-Nearest Neighbors (KNN):

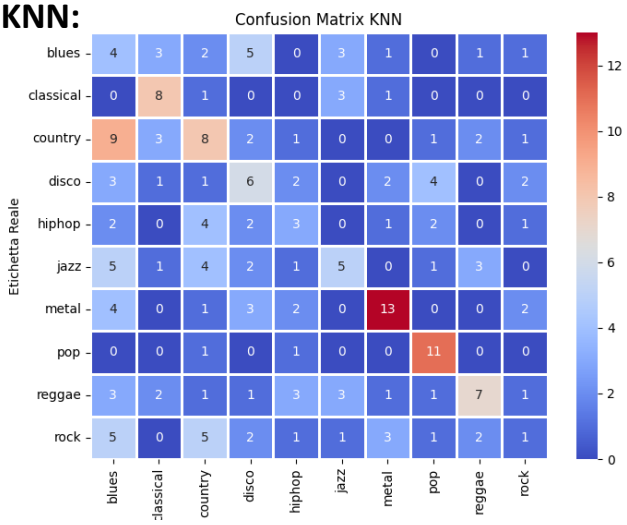
Per il modello KNN sono stati utilizzati i seguenti iperparametri:

- '*algorithm*': auto
- '*leaf_size*': 5
- '*n_neighbors*': 4

Le metriche di valutazione risultanti sono:

- **Accuracy**: 0.3300
- **Precision**: 0.3345
- **Recall**: 0.3543
- **F1 Score**: 0.3338

Confusion Matrix KNN:



La matrice di confusione ha le classi reali sulle righe (Etichetta Reale) e le classi predette sulle colonne (Predicted Labels). Ogni cella (i, j) indica il numero di campioni della classe i classificati come classe j.

Analisi delle Classi:

Blues:

- Correttamente classificati: 4
- Totale esempi: 22 (4 corretti + 18 errati)
- Esempi classificati erroneamente come: Country (9), Disco (3), HipHop (1), Jazz (3), Metal (1), Rock (1)

Classical:

- Correttamente classificati: 8
- Totale esempi: 13 (8 corretti + 5 errati)
- Esempi classificati erroneamente come: Blues (2), Country (1), Disco (1), Pop (1)

Country:

- Correttamente classificati: 8
- Totale esempi: 19 (8 corretti + 11 errati)
- Esempi classificati erroneamente come: Blues (3), Classical (1), Disco (6), HipHop (2), Jazz (1), Rock (1)

Disco:

- Correttamente classificati: 6
- Totale esempi: 19 (6 corretti + 13 errati)
- Esempi classificati erroneamente come: Blues (3), Country (1), HipHop (2), Jazz (2), Metal (2), Reggae (1), Rock (2)

HipHop:

- Correttamente classificati: 4
- Totale esempi: 19 (4 corretti + 15 errati)
- Esempi classificati erroneamente come: Blues (2), Country (2), Disco (4), Jazz (3), Metal (2), Pop (2)

Jazz:

- Correttamente classificati: 5
- Totale esempi: 20 (5 corretti + 15 errati)
- Esempi classificati erroneamente come: Blues (2), Classical (1), Country (1), Disco (4), HipHop (2), Metal (2), Reggae (3)

Metal:

- Correttamente classificati: 13
- Totale esempi: 20 (13 corretti + 7 errati)
- Esempi classificati erroneamente come: Blues (1), Classical (2), Country (4)

Pop:

- Correttamente classificati: 11
- Totale esempi: 17 (11 corretti + 6 errati)
- Esempi classificati erroneamente come: Classical (1), Country (1), Disco (1), HipHop (1), Rock (2)

Reggae:

- Correttamente classificati: 7

- Totale esempi: 13 (7 corretti + 6 errati)
- Esempi classificati erroneamente come: Blues (3), Disco (1), Jazz (1), Rock (1)

Rock:

- Correttamente classificati: 2
- Totale esempi: 20 (2 corretti + 18 errati)
- Esempi classificati erroneamente come: Blues (5), Country (5), Disco (1), HipHop (1), Jazz (3), Metal (2), Pop (1)

Conclusioni:

La matrice di confusione del modello KNN mostra che il modello ha difficoltà significative nel classificare correttamente le diverse classi. Le prestazioni del modello sono relativamente basse in termini di accuracy, precision, recall, e F1 score. Le classi con il maggior numero di esempi correttamente classificati sono **Metal** e **Pop**, ma molte altre classi, come **Rock** e **Blues**, mostrano un alto tasso di errori.

Le metriche complessive indicano una performance debole del modello, suggerendo la necessità di miglioramenti. Possibili azioni per migliorare le performance potrebbero includere:

- L'ottimizzazione degli iperparametri,
- L'uso di tecniche di feature selection,
- L'esplorazione di modelli alternativi,
- L'incremento del numero di vicini (n_neighbors).

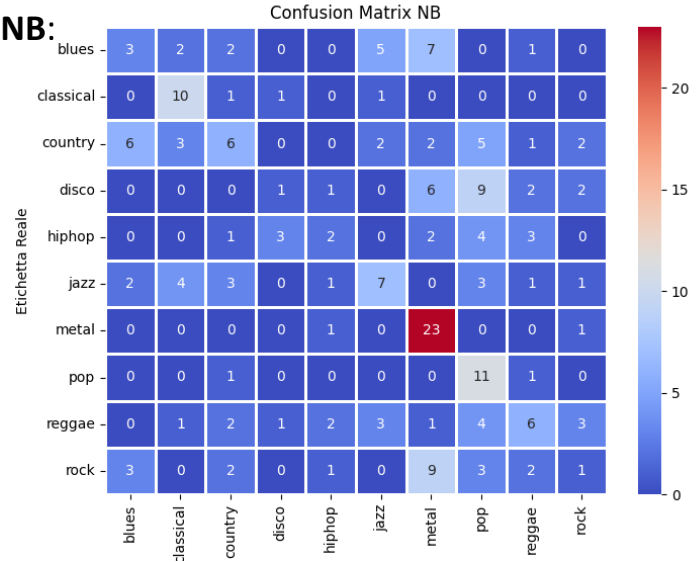
In generale, il KNN con i parametri attuali non riesce a distinguere chiaramente tra le diverse classi, il che si riflette nelle basse metriche di valutazione complessive.

Naive Bayes (NB):

Le metriche di valutazione per il modello Naive Bayes sono:

- **Accuracy**: 0.3500
- **Precision**: 0.3048
- **Recall**: 0.3715
- **F1 Score**: 0.3048

Confusion Matrix NB:



La matrice di confusione ha le classi reali sulle righe (Etichetta Reale) e le classi predette sulle colonne (Predicted Labels). Ogni cella (i, j) indica il numero di campioni della classe i classificati come classe j.

Analisi delle Classi:

Blues:

- Correttamente classificati: 3
- Totale esempi: 20 (3 corretti + 17 errati)
- Esempi classificati erroneamente come: Classical (2), Country (6), Disco (2), Jazz (2), Metal (5), Rock (2)

Classical:

- Correttamente classificati: 10
- Totale esempi: 14 (10 corretti + 4 errati)
- Esempi classificati erroneamente come: Country (1), Jazz (1), Metal (1), Reggae (1)

Country:

- Correttamente classificati: 3
- Totale esempi: 20 (3 corretti + 17 errati)
- Esempi classificati erroneamente come: Blues (6), Classical (1), Disco (1), HipHop (2), Jazz (2), Metal (2), Reggae (2), Rock (3)

Disco:

- Correttamente classificati: 1
- Totale esempi: 19 (1 corretto + 18 errati)
- Esempi classificati erroneamente come: Country (1), HipHop (3), Jazz (1), Metal (6), Pop (1), Reggae (1), Rock (6)

HipHop:

- Correttamente classificati: 3
- Totale esempi: 17 (3 corretti + 14 errati)
- Esempi classificati erroneamente come: Blues (1), Country (1), Disco (1), Jazz (3), Metal (4), Pop (2), Reggae (1), Rock (1)

Jazz:

- Correttamente classificati: 7
- Totale esempi: 20 (7 corretti + 13 errati)
- Esempi classificati erroneamente come: Blues (2), Country (4), Disco (1), HipHop (3), Metal (3), Rock (3)

Metal:

- Correttamente classificati: 23
- Totale esempi: 28 (23 corretti + 5 errati)

- Esempi classificati erroneamente come: Country (2), Disco (1), Jazz (1), Pop (1)

Pop:

- Correttamente classificati: 11
- Totale esempi: 12 (11 corretti + 1 errato)
- Esempi classificati erroneamente come: Metal (1)

Reggae:

- Correttamente classificati: 6
- Totale esempi: 13 (6 corretti + 7 errati)
- Esempi classificati erroneamente come: Blues (1), Disco (1), Jazz (1), Metal (4)

Rock:

- Correttamente classificati: 2
- Totale esempi: 20 (2 corretti + 18 errati)
- Esempi classificati erroneamente come: Blues (3), Country (2), Disco (1), Jazz (1), Metal (9), Pop (1), Reggae (3)

Conclusioni:

La matrice di confusione del modello Naive Bayes mostra che il modello ha prestazioni leggermente migliori rispetto al KNN ma continua a incontrare difficoltà significative nel classificare correttamente le diverse classi. Le prestazioni del modello, come evidenziato dalle metriche di valutazione, sono ancora relativamente basse.

- **Blues** e **Rock**: Mostrano alti tassi di errori, con una grande parte di esempi classificati erroneamente come altre classi.
- **Metal** e **Pop**: Hanno una performance migliore, con una buona parte di esempi correttamente classificati.
- **Country**, **Disco**, e **HipHop**: Hanno alti tassi di errori, suggerendo che queste classi sono spesso confuse con altre.

Le metriche complessive indicano una performance mediocre del modello Naive Bayes, suggerendo la necessità di miglioramenti. Possibili azioni per migliorare le performance potrebbero includere:

- Ottimizzazione degli iperparametri,
- Uso di tecniche di feature selection,
- Esplorazione di modelli alternativi,
- Preprocessing dei dati, come la normalizzazione o la trasformazione delle feature.

In generale, il modello Naive Bayes con i parametri attuali mostra una moderata capacità di classificazione, ma le basse metriche di valutazione indicano che c'è ancora molto spazio per migliorare le performance del modello.

Random Forest Classifier (RFC):

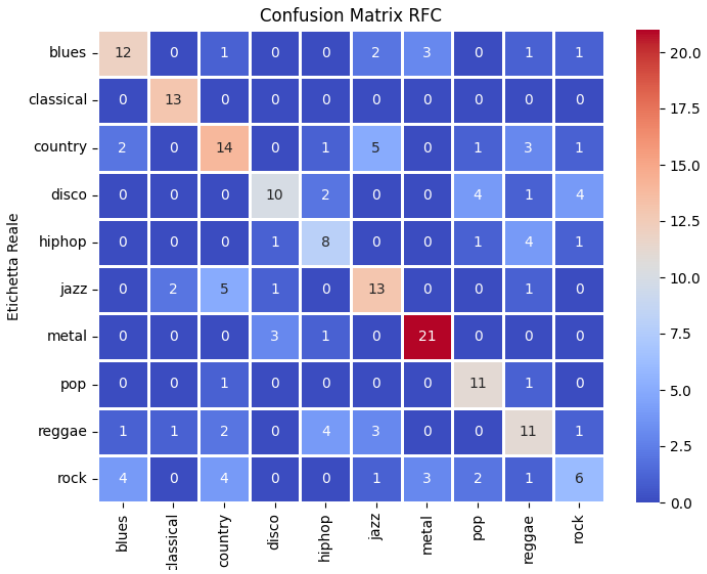
Per il modello RFC sono stati utilizzati i seguenti iperparametri:

- '*bootstrap*': True
- '*max_depth*': 25
- '*max_features*': 20
- '*min_samples_leaf*': 1
- '*min_samples_split*': 2
- '*n_estimators*': 250

Le metriche di valutazione risultanti sono:

- **Accuracy**: 0.6050
- **Precision**: 0.5971
- **Recall**: 0.6260
- **F1 Score**: 0.6044

Confusion Matrix RFC:



La matrice di confusione ha le classi reali sulle righe (Etichetta Reale) e le classi predette sulle colonne (Predicted Labels). Ogni cella (i, j) indica il numero di campioni della classe i classificati come classe j.

Analisi delle classi:

Blues:

- Correttamente classificati: 12
- Totale esempi: 20 (12 corretti + 8 errati)
- Esempi classificati erroneamente come: Country (2), Disco (3), Pop (1), Reggae (1), Rock (1)

Classical:

- Correttamente classificati: 13

- Totale esempi: 13 (13 corretti + 0 errati)
- Nessun esempio classificato erroneamente

Country:

- Correttamente classificati: 14
- Totale esempi: 20 (14 corretti + 6 errati)
- Esempi classificati erroneamente come: Blues (2), Disco (1), Jazz (1), Pop (1), Reggae (1)

Disco:

- Correttamente classificati: 10
- Totale esempi: 20 (10 corretti + 10 errati)
- Esempi classificati erroneamente come: Blues (1), Country (2), HipHop (4), Pop (1), Reggae (1), Rock (1)

HipHop:

- Correttamente classificati: 8
- Totale esempi: 15 (8 corretti + 7 errati)
- Esempi classificati erroneamente come: Country (1), Disco (1), Jazz (4), Metal (1)

Jazz:

- Correttamente classificati: 13
- Totale esempi: 21 (13 corretti + 8 errati)
- Esempi classificati erroneamente come: Country (2), HipHop (1), Metal (1), Reggae (3), Rock (1)

Metal:

- Correttamente classificati: 21
- Totale esempi: 25 (21 corretti + 4 errati)
- Esempi classificati erroneamente come: Country (3), Pop (1)

Pop:

- Correttamente classificati: 11
- Totale esempi: 12 (11 corretti + 1 errato)
- Esempi classificati erroneamente come: Disco (1)

Reggae:

- Correttamente classificati: 11
- Totale esempi: 15 (11 corretti + 4 errati)
- Esempi classificati erroneamente come: Blues (1), Disco (1), Jazz (3)

Rock:

- Correttamente classificati: 6
- Totale esempi: 20 (6 corretti + 14 errati)
- Esempi classificati erroneamente come: Blues (4), Country (1), Disco (1), HipHop (1), Jazz (2), Metal (2), Pop (1), Reggae (1), Rock (2)

Conclusioni:

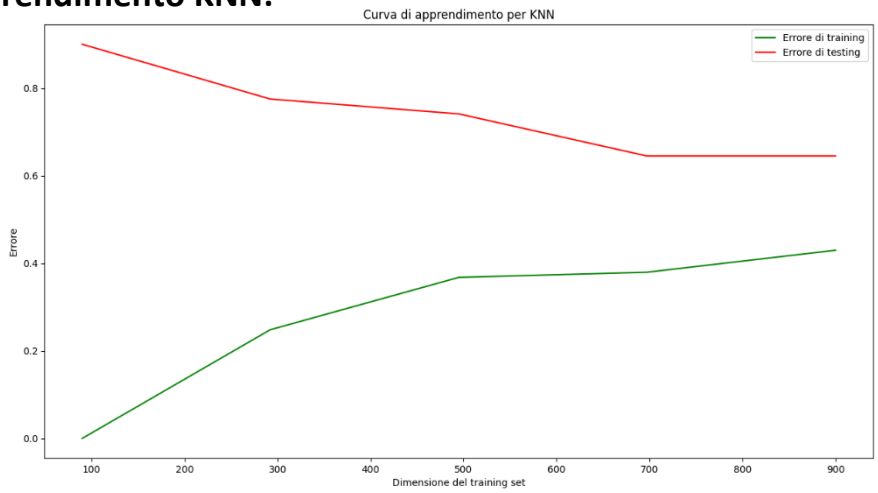
La matrice di confusione evidenzia che il modello ha una buona performance per alcune classi, come **Classical** (perfettamente classificata) e **Metal** (21 su 25 esempi correttamente classificati). Tuttavia, mostra anche difficoltà nel distinguere altre classi, come **Rock**, che ha una alta percentuale di errori (solo 6 su 20 esempi correttamente classificati).

Le metriche di valutazione complessive (accuracy, precision, recall, F1 score) indicano una performance moderata del modello, suggerendo che ci sono margini di miglioramento. Possibili azioni per migliorare le performance potrebbero includere l'ottimizzazione degli iperparametri, l'uso di tecniche di feature selection, o l'esplorazione di modelli alternativi.

5.2 Visualizzazione delle Curve di Apprendimento

Le curve di apprendimento forniscono un'analisi visiva di come le prestazioni dei modelli migliorano con l'aumentare del numero di esempi di addestramento. Di seguito sono riportate le curve di apprendimento per ciascun modello.

Curva di Apprendimento KNN:



La curva di apprendimento mostra l'andamento degli errori di training e di testing in funzione della dimensione del training set. In questa rappresentazione:

- **Asse x:** Dimensione del training set
- **Asse y:** Errore
- **Linea verde:** Errore di training
- **Linea rossa:** Errore di testing

Osservazioni:

Errore di Training:

- La linea verde aumenta con l'aumentare della dimensione del training set, indicando che l'errore di training diventa leggermente più elevato man mano che il modello ha più dati da apprendere.
- La deviazione standard dell'errore di training è circa 0.0104, suggerendo una variazione contenuta nella performance di training.
- La varianza dell'errore di training è circa 0.0001, confermando la stabilità nel training.

Errore di Testing:

- La linea rossa mostra che l'errore di testing diminuisce inizialmente con l'aumentare della dimensione del training set, ma tende poi a stabilizzarsi. Questo comportamento è comune poiché, dopo un certo punto, aggiungere più dati di training potrebbe non portare a significativi miglioramenti nelle performance.
- La deviazione standard dell'errore di testing è circa 0.0524, indicando una variazione più significativa rispetto all'errore di training, ma ancora moderata.
- La varianza dell'errore di testing è circa 0.0027, indicando una dispersione moderata nei valori dell'errore di testing.

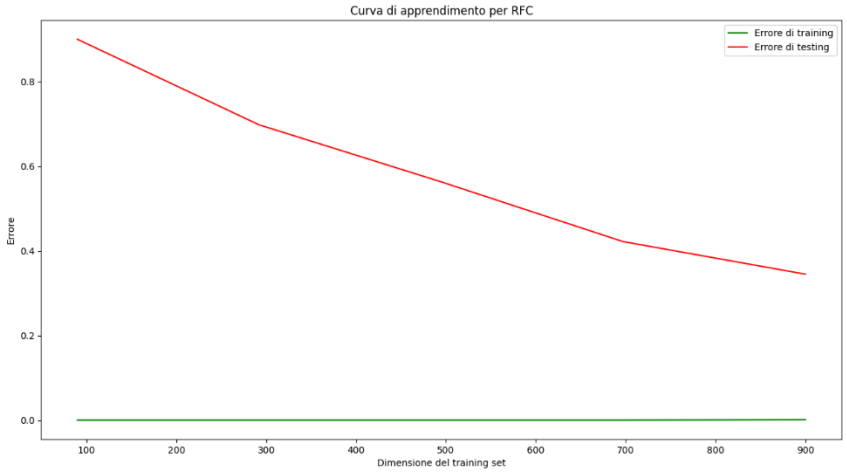
Interpretazione:

- **Overfitting:**
    - Il modello KNN non mostra segni evidenti di overfitting, poiché l'errore di training non è vicino a 0 e l'errore di testing, sebbene maggiore, è comunque contenuto e tende a stabilizzarsi.
- **Generalizzazione:**
    - L'errore di testing inizialmente diminuisce con l'aumentare della dimensione del training set, suggerendo che il modello sta migliorando la sua capacità di generalizzazione.
    - Con un ulteriore aumento dei dati di training, l'errore di testing potrebbe stabilizzarsi ulteriormente, ma non è previsto un miglioramento drastico.
- **Deviazione Standard e Varianza:**
    - Le basse deviazioni standard e varianze per l'errore di training indicano che il modello ha una performance consistente durante il training.
    - Le moderate deviazioni standard e varianze per l'errore di testing suggeriscono che il modello è abbastanza stabile, ma c'è ancora una certa variazione nei risultati di testing che potrebbe essere mitigata con ulteriori dati o tecniche di ottimizzazione.

Conclusioni:

Il modello K-Nearest Neighbors attualmente mostra una buona capacità di generalizzazione, senza segni evidenti di overfitting. Per migliorare ulteriormente le performance del modello, si potrebbe considerare l'uso di tecniche di ottimizzazione dei parametri o l'aumento della dimensione del training set.
Le metriche di deviazione standard e varianza suggeriscono che il modello è abbastanza stabile, ma c'è spazio per ridurre ulteriormente l'errore di testing con ulteriori dati o tecniche di ottimizzazione.

Curva di Apprendimento RFC:



La curva di apprendimento mostra l'andamento degli errori di training e di testing in funzione della dimensione del training set. In questa rappresentazione:

- **Asse x**: Dimensione del training set
- **Asse y**: Errore
- **Linea verde**: Errore di training
- **Linea rossa**: Errore di testing

Osservazioni:

Errore di Training:

- La linea verde è molto vicina a 0, indicando che l'errore di training è quasi nullo indipendentemente dalla dimensione del training set.
- La deviazione standard dell'errore di training è estremamente bassa (circa 0.0004), suggerendo che il modello è molto stabile nel training.
- La varianza dell'errore di training è anche estremamente bassa (1.9753086419751372e-07), confermando l'alta stabilità.

Errore di Testing:

- La linea rossa mostra che l'errore di testing diminuisce con l'aumentare della dimensione del training set, il che è un comportamento atteso in quanto più dati di training generalmente portano a un modello più accurato.
- La deviazione standard dell'errore di testing è circa 0.034, il che indica una variazione più significativa rispetto all'errore di training ma comunque abbastanza contenuta.
- La varianza dell'errore di testing è circa 0.001156, indicando una moderata dispersione nei valori dell'errore di testing.

Interpretazione:

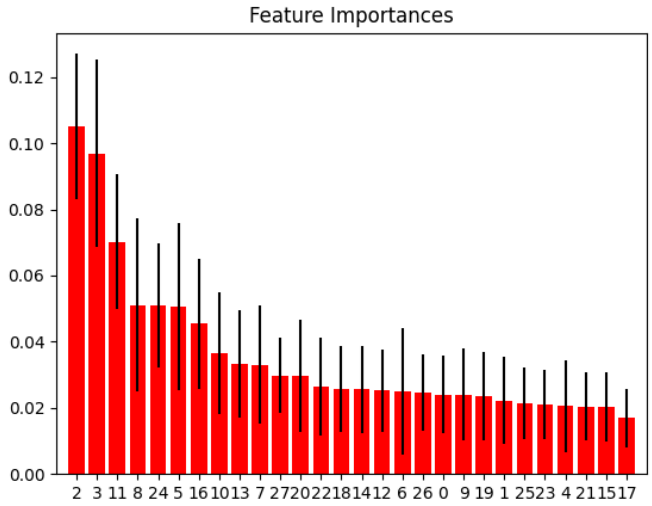
- **Underfitting**:
    - La grande differenza tra l'errore di training (praticamente nullo) e l'errore di testing (relativamente alto) suggerisce che il modello è troppo complesso e si adatta troppo bene ai dati di training, ma non è in grado di catturare la complessità dei dati di testing. Dunque la curva evidenzia un underfitting.
- **Generalizzazione**:
    - L'errore di testing sta comunque diminuendo con l'aumentare della dimensione del training set, suggerendo che con ancora più dati di training, l'errore di testing potrebbe continuare a diminuire e quindi migliorare la capacità di generalizzazione del modello.
- **Deviazione Standard e Varianza**:
    - Le basse deviazioni standard e varianze per l'errore di training indicano che il modello ha una performance consistente durante il training.
    - Le più alte (ma ancora moderate) deviazioni standard e varianze per l'errore di testing suggeriscono che, mentre il modello è abbastanza stabile, c'è ancora un po' di variazione nei risultati di testing che potrebbe essere mitigata con ulteriori dati o tecniche di regolarizzazione.

Conclusioni:

Il modello Random Forest Classifier attualmente mostra segnali di underfitting, come indicato dal fatto che l'errore di testing è significativamente più alto rispetto all'errore di training, e quest'ultimo è prossimo a 0 lungo tutta la dimensione del training set. Per migliorare le performance del modello, si potrebbe considerare l'uso di tecniche come l'incremento della complessità del modello (ad esempio, aumentando il numero di alberi o la profondità massima degli alberi), l'aggiunta di più feature rilevanti, o l'uso di un modello diverso che meglio si adatti ai dati. Le metriche di deviazione standard e varianza suggeriscono che il modello attuale non riesce a catturare sufficientemente la complessità dei dati, indicando la necessità di miglioramenti per ridurre l'errore complessivo.

5.3 Ri-esecuzione esperimenti con Feature più Importanti

Da qui possiamo vedere quali sono le feature più importanti, quindi prendiamo le prime 5 e ripetiamo gli stessi esperimenti con le feature selezionate, quindi riaddestriamo il modello per vedere come varia.



Le feature selezionate saranno quindi:

- beats
- chroma_stft
- mfcc3
- spectral_centroid
- zero_crossing_rate

ed il label sarà mfcc16, il che è essenziale perché rappresenta il target che il modello deve imparare a predire correttamente utilizzando le feature fornite.

K-Nearest Neighbors (KNN) dopo Selezione delle Feature:

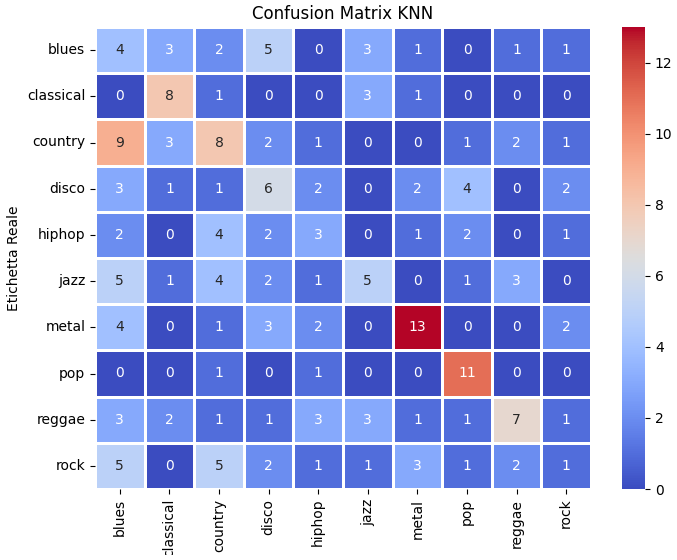
Per il modello KNN sono stati utilizzati i seguenti iperparametri:

- '*algorithm*': auto
- '*leaf_size'*: 5
- '*n_neighbors'*: 4

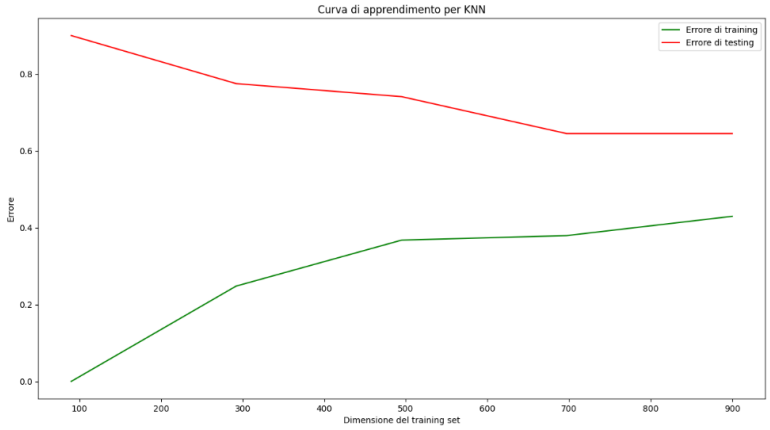
Le metriche di valutazione risultanti sono:

- **Accuracy**: 0.3300
- **Precision**: 0.3345
- **Recall**: 0.3543
- **F1 Score**: 0.3338

Confusion Matrix KNN:



Learning Curve KNN:

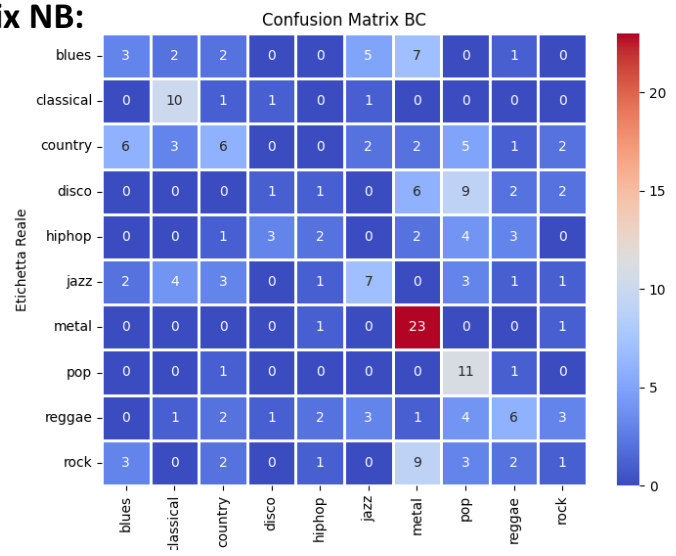


Naive Bayes dopo Selezione delle Feature:

Le metriche di valutazione per il modello Naive Bayes sono:

- **Accuracy**: 0.3500
- **Precision**: 0.3048
- **Recall**: 0.3715
- **F1 Score**: 0.3048

Confusion Matrix NB:



Random Forest Classifier (RFC):

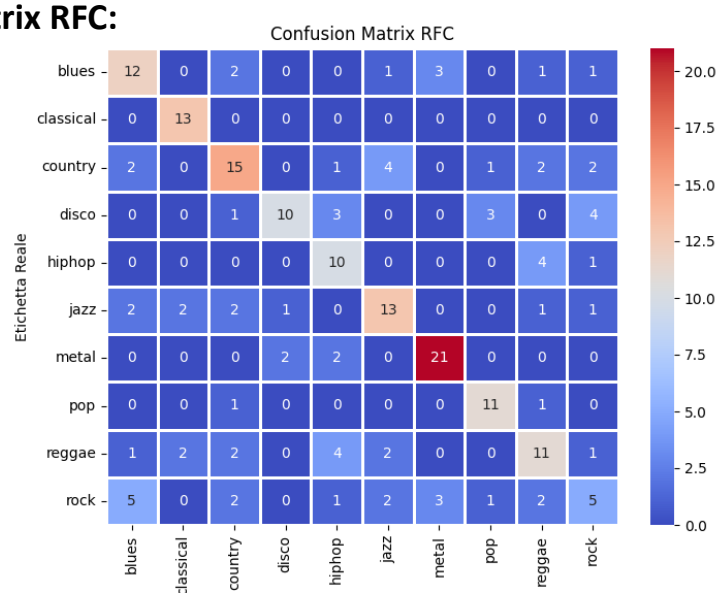
Per il modello RFC sono stati utilizzati i seguenti iperparametri:

- '*bootstrap*': True
- '*max_depth*': 15
- '*max_features*': 20
- '*min_samples_leaf*': 1
- '*min_samples_split*': 2
- '*n_estimators*': 100

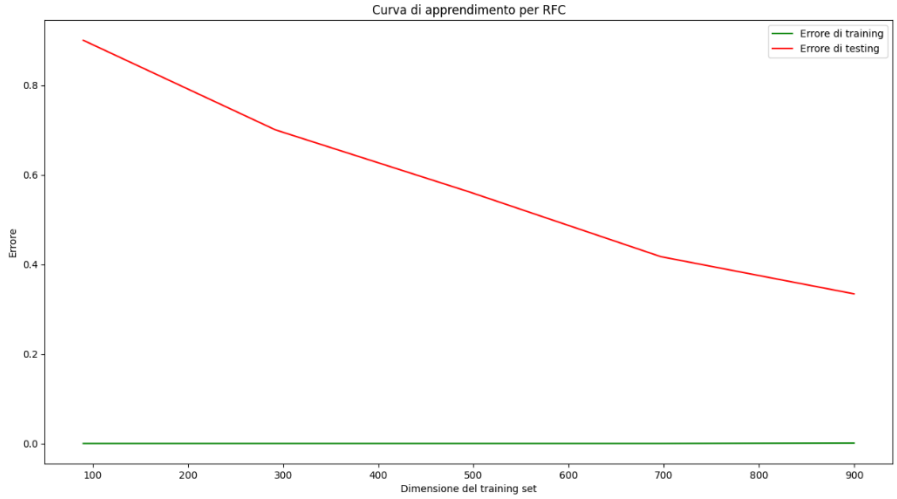
Le metriche di valutazione risultanti sono:

- **Accuracy**: 0.6050
- **Precision**: 0. 6136
- **Recall**: 0. 6136
- **F1 Score**: 0. 6073

Confusion Matrix RFC:



Learning Curve RFC:



6 Conclusioni

Dall'analisi dei risultati, possiamo trarre le seguenti conclusioni riguardo l'impatto della selezione delle 5 feature più importanti:

KNN e Naive Bayes

Per i classificatori KNN e Naive Bayes, le metriche di valutazione (Accuracy, Precision, Recall e F1 Score), la matrice di confusione e la curva di apprendimento rimangono invariate dopo la selezione delle feature più importanti. Questo suggerisce che la riduzione delle feature non influisce sulle prestazioni di questi modelli.

Random Forest Classifier

Per il Random Forest Classifier, si osserva un leggero cambiamento nelle metriche di valutazione:

Metriche	Prima esecuzione	Seconda esecuzione
Accuracy	0.6050	0.6050
Precision	0.5971	0.6136
Recall	0.6260	0.6136
F1 Score	0.6044	0.6073

Confrontando le matrici di confusione del RFC, notiamo un miglioramento in alcune classificazioni:

- **Country**: Il numero di tracce correttamente classificate è aumentato da 14 a 15, con una diminuzione degli errori di classificazione con "disco".
- **Disco**: Le tracce correttamente classificate sono passate da 10 a 11.
- **Rock**: Sebbene ci siano ancora molte tracce erroneamente classificate come altri generi, il numero di tracce correttamente classificate è aumentato da 6 a 7.

Analisi Generale

L'uso delle 5 feature più importanti ha portato a un leggero miglioramento delle prestazioni per alcuni generi, riducendo gli errori di classificazione per "country" e "disco". Questo indica che le feature selezionate come più importanti sono sufficientemente informative per distinguere la maggior parte dei generi musicali nel dataset, anche se persistono alcune confusioni.

Le curve di apprendimento risultano praticamente identiche, sia utilizzando tutte le feature che solo le 5 più importanti.

Considerazioni Finali

Le prestazioni rimangono sostanzialmente invariate con l'uso delle feature selezionate, mentre il tempo di esecuzione diminuisce. Pertanto, risulta più conveniente allenare i modelli utilizzando solo le feature migliori.