Relazione Caso di Studio Ingegneria Della Conoscenza

EchoFind: trovare “l’eco” dei tuoi gusti musicali

Sviluppo di un sistema di classificazione e raccomandazione basato su conoscenza

Giorgia Summa, 775197

g.summa4@studenti.uniba.it

Repository GitHub:

https://github.com/giosumma/EchoFind-ICON.git

A.A. 2024-2025

Sommario

1) Introduzione

Per questo caso di studio ho scelto di sviluppare un **recommender system di canzoni**, utilizzando un dataset di Spotify. Il dataset contiene circa 230.000 brani distribuiti in 26 generi musicali.

1.1) Strumenti

Ho deciso di utilizzare **Python** (www.Python.org) come linguaggio di programmazione per questo progetto grazie alla sua versatilità e ampia diffusione nel campo dell'analisi dei dati. Come servizio di hosting è stato scelto **GitHub**.

1.2) Librerie

Per l'analisi, la manipolazione dei dati e lo sviluppo degli algoritmi di raccomandazione e classificazione, sono state impiegate le seguenti librerie:

- **Scikit-learn (Sklearn):** libreria open source per apprendimento automatico in Python. Include algoritmi per classificazione, regressione e clustering, tra cui K-Means, DBSCAN, regressione logistica, classificatore bayesiano e macchine a vettori di supporto (SVM). È progettata per integrarsi facilmente con le librerie NumPy e SciPy.
- **Pandas:** libreria Python per la manipolazione e analisi dei dati. Fornisce strutture dati efficienti e strumenti per gestire tabelle numeriche, serie temporali e operazioni di filtraggio, aggregazione e trasformazione dei dati.
- **NumPy:** libreria fondamentale per il calcolo scientifico in Python, offre supporto per array multidimensionali e operazioni matematiche ad alte prestazioni.
- **Matplotlib:** libreria Python per la visualizzazione di dati. Consente di creare grafici statici, animati o interattivi, integrandosi con NumPy e supportando diversi toolkit GUI come WxPython, Qt o GTK.

2) Preprocessing per il clustering

La fase di **preprocessing** dei dati è stata cruciale per garantire la qualità delle analisi successive. Lavorando sul dataset spotify_features.csv ho eseguito le seguenti operazioni:

- **Creazione dell'indice:** ho definito un identificativo univoco basato sulle colonne track_name e artist_name.
- **Costruzione della tabella attributes:** a partire dal dataset originario, ho eliminato le colonne ridondanti o non utili al clustering (track_id, track_name, time_signature, artist_name, key).
- **Creazione della tabella genres:** per ciascun genere musicale ho introdotto una **variabile binaria** che indica la presenza (1) o l'assenza (0) del genere in relazione a un determinato brano.
- **Integrazione delle tabelle:** le due tabelle, attributes e genres, sono state unite grazie all'indice precedentemente creato, ottenendo la tabella finale denominata **songs**.
- **Eliminazione dei duplicati:** ho rimosso le occorrenze ripetute al fine di preservare l'integrità statistica dei dati.

I generi presenti nel dataset includono, tra gli altri: *Alternative, Blues, Classical, Comedy, Country, Dance, Hip-Hop, Indie, Electronic, Jazz, Folk, Pop, R&B, Rap, Reggae, Rock, Soundtrack, World, Ska, Soul*.

3) Clustering

Il **clustering** è una tecnica di machine learning non supervisionato che consente di suddividere un insieme di oggetti in gruppi omogenei, detti *cluster*. Gli elementi appartenenti a uno stesso cluster risultano simili tra loro secondo una metrica definita, mentre differiscono dagli elementi appartenenti ad altri cluster.

In un sistema di raccomandazione, il clustering riveste un ruolo fondamentale poiché:

- permette di raggruppare canzoni con caratteristiche simili,
- riduce la complessità del dataset,

- migliora le prestazioni dei modelli di raccomandazione, soprattutto se combinato con classificatori supervisionati.

3.1) K-Means

Per il progetto ho scelto di applicare l'algoritmo **K-Means**, uno dei più utilizzati nel campo del clustering. Esso mira a suddividere i dati in *k* gruppi minimizzando la distanza intra-cluster, ossia la distanza tra ciascun punto e il centroide del cluster a cui appartiene.

Il funzionamento di K-Means è iterativo:

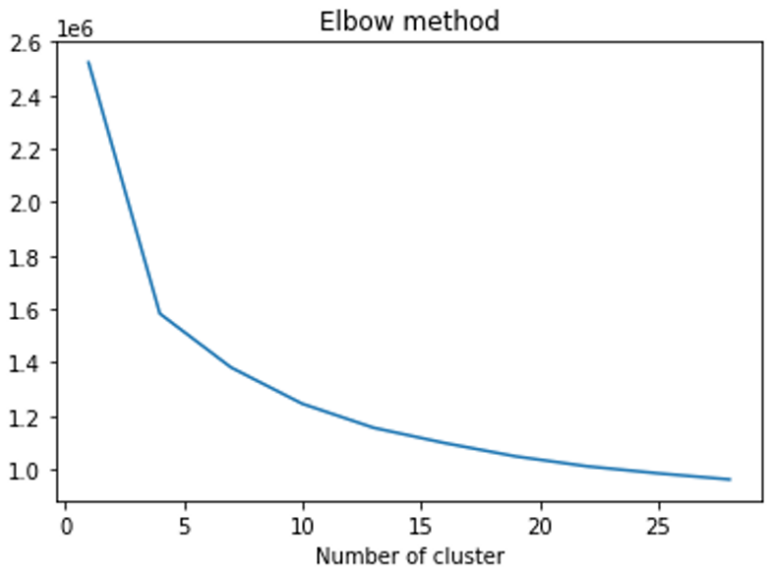
- all'inizio crea k gruppi (cluster) e assegna i brani a ciascun gruppo in modo casuale o usando alcune regole;
- calcola il centroide (il punto medio) di ogni gruppo;
- riassegna ogni brano al cluster con il centroide più vicino;
- ricalcola i centroidi in base alle nuove assegnazioni;
- ripete questo processo finché i gruppi non cambiano più, cioè quando l'algoritmo converge.

Grazie alla sua semplicità ed efficienza, K-Means si è rivelato particolarmente adatto al mio dataset.

3.2) Elbow Method

Per scegliere il numero ottimale di cluster (il valore di k) ho usato il "Metodo del Gomito" (Elbow Method).

Questo metodo è utile perché dà un criterio oggettivo per decidere quanti cluster usare. Tracciando un grafico con i valori di k sull'asse orizzontale e la somma delle distanze al quadrato (Sum of Squared Errors, SSE) tra i punti e i loro centroidi sull'asse verticale, si ottiene una curva che tende a decrescere.



Il punto in cui la curva comincia a “piegarsi” (a formare un gomito) indica il numero migliore di cluster perché, dopo quel punto aumentare k porta a un miglioramento minore. Ho quindi deciso di usare 5 cluster per ottenere una suddivisione più varia e significativa. L’uso del clustering, combinato con un modello di classificazione, mi ha permesso di migliorare la qualità del sistema di raccomandazione, ottenendo risultati più accurati e personalizzati.

4) Recommender System

A seguito del clustering, ho sviluppato il **recommender system**, basato sulla similarità tra i generi e sulle caratteristiche acustiche dei brani.

Per misurare la somiglianza tra le canzoni ho adottato la **cosine similarity**, una metrica che valuta il coseno dell’angolo tra due vettori nello spazio multidimensionale. Un angolo ridotto corrisponde a una maggiore somiglianza.

Il sistema di raccomandazione, per generare una playlist, richiede all’utente i seguenti input:

- il nome della canzone di partenza,
- l’artista,
- il numero di brani desiderati nella playlist

4.1) Guida all'uso

Visualizzazione del menù principale

Scelta dell'opzione 1

```
EchoFind: Scopri l'eco dei tuoi gusti musicali

Ciao, benvenuto nel sistema di raccomandazione di Playlist basato sulle canzoni di Spotify!
Vuoi che ti suggerisca una playlist? - Premi 1
Vuoi sapere se una canzone è popolare? - Premi 2
Vuoi uscire? - Premi 3
```

Scelta della traccia

```
Adesso dovrai suggerirmi su quale canzone basare la tua playlist
Qual'é il nome di una traccia che ti piace?
Alone

Adesso dimmi il nome dell'artista che ha scritto la traccia.
Marshmello

Quante canzoni vuoi inserire nella tua playlist?
5
```

Stampa delle raccomandazioni

```
Playlist basata sulla canzone "Alone" di Marshmello

IDOL - BTS
Save Me - BTS
RBB (Really Bad Boy) - Red Velvet
Castle on the Hill - Ed Sheeran
FIANCÉ - MINO
```

5) Preprocessing per la classificazione

Per la fase di classificazione ho effettuato ulteriori trasformazioni sul dataset:

- **key:** le 12 tonalità sono state convertite in valori numerici tramite un indice;
- **mode:** le modalità Major e Minor sono state rispettivamente codificate come 1 e 0;
- **time_signature:** i valori di battuta sono stati convertiti in numeri interi;

- **popularity:** la popolarità è stata trasformata in una variabile binaria, con valore 1 se ≥ 75 e 0 altrimenti.

6) Classificazione

I classificatori utilizzati sono stati:

- **Random Forest Classifier**,
- **K-Nearest Neighbors (KNN) Classifier**,
- **Decision Tree Classifier**,
- **Logistic Regression**

6.1) K-Nearest Neighbors

Il **K-Nearest Neighbors (KNN)** è uno degli algoritmi di machine learning più conosciuti.

Un oggetto viene assegnato alla classe predominante tra i suoi k vicini più prossimi, dove k è un intero positivo, solitamente di piccola dimensione. Ad esempio, se k = 1, l'oggetto assume la classe del vicino più vicino.

Il funzionamento del KNN si basa sulla misurazione della somiglianza tra le caratteristiche degli oggetti. Nel mio progetto, la similarità è calcolata tra le canzoni del dataset e la traccia inserita dall'utente, permettendo di predire la variabile target, ossia la popolarità del brano.

6.2) Random Forest

Il **Random Forest (RF)** è un algoritmo molto utilizzato per classificazione, regressione e altri compiti di machine learning. Funziona creando una foresta di alberi di decisione e combinando le loro predizioni. Per la classificazione, l'output finale corrisponde alla classe scelta dalla maggioranza degli alberi.

Ogni albero della foresta viene addestrato su un campione casuale del dataset mediante il metodo bootstrap, una tecnica di aggregazione che migliora la robustezza del modello. Le predizioni individuali degli alberi vengono poi

aggregate, ad esempio tramite media o voto maggioritario, per ottenere il risultato finale.

6.3) Logistic Regression

La **regressione logistica** è un modello che permette di stimare la probabilità di verificarsi di un evento sulla base di un insieme di variabili indipendenti.

Poiché il risultato rappresenta una probabilità, la variabile dipendente assume valori compresi tra 0 e 1. Per gestire correttamente questa probabilità, il modello applica la trasformazione logit, che esprime il rapporto tra probabilità di successo e probabilità di insuccesso. Questo permette di collegare linearmente le variabili indipendenti alla probabilità dell'evento, facilitando la stima e l'interpretazione del modello.

6.4) Decision Tree

Normalmente un **albero di decisione** viene costruito utilizzando tecniche di apprendimento a partire dall'insieme dei dati iniziali (data set), il quale può essere diviso in due sottoinsiemi: il training set sulla base del quale si crea la struttura dell'albero e il test set che viene utilizzato per testare l'accuratezza del modello predittivo così creato. La sua evoluzione è la tecnica del Random Forest.

6.5) Accuratezza dei classificatori

6.5.1) Train/Validation/Test

Il dataset comprende un totale di **232.725 esempi**. Per valutare correttamente le performance dei modelli, ho operato una suddivisione stratificata in tre set distinti:

- **Training set (≈64%) – 148.944 esempi:** utilizzato per addestrare i modelli, cioè per stimare i parametri che consentono al modello di apprendere le relazioni tra le feature e la popolarità.

- **Validation set (≈16%) – 37.236 esempi:** impiegato per ottimizzare l'architettura dei modelli e selezionare i migliori iperparametri.
- **Test set (≈20%) – 46.545 esempi:** utilizzato esclusivamente per valutare le performance finali dei modelli, fornendo una stima imparziale su dati non visti.

Le feature considerate comprendono proprietà audio delle tracce quali **acousticness, danceability, energy, instrumentalness, liveness, speechiness, tempo, valence**, tra le altre, selezionate in seguito a un processo di preprocessing standardizzato.

Metriche:

- **Accuracy** → proporzione di predizioni corrette
- **AUC** → capacità di distinguere tra canzoni popolari e non

Risultati principali

```
Logistic Regression - Validation Set:
Accuracy = 0.9840, AUC = 0.7593
Logistic Regression - Test Set:
Accuracy = 0.9837, AUC = 0.7573
----------------------------------------
Random Forest - Validation Set:
Accuracy = 0.9940, AUC = 0.9166
Random Forest - Test Set:
Accuracy = 0.9936, AUC = 0.9241
----------------------------------------
K-Nearest Neighbors - Validation Set:
Accuracy = 0.9815, AUC = 0.8228
K-Nearest Neighbors - Test Set:
Accuracy = 0.9812, AUC = 0.8238
----------------------------------------
Decision Tree - Validation Set:
Accuracy = 0.9849, AUC = 0.8407
Decision Tree - Test Set:
Accuracy = 0.9857, AUC = 0.8343
```

Osservazioni:

- Random Forest mostra la migliore capacità discriminativa (AUC più alta)

Ho scelto di usare anche un validation set, oltre al training e al test, per rendere la valutazione più corretta. In pratica il training set serve ad addestrare i modelli, il validation set mi permette di confrontarli e scegliere il migliore senza toccare il test, e infine il test set rimane come esame finale, indipendente, per stimare le performance reali. Se usassi solo training e test rischierei di valutare i modelli direttamente sul test e di ottenere risultati meno affidabili.

6.5.2) Cross validation

Oltre alla suddivisione classica in *training/validation/test set*, ho introdotto un'ulteriore fase di validazione, strutturata in modo analogo, al fine di ottenere una maggiore robustezza nella valutazione del modello.

Considerando che il dataset utilizzato presenta uno sbilanciamento tra le classi, ho ritenuto opportuno adottare la **stratificazione**: in questo modo, ogni sottoinsieme mantiene la stessa proporzione di classi presente nel dataset originale, evitando distorsioni nelle fasi di addestramento e validazione.

```
Distribuzione classi dopo binarizzazione:
popularity
1    176315
0     56410
```

La procedura di **k-fold cross validation** che ho applicato è la seguente:

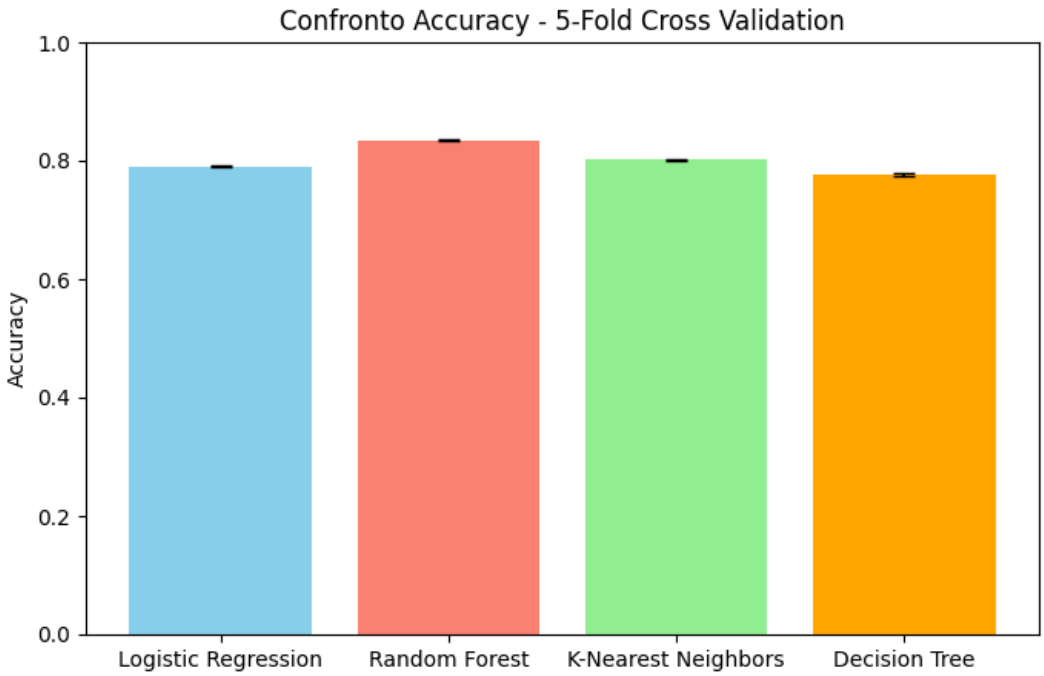
1. Suddivido il dataset in *k* sottoinsiemi (nel mio caso *k = 5*);
2. Ad ogni iterazione, utilizzo *k-1* fold per l'addestramento e il fold rimanente come validation set;
3. Ripeto il processo per tutti i fold;
4. Calcolo la media delle metriche di valutazione (in questo caso *Accuracy* e *AUC*).

Per implementare questa procedura ho utilizzato la funzione **StratifiedKFold** disponibile nella libreria *scikit-learn*.

Le scelte progettuali principali sono state:

- definire il valore di *k* (numero di fold);
- stabilire se applicare o meno la stratificazione (nel mio caso ho utilizzato la variante stratificata).

Analisi dei risultati




```
Tabella riassuntiva cross-validation:
                Modello  Accuracy Media  Deviazione Std Accuracy  AUC Media  Deviazione Std AUC
0   Logistic Regression        0.790108                 0.001400   0.745906            0.001961
1         Random Forest        0.834839                 0.001418   0.868349            0.001670
2   K-Nearest Neighbors        0.801788                 0.001187   0.759615            0.002440
3         Decision Tree        0.776741                 0.002135   0.687072            0.003354
```

Nella tabella seguente sono riportati i valori medi di *Accuracy* e *AUC*, insieme alla rispettiva deviazione standard:

- **Random Forest** si conferma il modello più performante, con un'*accuracy media* pari a **0.835** e un *AUC medio* di **0.868**. Inoltre, le deviazioni standard molto contenute indicano un'elevata stabilità del modello nei diversi fold.
- **Logistic Regression** e **K-Nearest Neighbors** presentano risultati intermedi. Logistic Regression mostra un buon compromesso tra

accuracy (0.790) e *AUC* (0.746), mentre KNN raggiunge un'accuracy leggermente superiore (0.802), ma un'AUC minore (0.760).

- **Decision Tree**, al contrario, è il modello meno competitivo: sia l'*accuracy* (0.777) che l'*AUC* (0.687) risultano significativamente inferiori rispetto agli altri algoritmi, e la deviazione standard più elevata evidenzia una maggiore instabilità tra i fold.

In conclusione, la cross-validation evidenzia come il **Random Forest** risulti il modello più stabile.

6.5.3) Confronto tra Train/Validation/Test e Cross-Validation

I risultati ottenuti tramite la suddivisione ***train/validation/test*** mostrano valori di accuracy molto elevati (oltre 0.98 per tutti i modelli). Tuttavia, questi valori possono sovrastimare le prestazioni effettive a causa della dipendenza dal singolo split.

Al contrario, la procedura di ***Stratified 5-Fold Cross Validation*** restituisce valori più contenuti (Accuracy tra 0.77 e 0.83), ma più attendibili e stabili, in quanto derivano dalla media di diverse suddivisioni del dataset.

Nonostante le differenze nei valori assoluti, l'ordine di performance tra i modelli rimane coerente: **Random Forest** risulta il più performante, seguito da **Logistic Regression** e **KNN**, mentre il **Decision Tree** si conferma il modello meno efficace.

6.6) Guida all'uso

Scelta della predizione sulla popolarità e

inserimento del nome del brano e dell'artista

```
Vuoi che ti suggerisca una playlist? - Premi 1
Vuoi sapere se una canzone è popolare? - Premi 2
Vuoi uscire? - Premi 3
2
Adesso dovrai suggerirmi la canzone su cui predire la popolaritá!
Qual'é il nome della traccia?
Alone
Adesso dimmi il nome dell'artista che ha scritto la traccia.
Marshmello
```

Risultato ottenuto tramite Random Forest

```
Canzone trovata nel dataset!
Quale classificatore vuoi utilizzare?
Random Forest Classifier - Premi 1
K-Nearest Neighbors Classifier - Premi 2
Decision Tree Classifier - Premi 3
Logistic Regression - Premi 4

1
La canzone é popolare!
```

7) Conclusioni

Il progetto EchoFind rappresenta un sistema di raccomandazione musicale. Dopo un attento preprocessing, i brani sono stati raggruppati in cluster con K-Means, mentre il recommender system basato su cosine similarity ha permesso di generare playlist coerenti con le preferenze dell'utente.

Per la classificazione della popolarità dei brani sono stati testati diversi algoritmi supervisionati, tra cui Random Forest, che ha mostrato le migliori performance valutate tramite train/validation/test set e Stratified 5-Fold Cross Validation.

La semplicità del progetto ha rappresentato un punto di forza, in quanto mi ha consentito di focalizzarsi sui concetti fondamentali. Nonostante la sua

essenzialità, è stato possibile applicare in maniera concreta alcuni dei principi studiati nell'ambito dell'Ingegneria della Conoscenza, approfondendo tematiche quali clustering, classificazione, sistemi di raccomandazione basati su conoscenza e valutazione delle performance dei modelli. I risultati ottenuti dimostrano come l'integrazione di tali tecniche possa condurre alla realizzazione di sistemi di suggerimento musicale personalizzati e affidabili.

Sviluppi futuri

Il lavoro può essere ampliato ed esteso in diverse direzioni:

1. **Integrazione di nuove feature:** includere variabili aggiuntive come testi delle canzoni, recensioni o metadati esterni per arricchire il sistema di raccomandazione.
2. **Valutazione con utenti reali:** testare il sistema in un contesto reale, raccogliendo feedback da utenti, per validarne l'efficacia non solo da un punto di vista quantitativo (metriche di accuratezza), ma anche qualitativo (soddisfazione dell'ascoltatore).
3. **Visualizzazione interattiva:** sviluppo di interfaccia grafica

Motivazione della scelta degli algoritmi

1. **K-Means (Clustering)**
   Ho scelto K-Means per la fase di clustering perché il dataset contiene circa 230.000 brani distribuiti in 26 generi musicali, rendendo necessario un metodo efficiente per raggruppare i brani in insiemi omogenei. Questo approccio:
   - Riduce la complessità dei dati per la fase di raccomandazione.
   - Facilita l'individuazione di somiglianze sonore e caratteristiche comuni tra le canzoni.
   - Si integra bene con i sistemi di raccomandazione basati sulla similarità, come la cosine similarity.

Per la classificazione della popolarità dei brani ho utilizzato **KNN, Random Forest, Logistic Regression e Decision Tree**.

- **KNN**: semplice e basato sulla similarità, utile per ragionamento su feature continue
- **Random Forest**: robusto e stabile, riduce overfitting, adatto a dataset ampi e variabili eterogenee
- **Logistic Regression**: interpretabile, fornisce probabilità e gestione dell'incertezza
- **Decision Tree**: modello predittivo che utilizza nodi e foglie per classificare i brani in base alle loro caratteristiche

8) Riferimenti Bibliografici

[1] Scikit-learn – Machine Learning in Python: https://scikit-learn.org/
[2] Pandas – Data analysis and manipulation tool: https://pandas.pydata.org/
[4] Matplotlib – Visualization with Python: https://matplotlib.org/
[5] NumPy – Fundamental package for scientific computing: https://numpy.org/