Università degli Studi di Salerno

Progetto di Fondamenti di Intelligenza Artificiale

PLAYLIST-GENERATOR

30/01/2025

AUTORE	MATRICOLA
Carmine Citro	0512116133
Andrea Filipuzzi	0512117756
Davide Santillo	0512118725

https://github.com/FIL-04/ProgettoFIA.git

Sommario

1 INTRODUZIONE 3
2 DEFINIZIONE DEL PROBLEMA 3
2.1 OBIETTIVI 3
2.2 SPECIFICA PEAS 4
2.3 ANALISI DEL PROBLEMA 6
3 SOLUZIONE DEL PROBLEMA 7
3.1 Importazione dei Dati 7
3.2 Pulizia del Dataset 7
3.3 Assegnazione del Mood 7
3.4 Divisione del Dataset 8
3.5 Selezione del Mood 8
3.6 Calcolo di Nuove Caratteristiche 9
3.7 Clustering 9
3.8 Visualizzazione dei Cluster 11
3.9 Generazione della Playlist 11
3.10 Esportazione della Playlist 12
3.11 Gestione secondaria della playlist 12
3.12 Ritorno dei Risultati 13

1 INTRODUZIONE

La musica ha un ruolo unico nel modellare le nostre emozioni e stati d'animo. Questo progetto nasce dall'idea di utilizzare tecniche di intelligenza artificiale per analizzare le caratteristiche musicali di un brano, come energia, ballabilità e positività, al fine di associare ciascuna canzone a un determinato stato emotivo (o "mood"). Successivamente, viene creato un sistema di clustering che permette di organizzare i brani in playlist coerenti, offrendo un'esperienza di ascolto personalizzata per l'utente.

2 DEFINIZIONE DEL PROBLEMA

2.1 OBIETTIVI

Analizzare le caratteristiche musicali: Utilizzare parametri come energia, ballabilità e valenza emotiva per definire il mood di una canzone.

Creare playlist emozionali: Generare playlist che riflettano lo stato d'animo scelto dall'utente.

Clustering basato su caratteristiche: Applicare algoritmi di clustering (come K-Means) per raggruppare canzoni con caratteristiche simili.

Visualizzazione e interpretazione: Rappresentare graficamente i cluster per una migliore comprensione della distribuzione dei brani.

2.2 SPECIFICA PEAS

Performance Measure (Misure di Prestazione)

Il successo del sistema viene misurato in base a:

Accuratezza dell'assegnazione del mood: La capacità del sistema di classificare correttamente i brani in base alle emozioni, utilizzando parametri come energia, ballabilità, e valenza.

Coerenza dei cluster: I brani raggruppati nello stesso cluster devono condividere caratteristiche musicali simili e corrispondere al mood selezionato.

Soddisfazione dell'utente: La playlist generata deve risultare soddisfacente e coerente con il mood scelto dall'utente.

Efficienza computazionale: Il sistema deve processare i dati e generare playlist in tempi ragionevoli.

Facilità d'uso: L'interazione utente-sistema (es. selezione del mood e generazione della playlist) deve essere semplice e intuitiva.

Environment (Ambiente)

L'ambiente in cui opera il sistema è caratterizzato da:

Dataset musicale: Una collezione di brani con parametri predefiniti come energia, ballabilità, acusticità e valenza. I dati provengono da file CSV o altre fonti di dataset musicali.

Ambiente utente: Un contesto virtuale in cui l'utente seleziona un mood (es. Felicità, Relax) e riceve in output una playlist personalizzata.

Contesto limitato: Il sistema opera su un dataset statico e predefinito (non aggiornato in tempo reale), limitando il numero di canzoni disponibili.

Actuators (Attuatori)

Gli attuatori sono le azioni intraprese dal sistema per raggiungere gli obiettivi:

Generazione di playlist: Creazione di una playlist basata sul mood selezionato dall'utente.

Clustering dei brani: Raggruppamento delle canzoni in cluster basati su caratteristiche musicali.

Esportazione della playlist: Salvataggio della playlist generata in formato CSV per consentire ulteriori utilizzi.

Sensors (Sensori)

I sensori rappresentano le informazioni che il sistema raccoglie per prendere decisioni:

Caratteristiche musicali: Parametri estratti dal dataset, come energia, ballabilità, valenza emotiva, acusticità.

Input utente: Il mood selezionato dall'utente (es. Felicità, Tristezza, Carica).

Metriche derivate: Nuove feature calcolate, come:

- Rapporto tra ballabilità e positività (dance_valence_ratio).
- Differenza tra energia e acusticità (energy_acoustic_diff).
- Combinazione di energia e ballabilità/acusticità (energy_dance_combo, acoustic_valence_combo).

2.3 ANALISI DEL PROBLEMA

La musica è un mezzo potente per esprimere e influenzare emozioni. Gli ascoltatori spesso scelgono brani o playlist in base al proprio stato d'animo o alle attività che stanno svolgendo. Tuttavia, trovare canzoni che si adattino perfettamente a un particolare mood può essere un processo lungo e soggettivo, soprattutto quando si dispone di grandi librerie musicali. Questo problema evidenzia la necessità di un sistema automatizzato che analizzi le caratteristiche musicali di un brano e crei playlist coerenti con uno stato emotivo selezionato.

3 SOLUZIONE DEL PROBLEMA

Per affrontare il problema di generare playlist basate su emozioni e caratteristiche delle canzoni, è stata implementata una pipeline di analisi dei dati e clustering. Ecco i principali passi del processo:

3.1 Importazione dei Dati

- Il dataset musicale è stato importato utilizzando **pandas**.
- Sono state visualizzate le prime righe del dataset per verificare il contenuto e la struttura.

3.2 Pulizia del Dataset

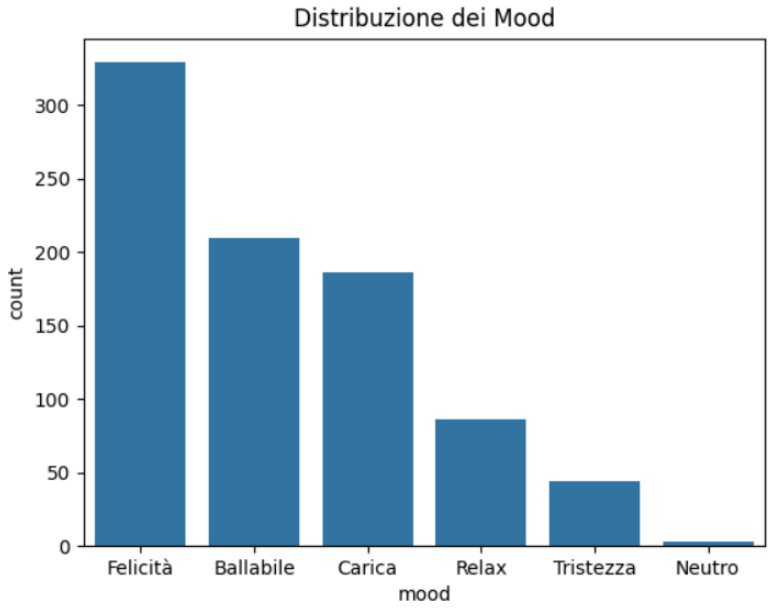
- Sono state rimosse colonne non necessarie: artist_count, in_spotify_charts, in_apple_charts, in_deezer_charts, in_shazam_charts, released_month, released_day.
- È stato controllato il dataset per la presenza di valori nulli e le righe contenenti valori mancanti sono state eliminate.

3.3 Assegnazione del Mood

- È stata applicata una funzione personalizzata (assign_mood) per aggiungere una nuova colonna chiamata mood, basata su parametri come energia, valence, e altre caratteristiche delle canzoni.

```
def assign_mood(row):
    if row['valence_%'] >= 65 and row['energy_%'] >= 55 and row['danceability_%'] >= 60:
        return 'Felicità'  # Ridotte leggermente le soglie
    elif row['energy_%'] <= 60 and row['valence_%'] >= 20 and row['valence_%'] <= 70 and row['acousticness_%'] >= 35 and row['bpm'] <= 130:
        return 'Relax'  # Maggiore tolleranza su bpm e acousticness
    elif row['valence_%'] <= 40 and row['energy_%'] <= 50 and row['acousticness_%'] >= 25:
        return 'Tristezza'  # Soglie più ampie
    elif row['energy_%'] >= 60 and row['valence_%'] >= 25 and row['valence_%'] <= 80 and row['bpm'] >= 110:
        return 'Carica'  # Aumentato il range di valence
    elif row['danceability_%'] >= 60 and row['bpm'] >= 80 and row['bpm'] <= 160 and row['energy_%'] >= 35:
        return 'Ballabile'  # Maggiore flessibilità su energia e bpm
    else:
        # Logica di fallback per ridurre i Neutri
        if row['valence_%'] >= 50 or row['energy_%'] >= 50:
            return 'Felicità'
        elif row['acousticness_%'] >= 25 and row['bpm'] <= 110:
            return 'Relax'
        elif row['energy_%'] >= 55 and row['bpm'] >= 100:
            return 'Carica'
        elif row['danceability_%'] >= 50:
            return 'Ballabile'
        else:
            return 'Neutro'
```

- La distribuzione dei mood è stata visualizzata tramite grafici scatter e grafici a barre (utilizzando **Seaborn**).



3.4 Divisione del Dataset

- Il dataset è stato diviso in due sottoinsiemi:
  - **Train set**: 70% dei dati, usato per l'addestramento e l'analisi.
  - **Test set**: 30% dei dati, usato per la validazione e la creazione di playlist.
- I due sottoinsiemi sono stati esportati in file CSV per future analisi o utilizzi.

3.5 Selezione del Mood

- L'utente ha avuto la possibilità di selezionare un mood da una lista predefinita (Felicità, Relax, Tristezza, Carica, Ballabile).

```
Scegli un mood (Felicità, Relax, Tristezza, Carica,Ballabile): Ballabile
```

- Le canzoni del test set appartenenti al mood selezionato sono state filtrate per ulteriori analisi.

3.6 Calcolo di Nuove Caratteristiche

Per ogni canzone del mood selezionato, sono state calcolate nuove caratteristiche per arricchire l'analisi:

- **Rapporto Ballabilità/Felicità (dance_valence_ratio)**: rapporto tra danceability e valence, utile per valutare quanto una canzone è ballabile rispetto alla sua positività.
- **Differenza Energia-Acustica (energy_acoustic_diff)**: differenza tra energy e acousticness, utile per confrontare l'energia di una traccia rispetto alla sua dolcezza.
- **Combinazione Energia-Ballabilità (energy_dance_combo)**: prodotto tra energy e danceability, indicatore dell'intensità della canzone.
- **Combinazione Acustica-Felicità (acoustic_valence_combo)**: prodotto tra acousticness e valence, per individuare tracce acustiche con positività elevata.

```
101 mood_st['dance_valence_ratio'] = mood_st['danceability_%'] / (mood_st['valence_%'] + 1e-5)
102 mood_st['energy_acoustic_diff'] = mood_st['energy_%'] - mood_st['acousticness_%']
103 mood_st['energy_dance_combo'] = mood_st['energy_%'] * mood_st['danceability_%']
104 mood_st['acoustic_valence_combo'] = mood_st['acousticness_%'] * mood_st['valence_%']
```

3.7 Clustering

- Sono stati selezionati i nuovi parametri calcolati come caratteristiche per il clustering.
- I dati sono stati normalizzati utilizzando **StandardScaler**, che calcola la media e la deviazione standard per ogni caratteristica, standardizzando ogni valore, dopo la standardizzazione ogni caratteristica avrà una media pari a 0 e deviazione standard pari a1, significa che i valori saranno trasformati in una distribuzione normale standard, rendendo le diverse caratteristiche comparabili.Molti algoritmi (es. K-Means, SVM, PCA) funzionano meglio quando i dati sono su scale simili, poiché sono sensibili all'ampiezza dei valori.Senza la standardizzazione, caratteristiche con valori molto grandi potrebbero dominare il processo di apprendimento rispetto a quelle con valori piccoli.

- È stato utilizzato l'algoritmo **KMeans** per suddividere le canzoni in un numero dinamico di cluster (determinato tramite la funzione personalizzata dynamic_clusters).

```
def dynamic_clusters(song_count):
    return max(1, song_count // 25)  # Ogni cluster avrà almeno 25 canzoni
```

L'algoritmo **K-Means** è uno dei metodi di **clustering** più utilizzati in **apprendimento non supervisionato**. Ha l'obiettivo di suddividere un insieme di dati in un numero predefinito di gruppi (o cluster) in modo che i punti all'interno di un cluster siano più simili tra loro rispetto ai punti appartenenti a cluster diversi.

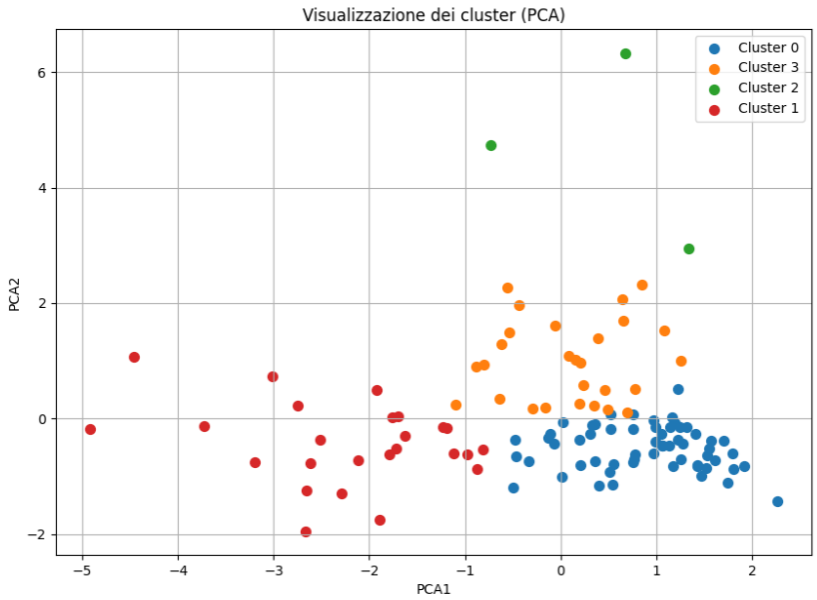
Passaggi dell'algoritmo K-Means

1. **Inizializzazione dei centroidi**:
    - Si scelgono casualmente KKK centroidi iniziali, dove KKK è il numero di cluster desiderato. Ogni centroide rappresenta il centro di un cluster.
2. **Assegnazione dei punti ai cluster**:
    - Per ogni punto nel dataset, si calcola la **distanza** da ciascun centroide.
    - Il punto viene assegnato al cluster il cui centroide è il più vicino (spesso si usa la distanza euclidea).
3. **Ricalcolo dei centroidi**:
    - Dopo che tutti i punti sono stati assegnati a un cluster, si ricalcola la posizione di ciascun centroide come la **media** dei punti assegnati al cluster.
4. **Ripetizione**:
    - I passaggi 2 e 3 vengono ripetuti fino a quando:
        - I centroidi non cambiano più (convergenza).
        - Oppure viene raggiunto un numero massimo di iterazioni.

- Ogni canzone è stata assegnata a un cluster e sono state calcolate le medie delle caratteristiche per ciascun cluster, fornendo un profilo chiaro delle canzoni raggruppate.

3.8 Visualizzazione dei Cluster

- Le dimensioni dei dati sono state ridotte a due componenti principali tramite **Principal Component Analysis (PCA)** per visualizzare i cluster in uno spazio bidimensionale.
- È stato creato un grafico scatter, dove ogni punto rappresenta una canzone, colorato in base al cluster di appartenenza.



3.9 Generazione della Playlist

- È stato selezionato casualmente un cluster dall'insieme dei cluster creati.
- È stata generata una playlist con le canzoni appartenenti al cluster scelto.
- La playlist è stata esportata in un file CSV (Playlist_scelta.csv).

track_name	artist(s)_name	cluster
Mïï¿½ï¿½n	Tini, Maria Becerra	1
Lovezinho	Treyce	1
CHORRITO PA LAS ANIMAS	Feid	1
La Santa	Daddy Yankee, Bad Bunny	1
Raindrops (Insane) [with Travis Scott]	Travis Scott, Metro Boomin	1
Envolver	Anitta	1
I Like You (A Happier Song) (with Doja Cat)	Post Malone, Doja Cat	1
Anti-Hero	Taylor Swift	1
Circo Loco	Drake, 21 Savage	1
MORE	j-hope	1
sentaDONA (Remix) s2	Luï¿½ï¿½sa Sonza, MC Frog, Dj Gabriel do Borel, Davi K	1
Still With You	Jung Kook	1
Search & Rescue	Drake	1
Niï¿½ï¿½a Bo	Sean Paul, Feid	1
Streets	Doja Cat	1
X ï¿½ï¿½LTIMA	Daddy Yankee, Bad Bunny	1
Nail Tech	Jack Harlow	1
Blank Space	Taylor Swift	1
Stan	Eminem, Dido	1
Nostï¿½ï¿½l	Chris Brown, Rvssian, Rauw Alejandro	1
Is There Someone Else?	The Weeknd	1
Gangsta's Paradise	Coolio, L.V.	1
BackOutsideBoyz	Drake	1
Coco Chanel	Bad Bunny, Eladio Carrion	1

3.10 Esportazione della Playlist

- La playlist generata è stata salvata su disco per condividerla o utilizzarla in altre applicazioni.

3.11 Gestione secondaria della playlist

La playlist generata viene gestita tramite il modulo esterno playlistSpotify che, presumibilmente, permette di salvare o interagire con la playlist attraverso una piattaforma di streaming musicale (ad esempio, Spotify).

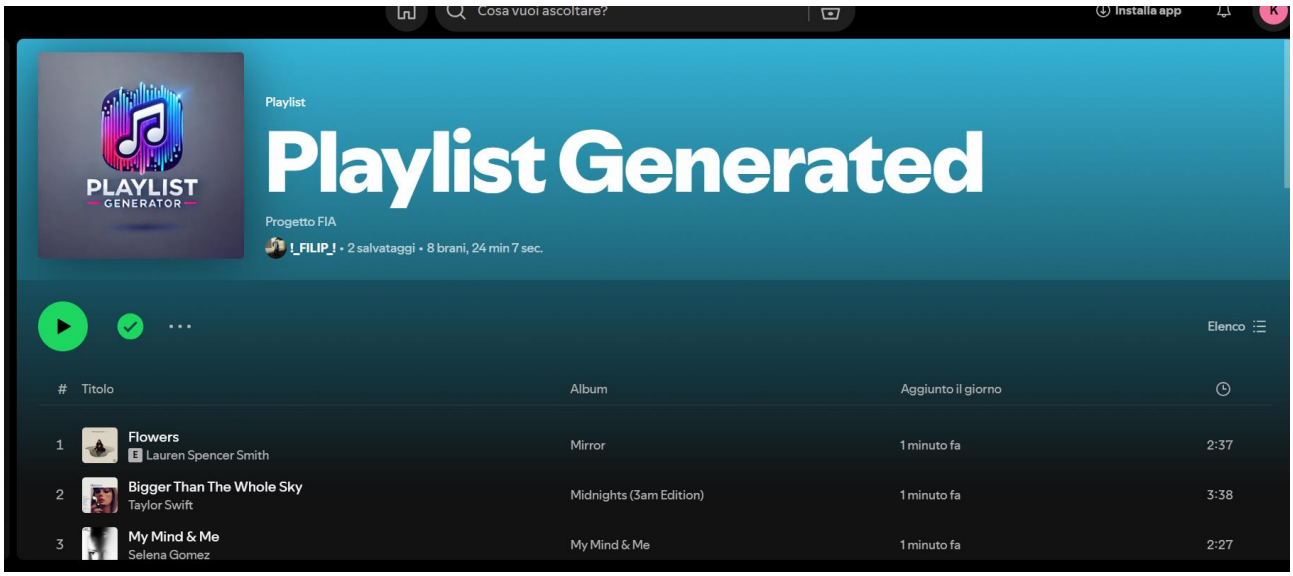
```python
# Funzione principale che gestisce la playlist
def manage_playlist(playlist):
    # Verifica se la playlist è vuota
    if not is_playlist_empty(playlist_id):
        print("La playlist non è vuota, svuotando...")
        clear_playlist(playlist_id)  # Svuota la playlist se non è vuota

    # Carica il CSV con i dati delle canzoni
    #df = pd.read_csv('Playlist_scelta.csv')  # Assicurati che il percorso del CSV sia corretto

    # Assicurati che il CSV abbia le colonne "Nome" e "Artista" (o i nomi delle colonne corretti)
    tracks_to_add = [(row['track_name'], row['artist(s)_name']) for _, row in playlist.iterrows()]

    # Cerca le tracce e ottieni gli URI
    track_uris_to_add = [search_track(name, artist) for name, artist in tracks_to_add]
    track_uris_to_add = [uri for uri in track_uris_to_add if uri is not None]

    # Aggiungi i brani alla playlist
    if track_uris_to_add:
        add_tracks_to_playlist(playlist_id, track_uris_to_add)
    # Costruisci il link Spotify
    playlist_link = f"https://open.spotify.com/playlist/{playlist_id}"
    return playlist_link
```



3.12 Ritorno dei Risultati

- La funzione restituisce tre valori:

  playlist_scelta: un DataFrame contenente le canzoni della playlist generata.

  filePath: il percorso del file CSV contenente la playlist.

  Collegamento diretto a spotify.