|  |  |
| --- | --- |
| Università degli studi di Salerno – FonDamenti di intelligenza Artificiale (2020/2021) | AUTORI  Angelo Afeltra  Antonio Cirillo  Antonio Cacciapuoti  Carmine Amendola |

1. **Idea del progetto**

Il nostro progetto di Intelligenza Artificiale nasce con l’obiettivo di riconosce i generi musicali grazie ai vari algoritmi di cluster, basandosi sulle feature delle canzoni (BPM, Energy, Danceability, etc.)

L’idea base era quello di sviluppare una web-app che permettesse ad un utente di poter selezionare un genere musicale e visualizzare di conseguenza un elenco canzoni collegate al genere da poter ascoltare.

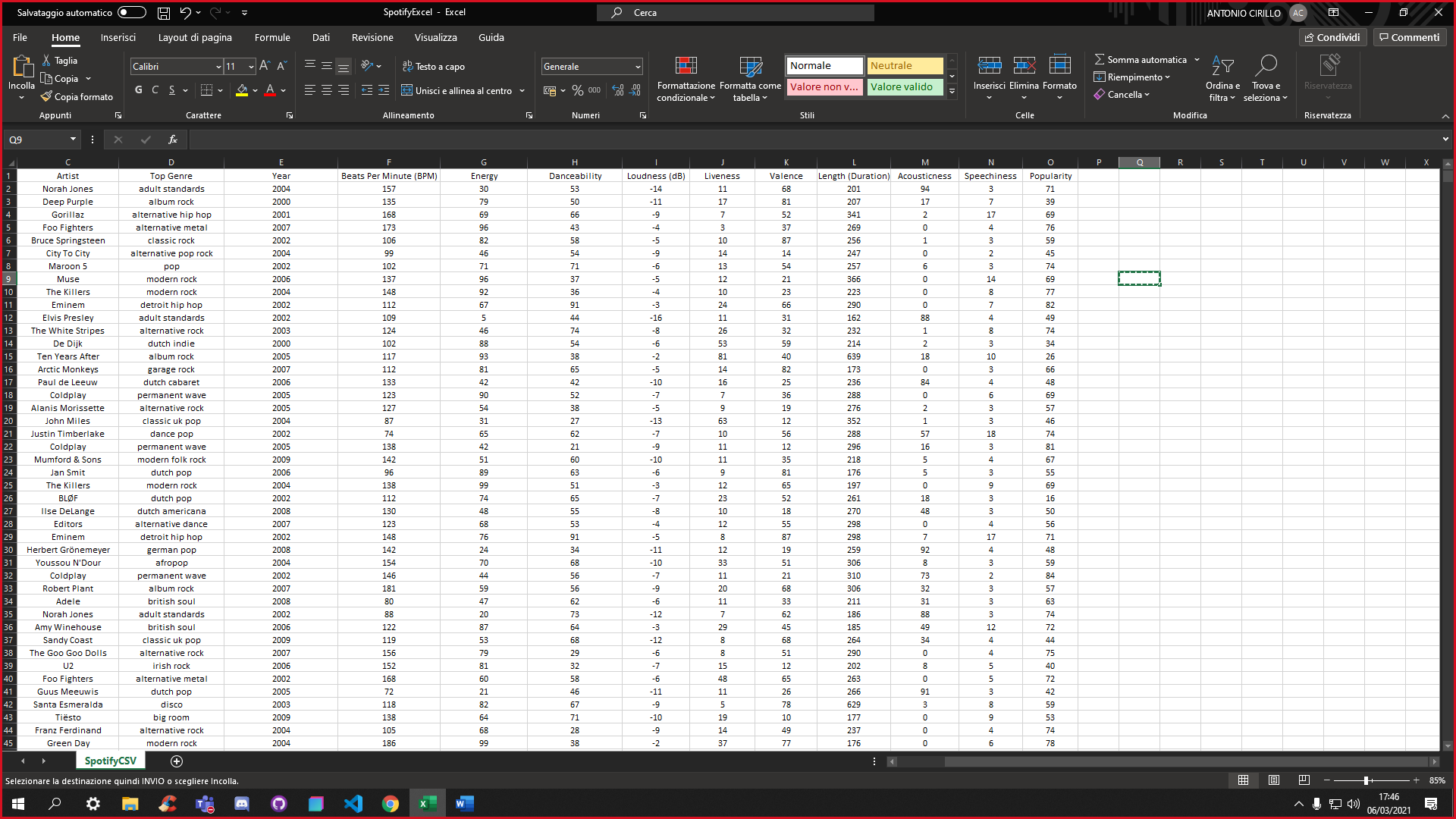
1. **Obiettivi del progetto**

In prima analisi, sono state scelti i dati con i quali l’IA si sarebbe interfacciata e le tecnologie da utilizzare per progettarla.  
Come già detto nel punto precedente, l’ obiettivo principale del progetto è quello di selezionare un elenco di canzoni in base ad un genere specifico. Nel nostro caso le canzoni saranno selezionate da un dataset rappresentativo ed estratto dalla piattaforma di streaming musicale **Spotify**.

Il secondo obiettivo è progettare una web-app che interagisca con l’IA, in modo da permettere a chiunque di poterla utilizzare. Per questo motivo, abbiamo deciso di sviluppare l’intero progetto in **JavaScript**, con l’utilizzo di **Node.js** per poter realizzare il front-end dell’applicazione e l’interazione con **l’API** di Spotify.

1. **Approccio al progetto**

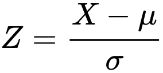
Il data-set utilizzato è così strutturato:



Le features per ogni canzone sono, rispettivamente:  
- **Beats Per Minute (BPM)**: unità di misura della frequenza;  
- **Energy**: unità di misura che indica l’intensità della canzone e l’energia trasmessa da quest’ultima;  
- **Danceability**: indica quanto la traccia è adatta ad essere ballata o meno;  
- **Loudness** **(dB)**: unità di misura della qualità del suono;  
- **Liveness**: indica la presenza o meno di spettatori durante l’esecuzione della traccia;  
- **Valence**: indica la positività trasmessa dalla canzone;  
- **Length** (Duration): lunghezza della traccia;  
- **Acousticness**: misura l’acustica della canzone, ossia quanto il suono è modificato in modo elettronico;  
- **Speechiness**: indica quanto è parlata la traccia;  
- **Popularity**: popolarità della canzone.

I primi due problemi da affrontare a questo punto sono stati **la standardizzazione** e **la normalizzazione** dei dati, in quanto presenti in un formato non unificato.

Per la standardizzazione abbiamo utilizzato la formula matematica seguente, avvalendoci della tecnica z-score, sostituendo ad ogni valore originariamente presente il valore derivante dall’utilizzo della suddetta tecnica:



Di seguito riportiamo gli snippet di codice relativi:

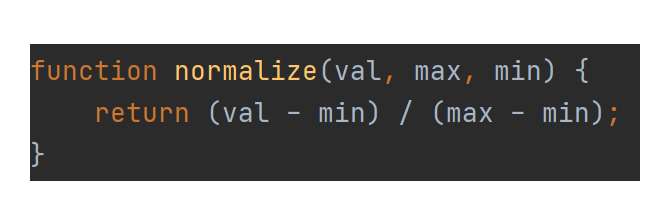


Analogamente per la normalizzazione, per ottenere dati compresi tra 0 e 1, abbiamo utilizzato la formula:

Immagine che contiene testo

Descrizione generata automaticamente

Riportiamo anche in questo caso lo snippet di codice creato:



Dopo aver normalizzato e standardizzato i dati del nostro data-set, il nostro obiettivo è stato quello di riuscire a diminuire il numero dei features al fine di poter rappresentare i dati in uno spazio tridimensionale. Ci siamo quindi avvalsi della **PCA** (principal component analysis), più precisamente della libreria **pca-js**.

La PCA presente in tale libreria è basata su una risoluzione matriciale. La prima fase della suddetta consiste nel calcolo degli autovettori e autovalori:

**const vectors = pca.getEigenVectors(data);**

Successivamente al calcolo degli autovettori si è presentata la necessità di scegliere quali e quanti usarne per ottenere una **precisione accettabile**. Avendo considerato una precisione dell’80% come ottimale per il nostro progetto, abbiamo deciso di utilizzare la combinazione di vettori che ci garantisse il suo raggiungimento.

In seguito a questa fase di preparazione l’unico passo da effettuare è stato l’esecuzione della nostra PCA.

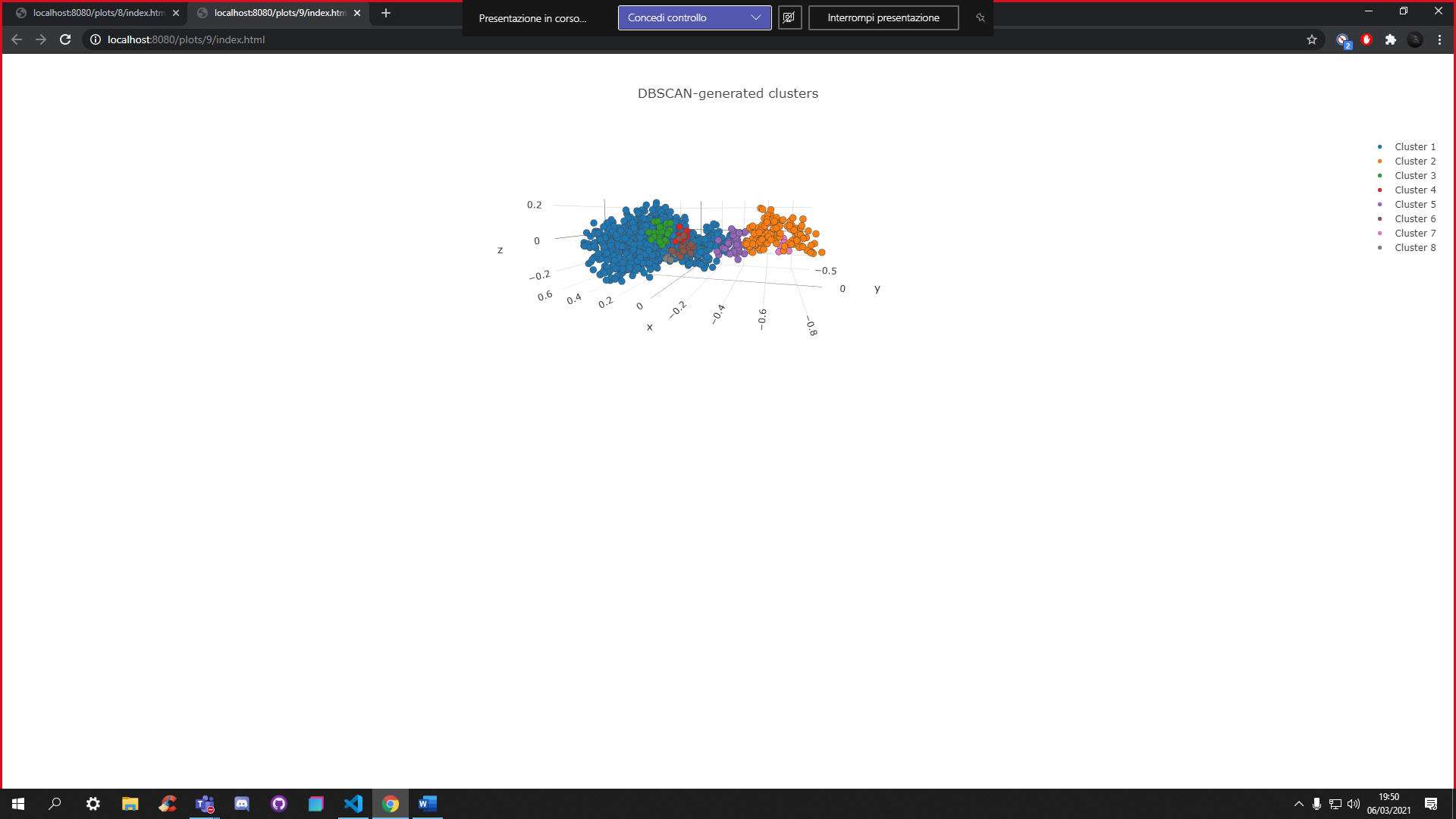
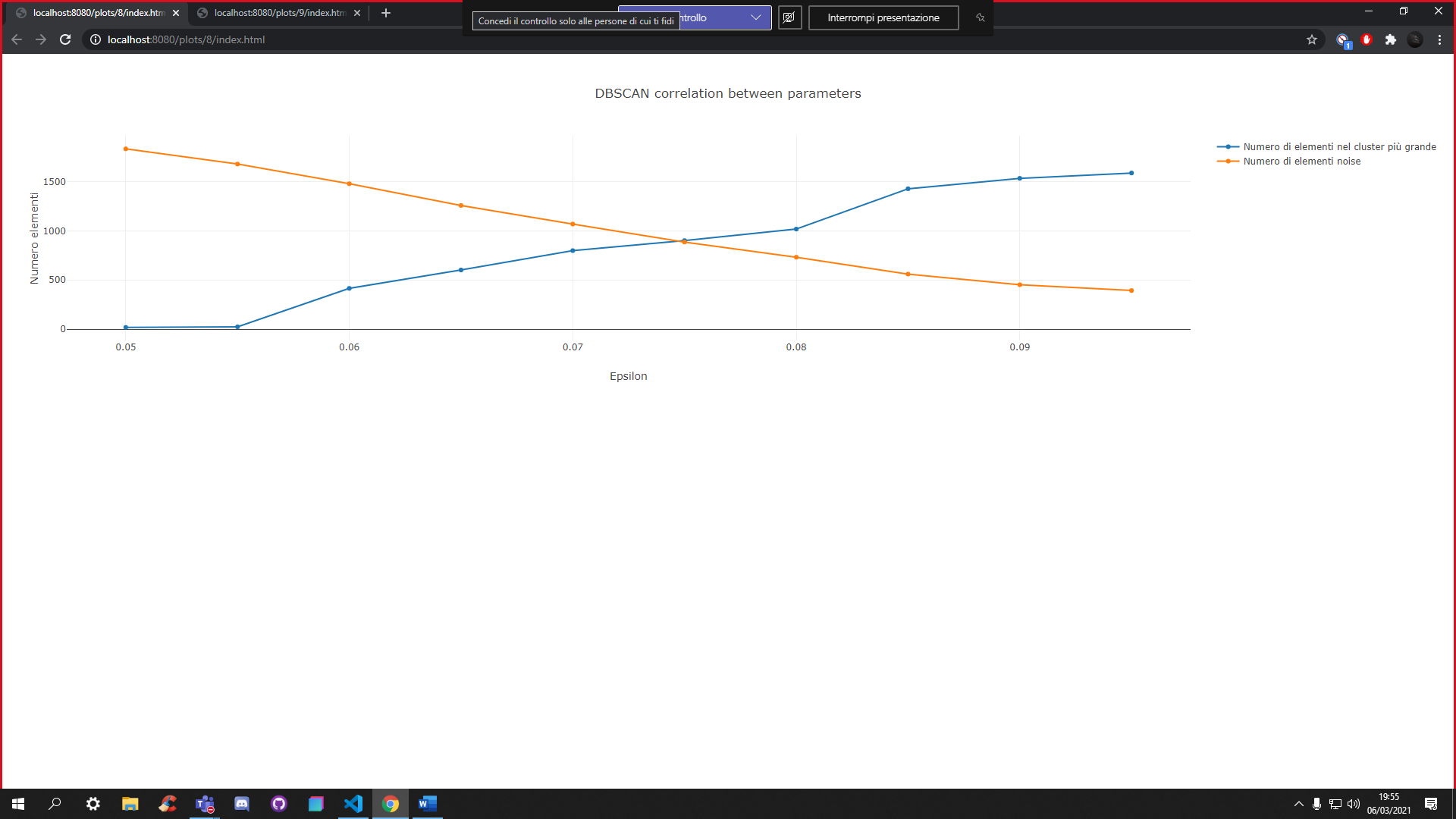
Utilizzando la funzione :

**pca.computeAdjustedData(data, vectors[0], vectors[1], vectors[2])**

siamo stati in grado di ridurre il numero delle nostre features da n ad m, con m<n.

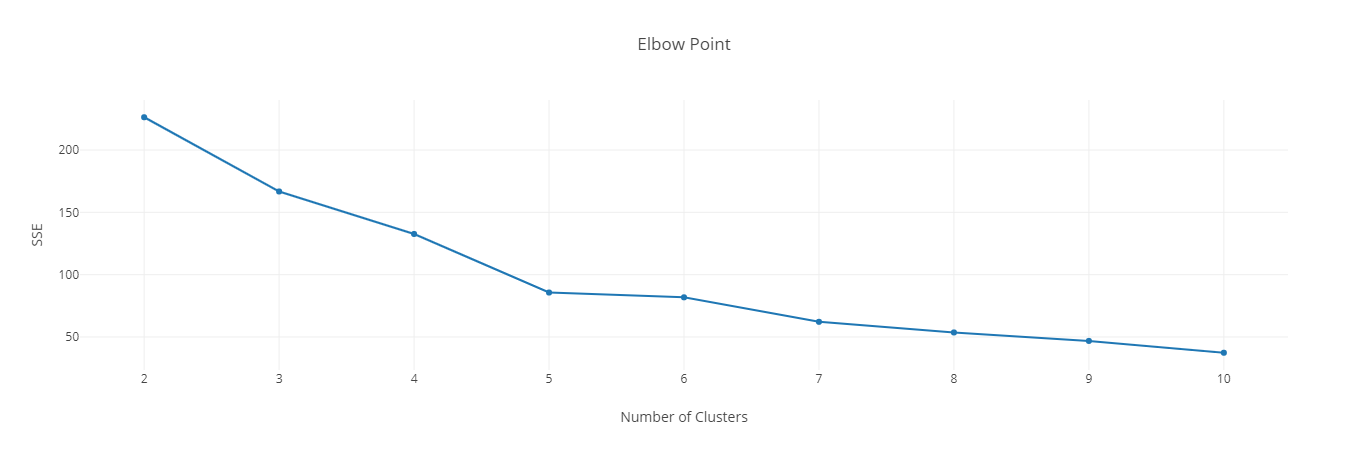
Il passo successivo si è concretizzato nel clustering.   
Le tipologie di clustering utilizzate in questo progetto sono state precisamente due, ossia il **k-means** e il **DBScan**, di seguito riportate:

**3.1 DBScan**

  
  
L’utilizzo del DBScan per il nostro progetto si è rivelato non essere una soluzione ottimale , come mostrato anche nel successivo grafico, in quanto il numero di elementi esclusi dall’analisi (**noise**) risulta essere troppo elevato o, in modo contrario, si vengono a creare pochi cluster.  


* 1. **K-means**

Utilizzando un approccio k-means è sorto un nuovo problema, ovvero decidere il valore di k-ottimale per effettuare il clustering. Come strumento di supporto per effettuare la scelta in modo giusto e corretto, abbiamo deciso di avvalerci dell’**Elbow method**.



Individuato il gomito all’interno del grafico, abbiamo di conseguenza individuato il k-ottimale per il nostro problema. Di seguito una funzione sviluppata per l’individuazione del k ottimale in modo automatico:



A questo punto siamo stati in gradi di generare il grafico ottimizzato e rappresentativo del nostro data-set diviso in k-cluster.

Immagine che contiene testo

Descrizione generata automaticamente

Il grafico ottenuto ad una prima analisi sembrava andare bene e sembrava essere una soluzione accettabile per il nostro problema, ma, andando ad analizzare i vari generi che rientravano in ogni k-cluster, abbiamo constatato la non esistenza di un genere prevalente sugli altri e ci siamo quindi resi conto dell’ambiguità delle canzoni presenti in ogni cluster.

Immagine che contiene testo

Descrizione generata automaticamente

Poiché né il DBScan né il k-means hanno portato ad una soluzione ottimale, siamo giunti alla conclusione che non è possibile stabilire il genere di una canzone basandosi solo sui suoi features.  
Dato che la nostra esigenza è comunque presentare qualcosa di funzionale, abbiamo rivisitato quella che era l’idea di base del progetto, usando le stesse metodologie applicate in precedenza.

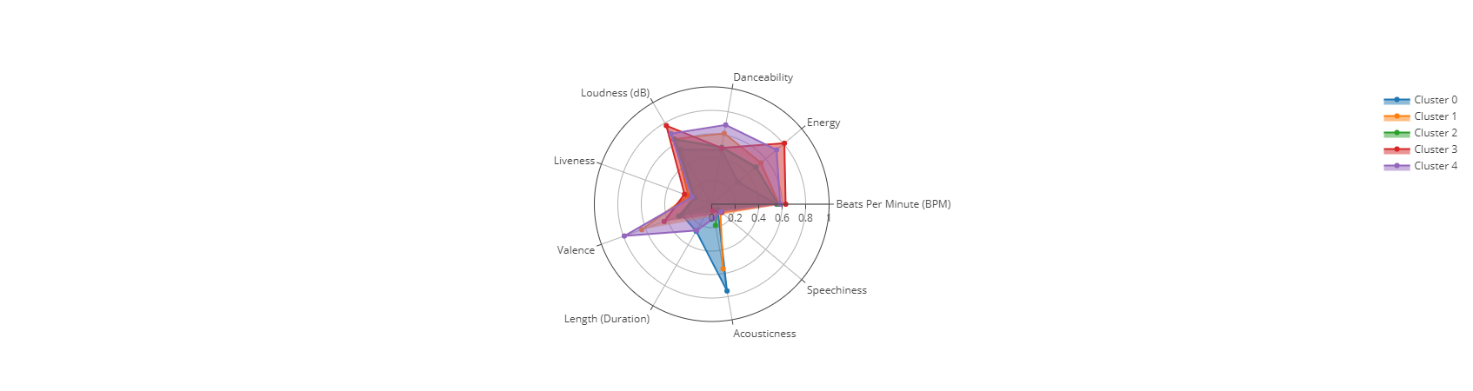
1. **Nuovo obiettivo**

Con l’aumento, negli ultimi dieci anni, delle diversità di musica presenti sulle svariate piattaforme di streaming come Spotify, Apple Music, Soundcloud, la linea che separa i vari generi musicali è diventata sempre più sottile di quanto non fosse in precedenza. Come si può essere in grado di identificare il genere rock se al suo interno possiamo trovare brani con caratteristiche e musicalità completamente diverse? Come si può stabilire se una persona proclamatasi fan del pop si riferisca al pop di Michael Jackson o a quello di Justin Bieber? Identificare i gusti o le canzoni in base ad un genere è diventato quindi praticamente impossibile, motivo del fallimento del precedente progetto.

In seguito a questa riflessione la nostra scelta è stata quella di riconoscere e raggruppare le canzoni solo in base alle loro caratteristiche non cercando necessariamente di trovare un’identificazione nel genere. Abbiamo quindi deciso di sfruttare la funzione di Spotify che permette di creare una libreria personale nella quale inserire i brani preferiti e di estrarre da quest’ultima i brani e raggrupparli , in base alle loro similitudini, in playlist create appositamente.

1. **Nuovo approccio**

Sfruttando i vari metodi utilizzati in precedenza, PCA e k-means, abbiamo notato un raggruppamento diverso dei brani ed abbiamo deciso di visualizzare i vari cluster non più su un piano tridimensionale ma un **grafico radar** per avere una lettura più immediata e precisa dei valori dei singoli cluster. Il grafico è di seguito mostrato:



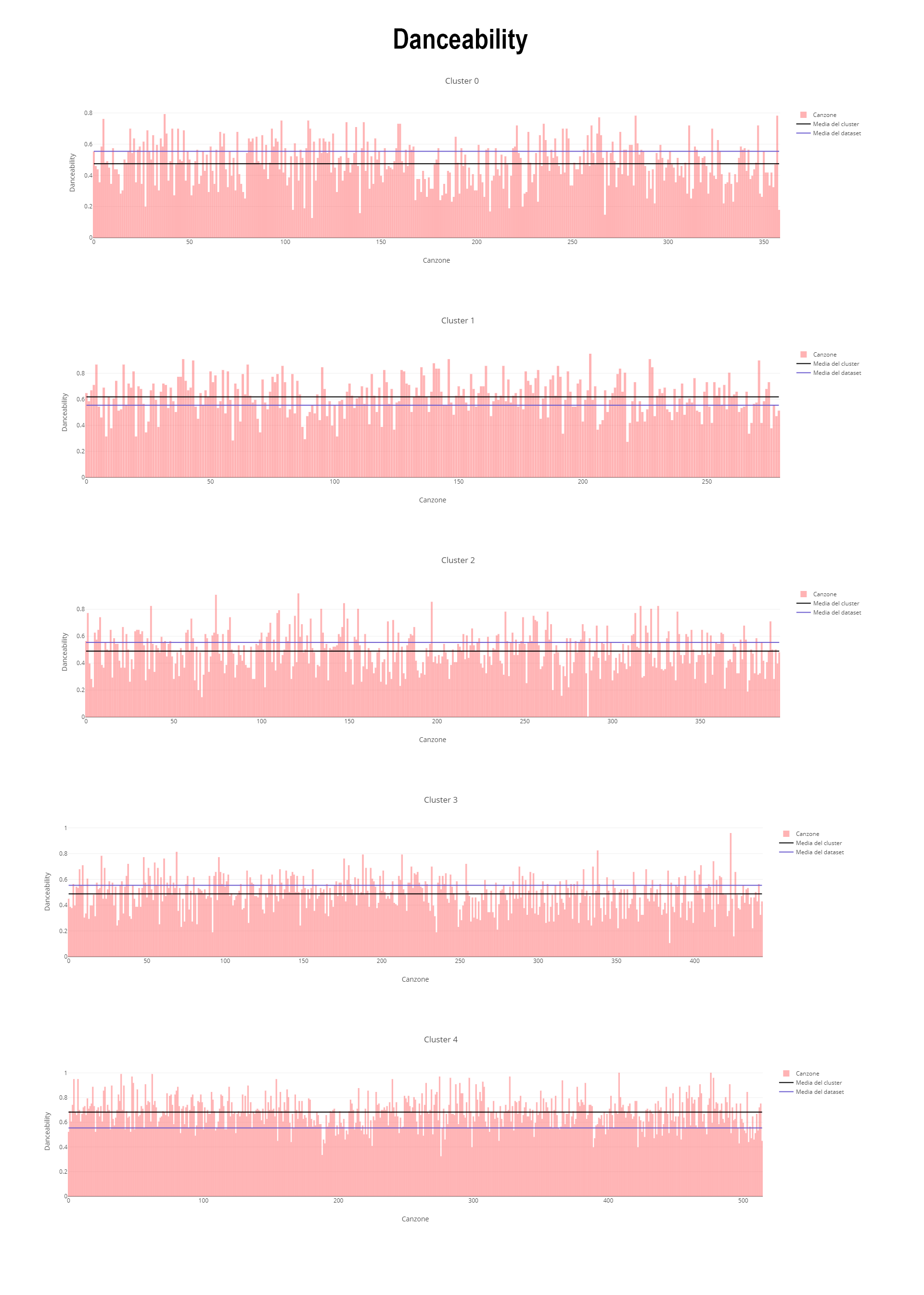
Prendendo come campione il cluster 0 possiamo notare che i valori di Acousticness sono molto maggiori rispetto agli altri cluster, ed infatti al suo interno abbiamo ritrovato canzoni caratterizzate dalla presenta di strumenti il cui suono non è distorto da effetti elettronici.

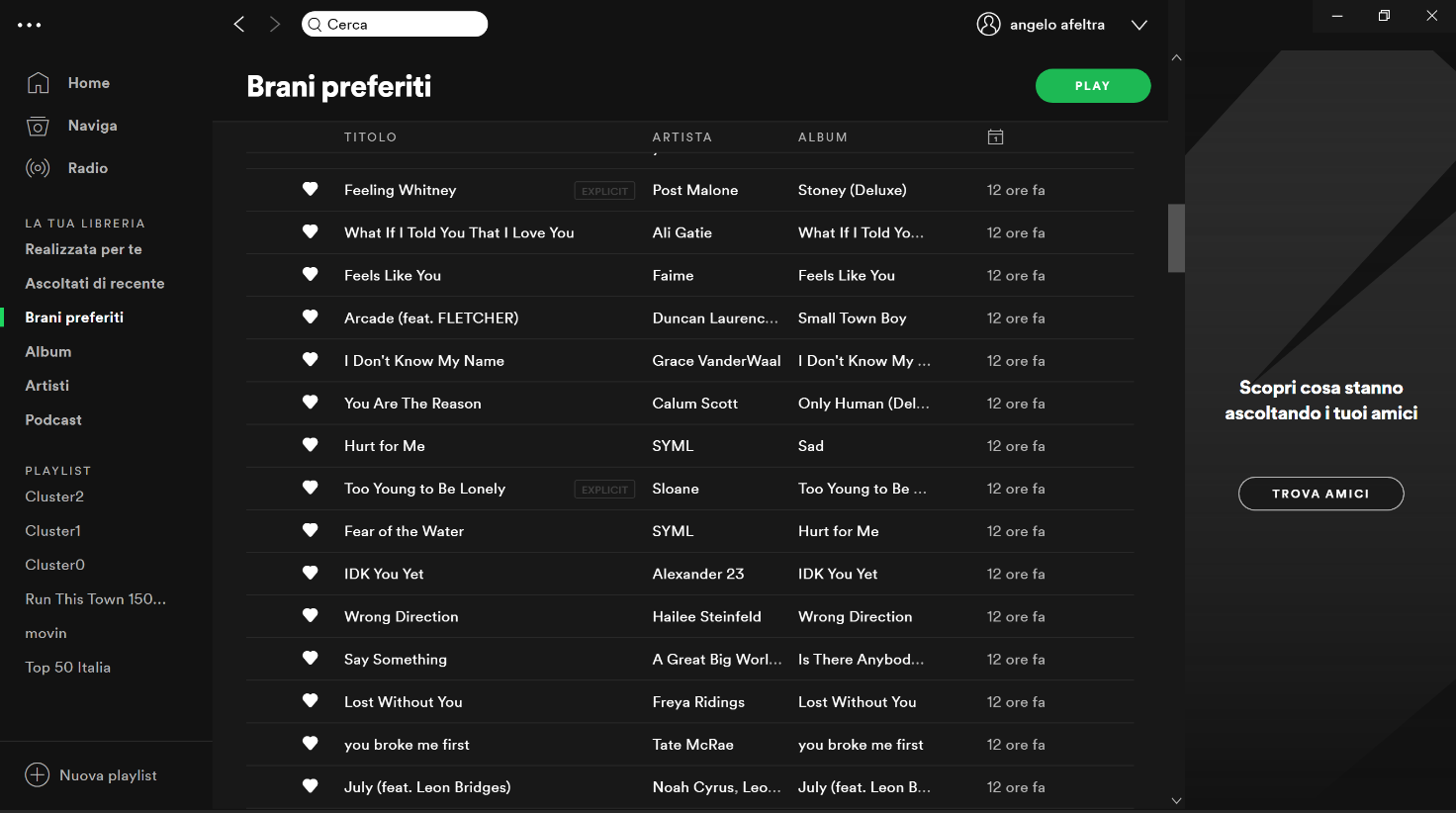
Si può notare inoltre la presenza di bassi valori di Speechness, trovando riscontro nell’assenza di canzoni rap o trap nel dataset usato, cosi come la presenza di bassi valori di Liveness, logicamente derivante dalla poca presenza su Spotify di canzoni dal vivo rispetto a quelle registrate in studio.

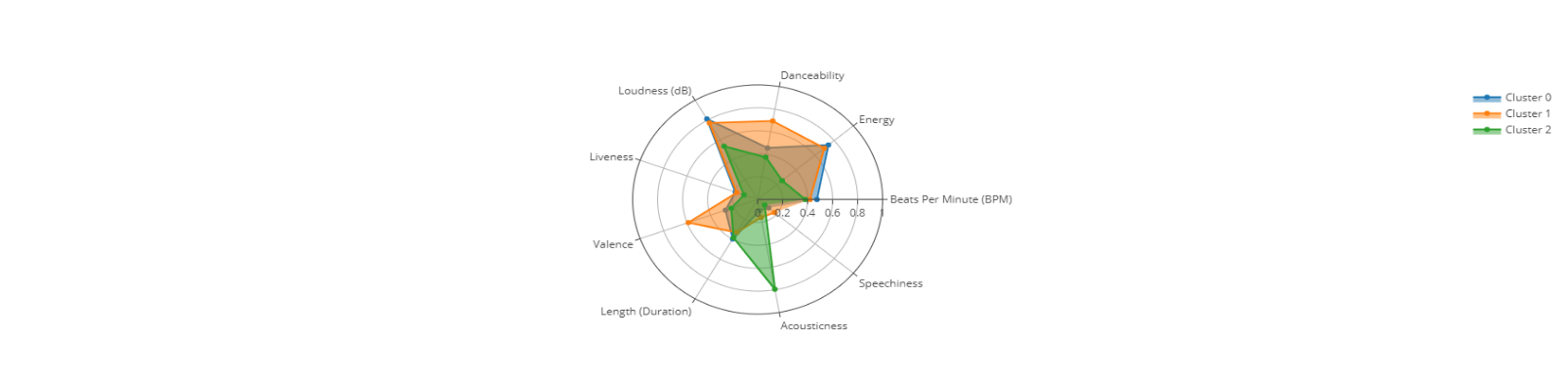
Tutto ciò trova riscontro e realizza i risultati che ci aspettavamo di ricevere oltre a raggruppare in modo appropriato brani etichettati anche con generi diversi.

Abbiamo infine rappresentato i valori delle features dei cluster in **bar charts** in modo da avere un ulteriore visione 2D dell’andamento dei valori delle features in ogni singolo cluster. Di seguito ne riportiamo alcuni:

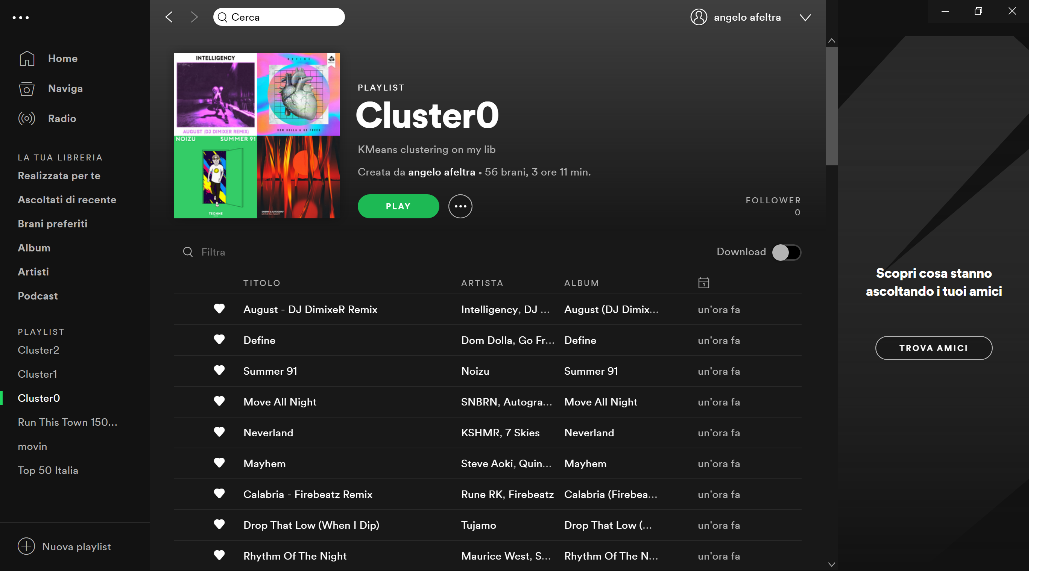
Immagine che contiene testo, antenna, dispositivo

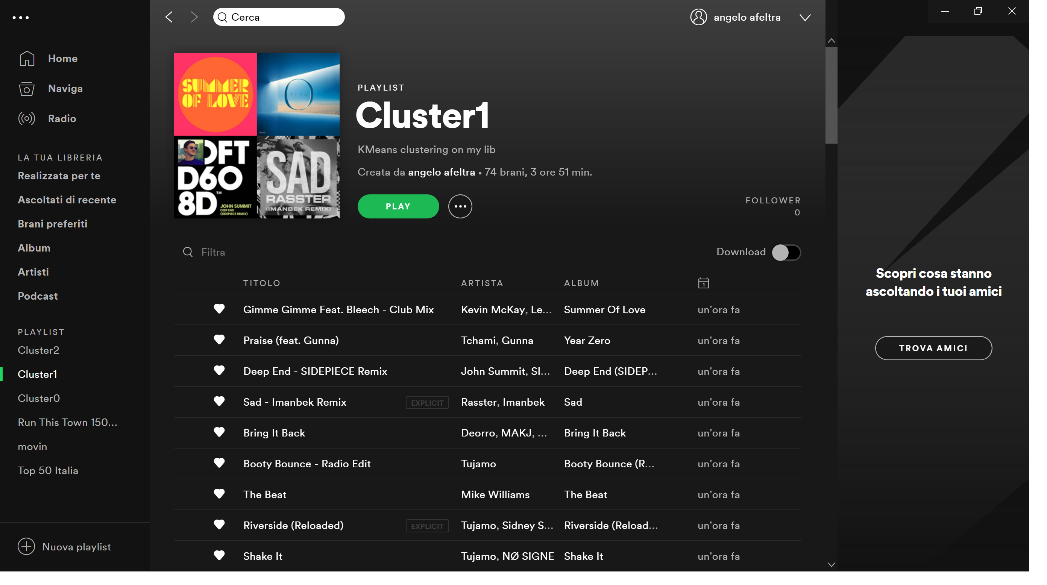
Descrizione generata automaticamente

1. **Esempio di interazione con l’utente**
   1. **Libreria personale utilizzata (194 brani)**
   2. **Radar ottenuto dal Clustering (k ottimale 3)**

****

* 1. **Playlist ottenute**

****

****

**Considerazioni**

Ai fini del nostro progetto possiamo quindi affermare che il secondo approccio rappresenta una soluzione ottimale ed alternativa al raggruppamento in generi, ormai datato. Le canzoni contenute nelle nuove playlist sono raggruppate con una precisione accettabile, che non va ad inficiare il raggruppamento stesso, dando luogo ad un’ottima interazione con l’utente.

Per ultimo, ma non ultimo, tutto ciò è servito ancora di più a farci rendere conto di quando sia importante analizzare e studiare situazioni, chiare solo apparentemente, sotto diversi punti di vista portando, anche matematicamente, a traguardi diversi.

Ci riteniamo quindi soddisfatti del risultato ottenuto e del percorso fatto.