|  |  |
| --- | --- |
| Università degli studi di Salerno – FonDamenti di intelligenza Artificiale (2020/2021) | AUTORI  Angelo Afeltra  Antonio Cacciapuoti  Antonio Cirillo  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. **Idea del progetto**

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

L’idea base era quella di sviluppare una web-app che permettesse ad un utente di poter selezionare un genere musicale ed ottenere, in base al genere scelto, un elenco di canzoni da poter ascoltare.

1. **Obiettivi del progetto**

In prima analisi sono state scelte le tecnologie da utilizzare per la progettazione della nostra IA ed i dati con i quali quest’ultima si sarebbe interfacciata.

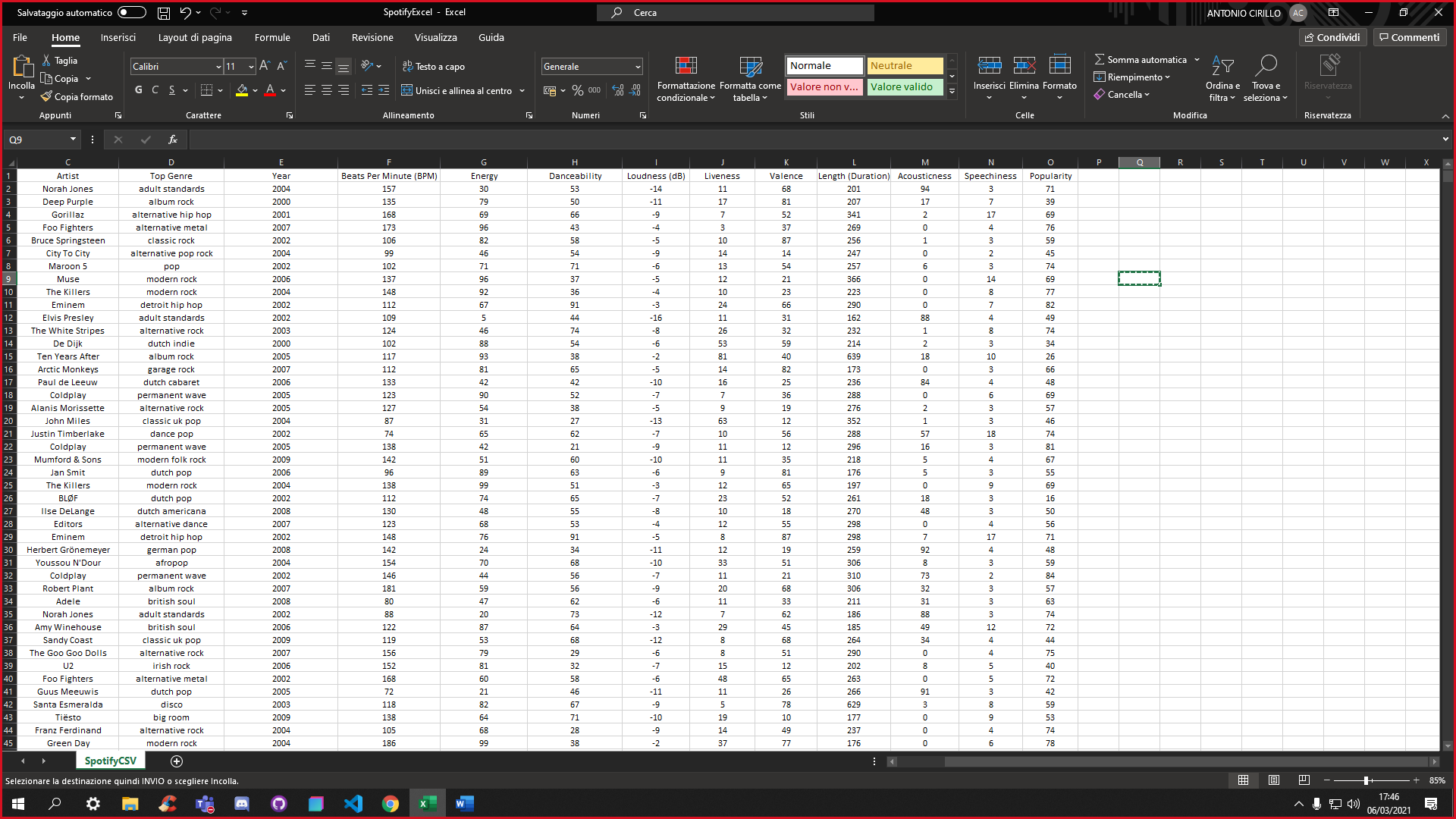
Come già espresso nella sezione precedente l’obiettivo primario del progetto è di selezionare, dalla piattaforma di streaming Spotify, un elenco di canzoni in base al loro genere. Si è deciso quindi di utilizzare un data-set rappresentativo dell’ambiente ed estratto proprio da esso.

Il secondo obiettivo è quello di 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 nostro approccio al progetto consta di vari step:

* **Scelta del dataset**: come già descritto nella sezione due il dataset scelto dovrà essere rappresentativo dell’ambiente Spotify. In seguito ad una ricerca su Kaggle abbiamo optato per un dataset estratto da Spotify cosi strutturato:



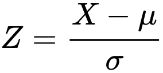
Le features per ogni canzone sono, rispettivamente:  
- **Beats Per Minute (BPM)**: unità di misura del numero di battiti per minuto;  
- **Energy**: unità di misura che indica l’intensità della canzone e l’energia trasmessa da quest’ultima;  
- **Danceability**: indica quanto la traccia è adatta ad essere ballata o meno;  
- **Loudness** **(dB)**: unità di misura dell’intensità acustica della traccia;  
- **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.

* **Standardizzazione;**
* **PCA;**
* **Clustering (K-means, DBScan);**

1. **Standardizzazione**

Il problema da affrontare dopo la scelta del dataset è stata **la standardizzazione** dei dati. Questo passaggio si è reso necessario in quanto, volendo utilizzare la metodologia PCA, andremo a calcolare nuove proiezioni del dataset. I nuovi assi ottenuti saranno logicamente basati sulla deviazione standard delle variabili, quindi una variabile con una deviazione standard maggiore avrà un impatto anch’esso maggiore sugli assi rispetto ad una variabile con deviazione standard più bassa. Andando a standardizzare i nostri dati, tutte le variabili avranno la stessa deviazione standard e di conseguenza lo stesso peso sui calcoli derivati dalla PCA.

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



Di seguito riportiamo gli snippet di codice relativi:



1. **PCA**

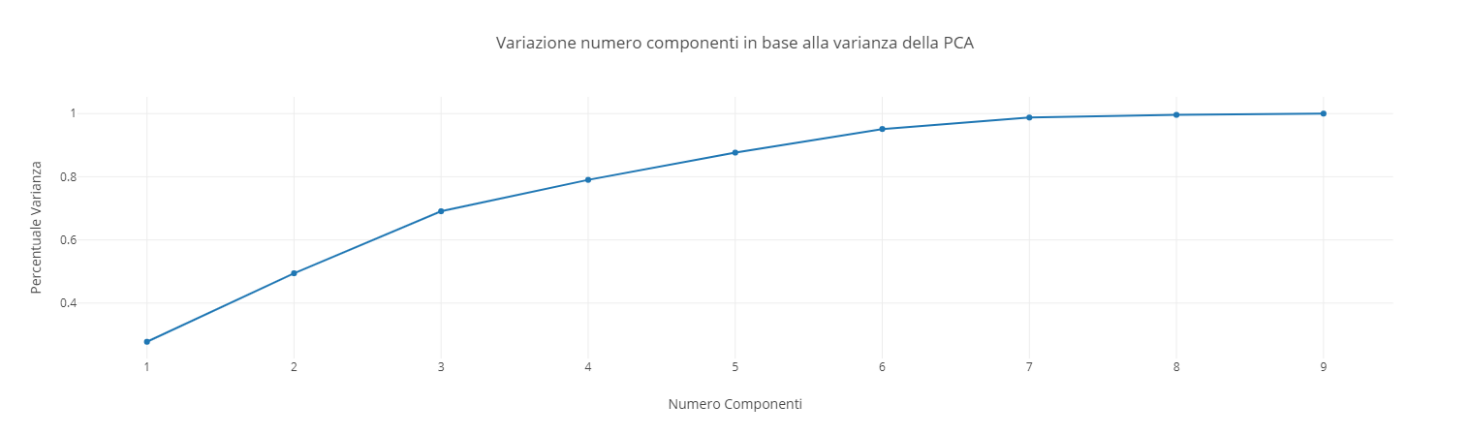
Dopo aver standardizzato i dati del nostro data-set, ci siamo trovati di fronte ad una scelta: ridurre le features da 8 a 3 avvalsi della **PCA** (principal component analysis), più precisamente della libreria **pca-js**, per varie motivazioni:

* Diminuzione computational cost: con la diminuzione delle dimensioni diminuisce la complessità del calcolo della distanza tra i punti del dataset. Riduciamo di conseguenza il tempo di calcolo e le risorse necessarie per esso;
* Possibilità di graficare i dati nello spazio usando le prime tre componenti;
* Riduzione del rumore (**noise**): l’algoritmo K-means, senza suggerimenti iniziali sui centroidi, prende come centroidi iniziali quelli più isolati. Utilizzando la PCA in modo preliminare neutralizziamo i valori anomali che giacciono sulle componenti minori, proiettandole nelle componenti maggiori derivate dalla PCA.

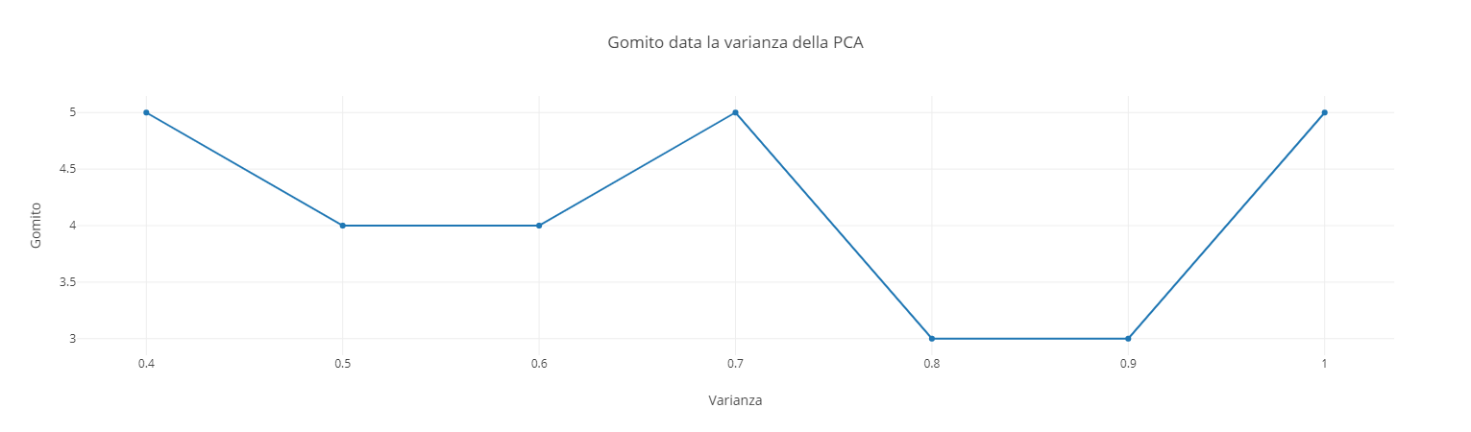
La PCA presente in tale libreria è basata su una risoluzione matriciale. La prima fase della suddetta consiste nel calcolo degli autovettori e autovalori:

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

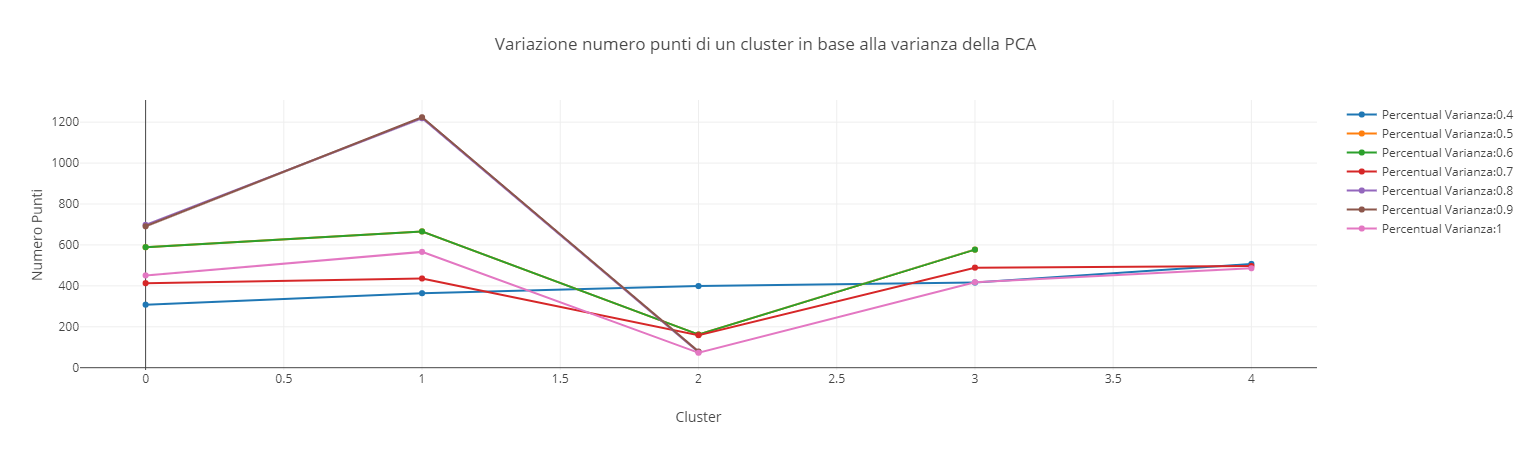
Successivamente al calcolo degli autovettori si è presentata la necessità di scegliere quali e quanti usarne per ottenere una **precisione accettabile**.



Il primo grafico mostra la percentuale di varianza rispetto al dataset originale, al variare del numero di componenti scelti.



Il secondo grafico mostra il k ottimale (calcolato tramite Elbow Method) da utilizzare al variare della soglia di varianza scelta nella PCA per il clustering k-means.



Il terzo ed ultimo grafico mostra il variare del numero di punti all’interno di ogni singolo cluster in base alla varianza scelta dalla PCA.

Analizzando i grafici generati abbiamo considerato una precisione del **70%** come ottimale per il nostro progetto, utilizzando quindi la combinazione di vettori che ci garantisce 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])**

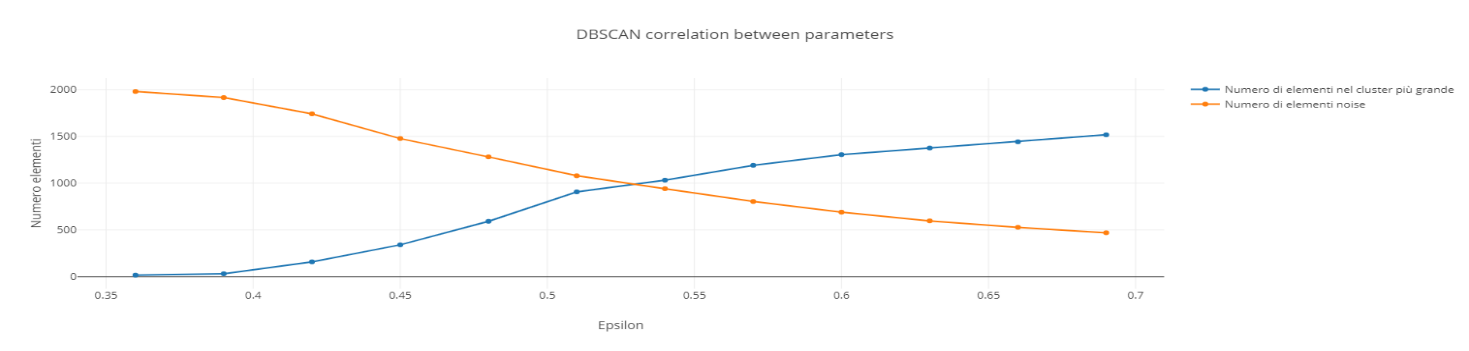
siamo stati in grado di ridurre il numero delle nostre features da n ad m, con m<n.

1. **Clustering**

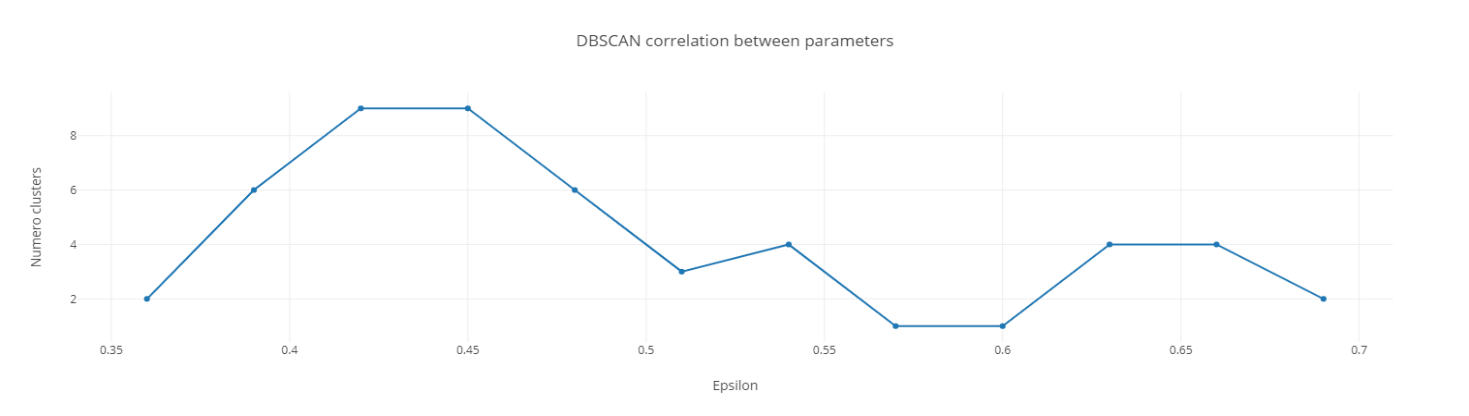
Il passo successivo si è concretizzato nel clustering.   
Gli algoritmi di clustering utilizzati in questo progetto sono state precisamente due, ossia il **DBScan** ed il **K-means**, di seguito riportate:

* 1. **DBScan**

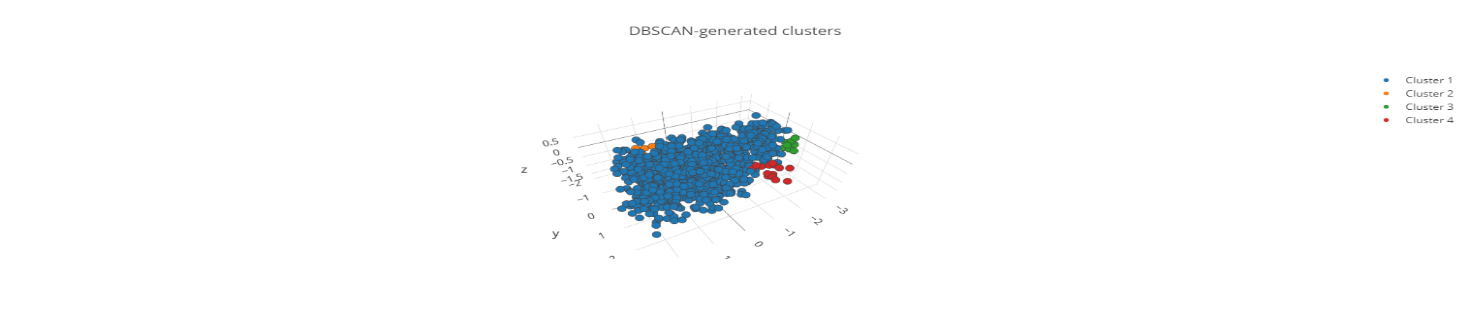
L’utilizzo del DBScan per il nostro progetto si è rivelato non essere una soluzione ottimale.

Come mostrato anche nel successivo grafico, infatti, il numero di elementi esclusi dall’analisi (**noise**) risulta essere troppo elevato o, in modo contrario, si vengono a creare pochi cluster ciascuno raggruppante un numero elevato di elementi risultando quindi poco efficace.

Dall’analisi del grafico precedente è stato scelto come buon compromesso per il valore di ε (Epsilon) 0.54. In questo modo avremo quattro cluster, il più grande dei quali con 1032 elementi e un noise con 941 elementi, non considerato accettabile.

****

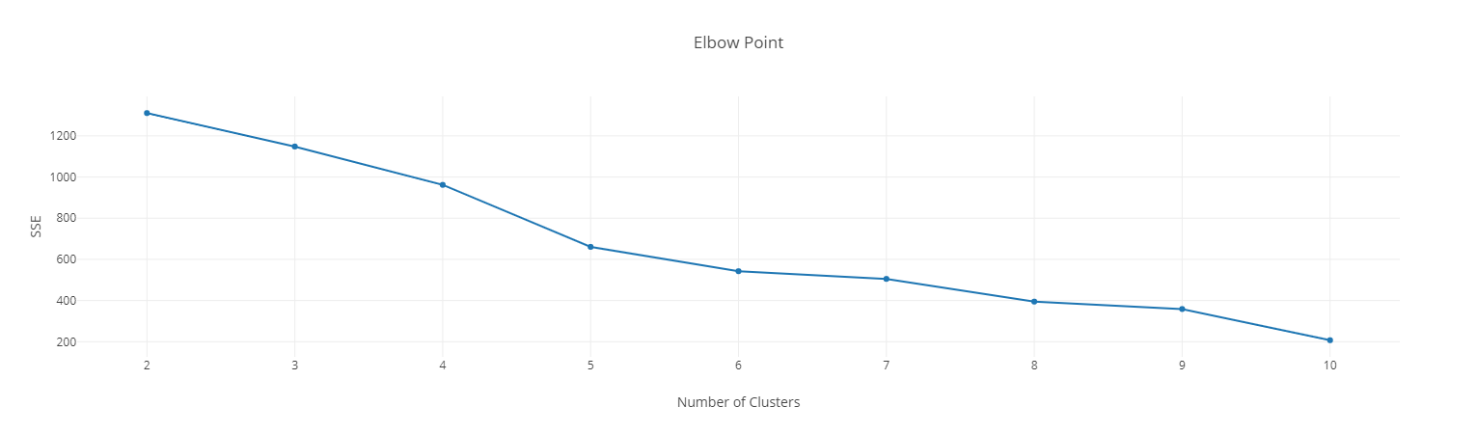
Il grafico di cui sopra mostra la correlazione tra ε e il numero di cluster che l’algoritmo genera.



Nell’ultimo grafico mostriamo in uno spazio 3d il risultato del DBScan con ε = 0.54 . Possiamo notare l’esistenza di un cluster predominante (colore blu) il quale fa supporre un risultato del processo di clustering non ottimale.

* 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**.

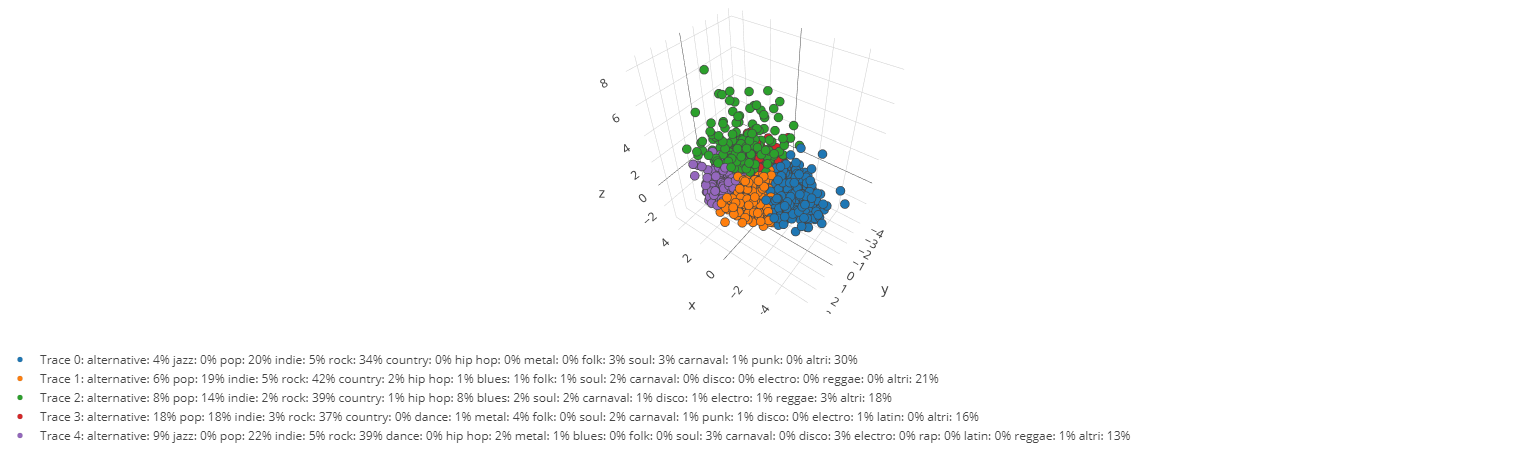


Individuato il gomito all’interno del grafico, abbiamo di conseguenza individuato il k-ottimale per il nostro problema. Di seguito una funzione sviluppata per l’individuazione del k ottimale in modo automatico:



L’Elbow point è individuato confrontando la variazione di SSE tra (K-1, K) e (K,K+1) calcolata per ogni K compreso tra 2 e 10. Il k per il quale questa variazione risulta essere maggiore è l’Elbow point restituito.

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



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 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.

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 sulle sue 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 ma con nuovo obiettivo.

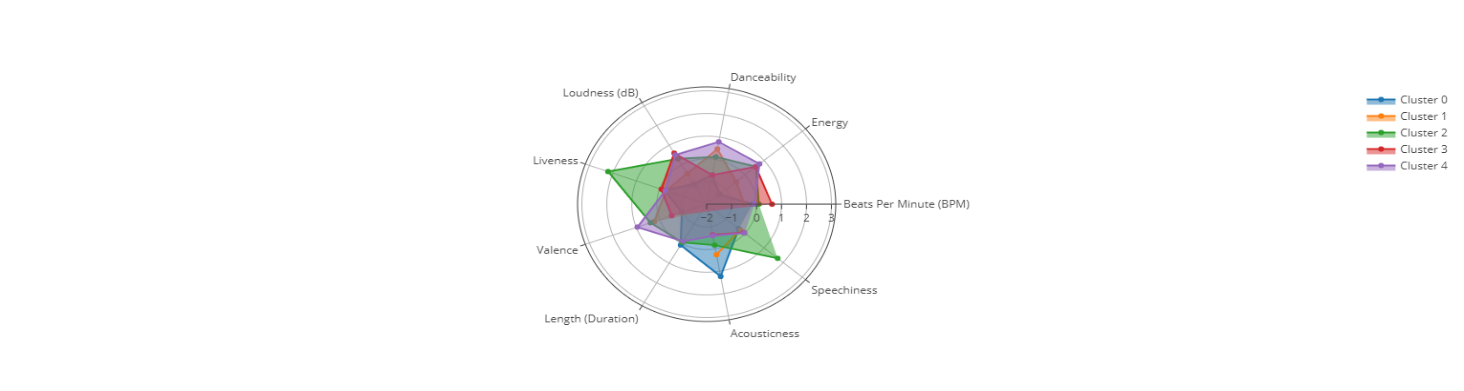
1. **Nuovo obiettivo**

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 su un **grafico radar** per avere una lettura più immediata e precisa dei valori medi delle features per ogni singolo 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.

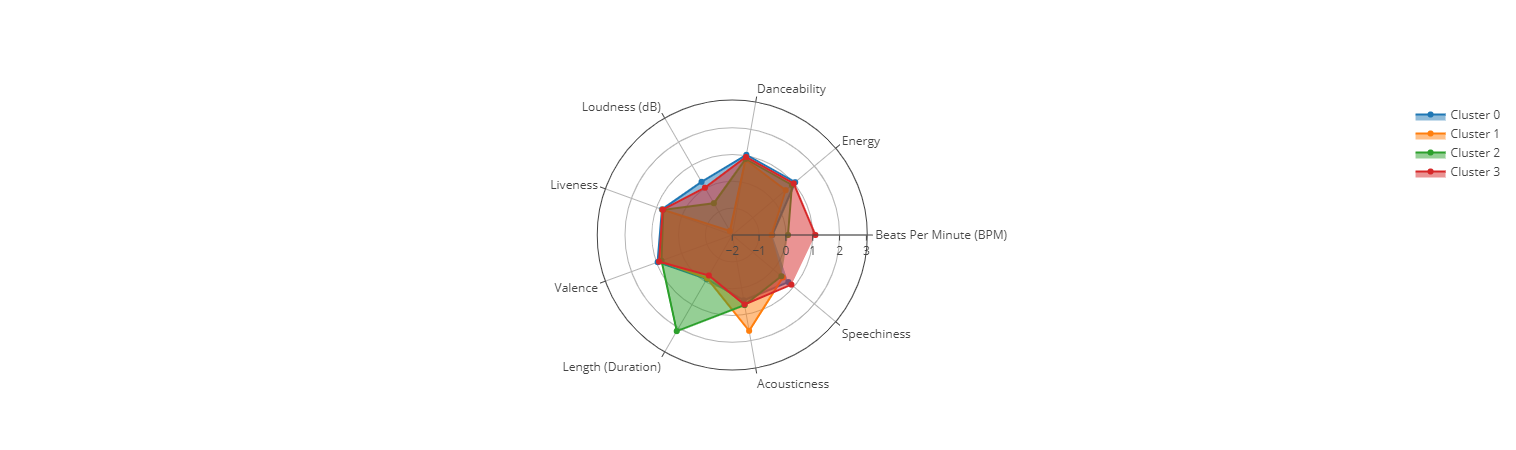
Ancora, prendendo come campione il cluster 3 si può notare l’alto valore di Speechiness con una conseguente presenza di canzoni molto parlate.

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:



1. **Esempio di interazione con l’utente**
   1. **Libreria personale utilizzata (1092 brani)**
   2. **Radar ottenuto dal Clustering (k ottimale 4)**

****

* 1. **Playlist ottenute**

****

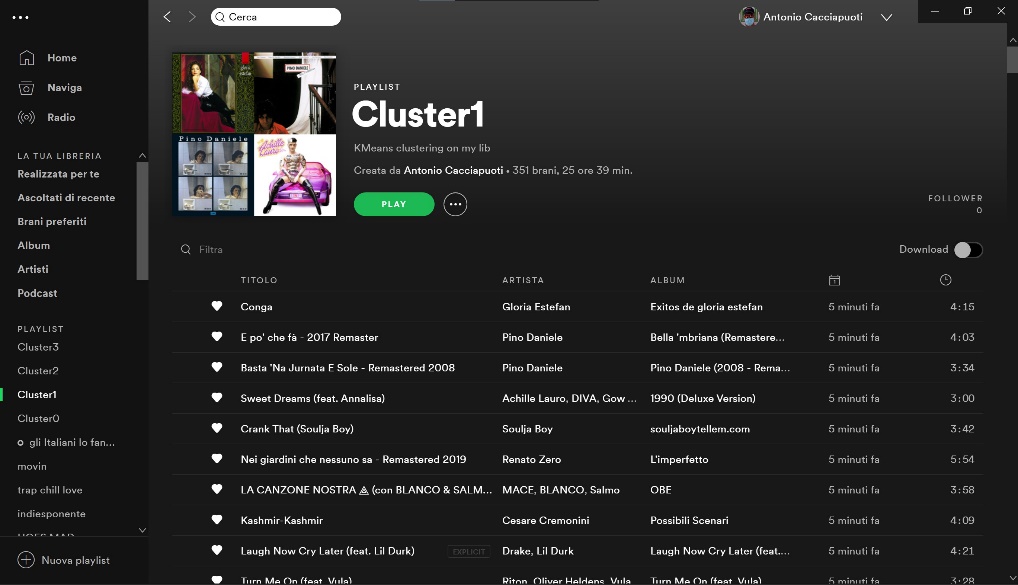
****

Immagine che contiene testo

Descrizione generata automaticamente

1. **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.