|  |  |
| --- | --- |
| 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 scoprire se è possibile identificare un genere musicale a partire da alcune caratteristiche comuni delle canzoni riconducibili al genere stesso. Per far questo abbiamo deciso di ricorrere all’utilizzo di un metodo di apprendimento non supervisionato quale il **clustering**. Dando quindi in pasto a diversi algoritmi di clustering e con vari parametri un insieme di canzoni (ognuna identificata, oltre che dal titolo e altre informazioni di base, da alcune features che la descrivessero musicalmente) vogliamo ottenere dei gruppi all’interno dei quali siano presenti in maggior percentuale canzoni riconducibili allo stesso genere.

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 partizionare un elenco di canzoni in base al loro genere. Abbiamo deciso di utilizzare come piattaforma musicale con la quale relazionarci Spotify, che attualmente risulta essere ben affermata ma allo stesso tempo tra le più innovative nel settore. L’utilizzo di Spotify, essendo essa la piattaforma di streaming musicale più diffusa al mondo, ci ha permesso di poter scegliere tra vari numerosi dataset già generati e pronti all’uso.

Ci ha inoltre consentito di poter creare uno script che potesse trasferire i risultati della nostra ricerca alle nostre librerie personali grazie all’utilizzo delle loro Web API.

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 è stato strutturato in vari step:

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

1. **Scelta del dataset**

Il dataset scelto deve essere rappresentativo delle features in base alle quali Spotify descrive le canzoni. Per questo motivo, abbiamo optato per un dataset estratto da Spotify stesso. Abbiamo ricercato il nostro dataset sulla piattaforma Kaggle, e ne abbiamo trovati diversi che potevano fare al caso nostro. Ciascun dataset si differenziava, oltre che per il numero di campioni contenuti, per la loro varietà. Alcuni dataset comprendevano canzoni popolari negli ultimi 10 o 20 anni: abbiamo reputato questi dataset non ottimali poiché contenevano molte canzoni pop e non rappresentevano in maniera equa ogni macro genere musicale (rock, rap, classica, etc.).

Abbiamo poi analizzato un dataset distribuito molto equamente sui generi ma contenente circa 300k campioni. Una data mole, in combinazione con le tecnologie da noi utilizzate, ci ha portato a performance basse e difficoltà computazionali in molte fasi del progetto. Abbiamo quindi in ultima istanza optato per un dataset comunque ben rappresentativo di ogni genere ma contente circa 2000 canzoni rilasciate tra il 1956 e il 2019.

Oltre a informazioni basilari quali nome, artista e anno le features per ogni canzone sono, rispettivamente:

* **Energy**: unità di misura che indica l’intensità della canzone e l’energia trasmessa da quest’ultima;
* **Beats Per Minute (BPM)**: unità di misura del numero di battiti per minuto;
* **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;
* **Top Genre**: genere *supposto* associato alla canzone in base al genere più rappresentativo dell’artista.

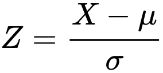
Immagine che contiene testo, computer, monitor, screenshot

Descrizione generata automaticamente

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, andiamo a generare un nuovo dataset ottenuto proiettando i campioni originali su un nuovo sistema di riferimento. I nuovi assi ottenuti sono basati sulla deviazione standard delle variabili, quindi una variabile con una deviazione standard maggiore avrà un impatto maggiore sugli assi rispetto ad una variabile con deviazione standard più bassa. Andando a standardizzare i nostri dati, tutte le variabili hanno 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 seguente formula matematica, sostituendo ad ogni valore X originariamente presente il valore Z pari a:



con μ pari alla media della variabile e σ pari alla sua deviazione standard.

Di seguito riportiamo gli snippet di codice relativi:



1. **PCA**

Dopo aver standardizzato i dati del nostro dataset, ci siamo ritrovati con un insieme di campioni ciascuno avente 10 features caratterizzanti. Abbiamo da subito deciso di non utilizzare la **popolarità** di una canzone, in quanto essa è calcolata da Spotify in funzione del numero di riproduzioni della canzone nel breve periodo. Poichè le canzoni del nostro dataset sono state rilasciate anche 70 anni fa, e poiché crediamo che una canzone rilasciata di recente faccia in media più ascolti rispetto a una canzone del passato, questa particolare misura non ci è sembrata abbastanza espressiva ed affidabile.

Inoltre abbiamo deciso di ignorare anche la feature **length:** questo perché è possibile notare che la durata di un brano è cambiata molto dal 1950 ad oggi (tendendo a ridursi) ed inoltre riteniamo che essa non sia indicativa di un genere.

Abbiamo poi deciso di utilizzare le rimanenti 8 features non singolarmente ma dopo un processo di Principal Component Analysis. La PCA è una tecnica che permette di ridurre il numero di variabili che descrivono un insieme di dati limitando il più possibile la perdita di informazioni. Nel nostro caso ciò equivale a descrivere una canzone non più per mezzo di tutte le 8 features originali ma attraverso una combinazione di esse, risultato della PCA.

La PCA presenta vari benefici:

* Diminuzione del 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 tridimensionale usando le prime tre componenti;
* Riduzione del rumore (**noise**): utilizzando la PCA in modo preliminare neutralizziamo i valori anomali che giacciono sulle componenti minori, proiettandole nelle componenti maggiori derivate dalla PCA.

In relazione alla parte implementativa del nostro progetto, ci siamo avvalsi 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**. Infatti, scegliendo un numero di componenti principali da utilizzare minore del numero originale di features, si ha inevitabilmente una perdita d’informazione derivante dalla perdita di varianza causata dal non utilizzo di tutte le componenti. Siamo quindi di fronte ad un compromesso: quante componenti utilizzare? Quale può essere una percentuale di “errore” (perdita di varianza) accettabile?

Inizialmente abbiamo pensato di utilizzare solo le prime tre componenti principali. In questo modo avremmo dato particolare senso all’utilizzo della PCA: ci avrebbe permesso di trovare le prime tre componenti più significative e avremmo utilizzato queste prime tre componenti per la clusterizzazione ma soprattuto per la rappresentazione grafica nello spazio dei risultati della clusterizzazione.

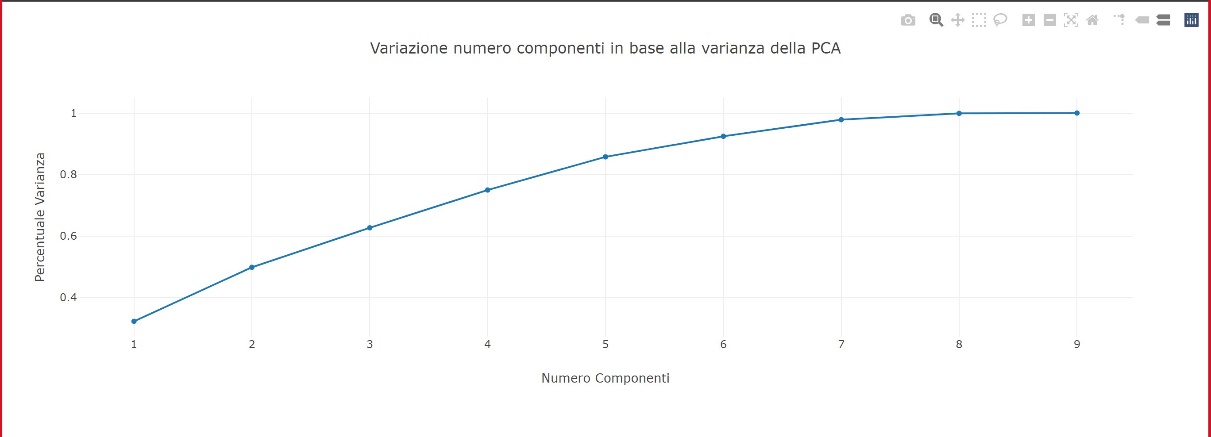
Ci siamo però resi conto che ciò non sarebbe stato sempre ottimale: nel corso dello sviluppo del progetto, provando anche con dataset estrapolati dalle nostre stesse librerie personali Spotify, le prime tre componenti hanno portato ad una perdita di varianza anche del 40%.

Per questo motivo, abbiamo deciso di procedere nel seguente modo:

dopo aver effettuato la PCA l’algoritmo di clusterizzazione avrebbe lavorato sul numero di componenti sufficiente a raggiungere almeno il 70% di varianza originale;

i grafici che mostrano visivamente il risultato della clusterizzazione sarebbero invece stati proiettati in uno spazio tridimensionale in cui le tre dimensioni sarebbero state sempre le prime tre componenti principali, in quanto esse sono ad ogni modo le componenti sulle quali più informazione è contenuta e sono quindi più significative.

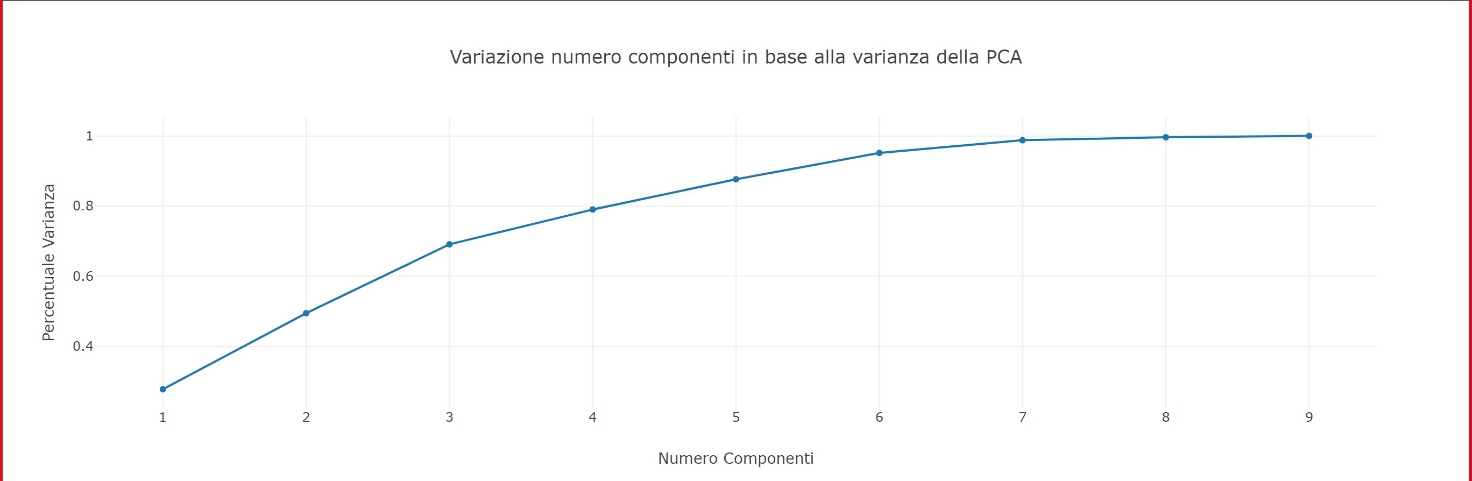
Il grafico di seguito si riferisce al dataset preso da Kaggle e che abbiamo fin’ora descritto. Esso mostra la percentuale di varianza dopo la PCA rispetto al dataset originale, al variare del numero di componenti scelti.



E’ da evidenziare quindi che nel caso del dataset canonico utilizzato fin’ora il numero di PC necessarie per raggiungere il 70% della varianza originale è 4.

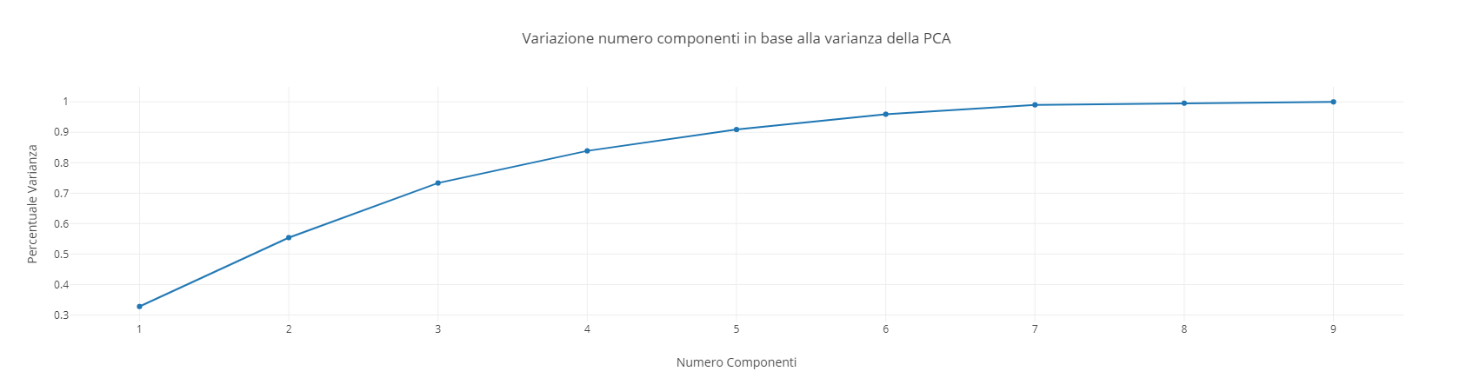
A dimostrazione del fatto che al variare del dataset il risultato di questa analisi varia, di seguito mostriamo lo stesso grafico numero componenti – varianza ma riferito al dataset estrapolato dalla libreria Spotify personale di alcuni di noi.

Ecco il grafico per la libreria personale di Antonio Cacciapuoti:

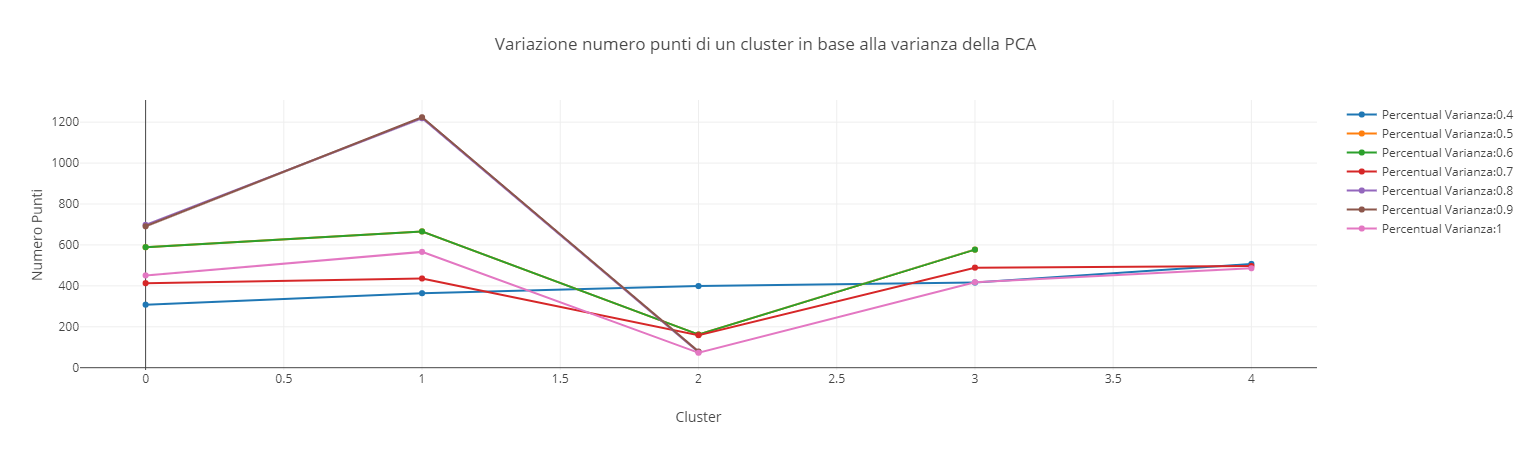


Il numero di componenti necessario a raggiungere il 70% di varianza originale su questo dataset è ancora 4.

Di seguito invece lo stesso grafico per la libreria personale di Angelo Afeltra:



Com’è possibile notare, in questo caso il numero di componenti necessarie a raggiungere il 70% di varianza originale è solo 3.



L’ultimo grafico mostra il variare del numero di punti all’interno di ogni singolo cluster in base alla varianza scelta dalla PCA, ancora sul dataset trovato su Kaggle.

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)**

(dove in vectors sono calcolate e conservate dinamicamente le componenti principali necessarie a raggiungere il 70% di varianza originale) 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 finalmente nel clustering.

Per clustering si intende un insieme di metodologie utilizzate per raggruppare oggetti in classi omogenee. Ogni cluster (classe) è un insieme di oggetti con caratteristiche simili agli altri oggetti dello stesso cluster ma che si differenziano più o meno notevolemente dagli oggetti presenti negli altri cluster. Il clustering rientra nel ramo dell’apprendimento non supervisionato.

Gli algoritmi di clustering utilizzati in questo progetto sono stati due, ossia il **DBScan** ed il **K-means**, ed i risultati del loro utilizzo sono riportati di seguito.

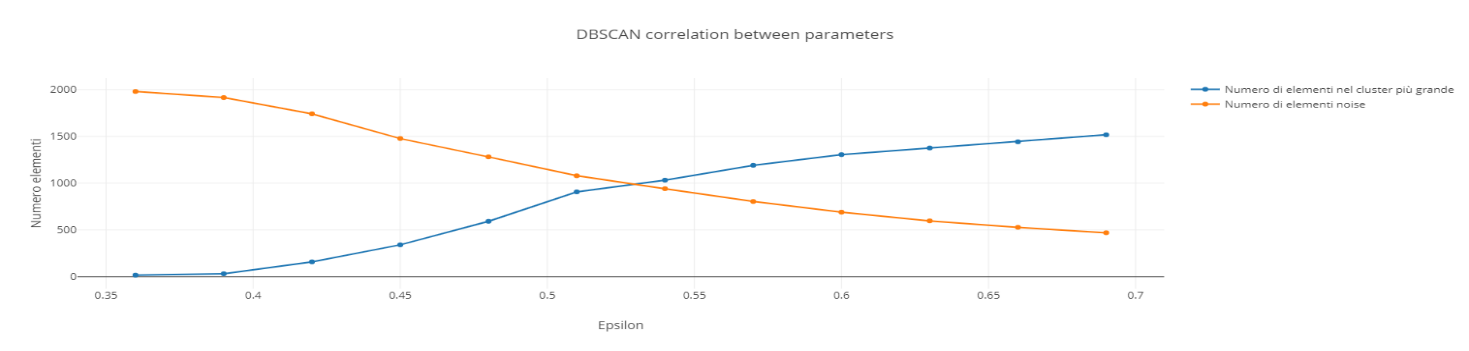
* 1. **DBScan**

Il DBScan è un algoritmo di clustering basato sulla densità: dato un set di punti in uno spazio, raggruppa insieme (e di conseguenza proietta nello stesso cluster) punti che sono sufficientemente vicini e segna come valori anomali quei punti che giacciono più o meno “da soli” in regioni poco dense (dove per ogni punto il suo prossimo vicino è comunque troppo lontano).

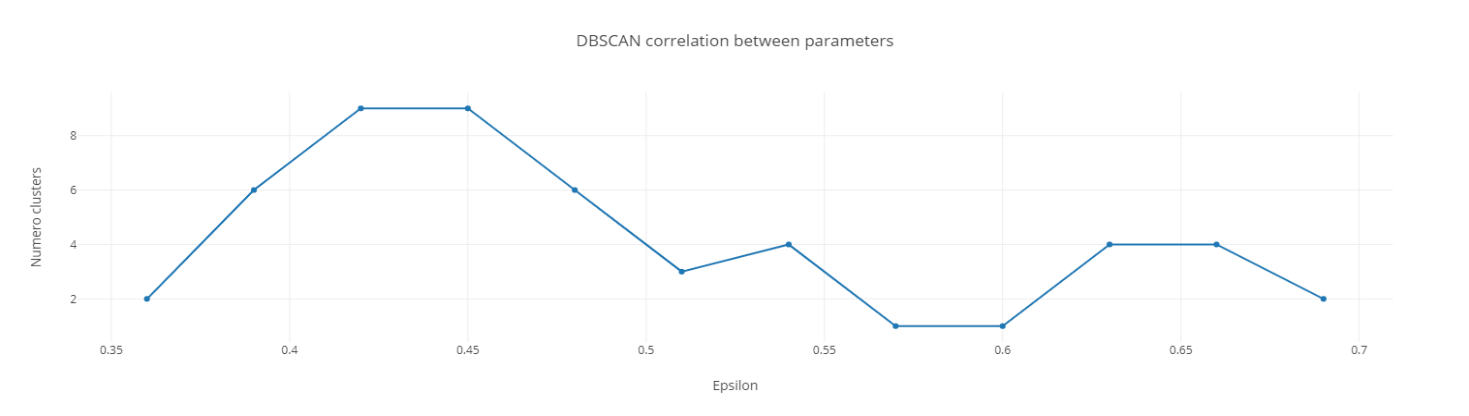
L’algoritmo ha due parametri:

* Epsilon: la massima distanza tra i punti. Consente di determinare se un punto fa parte di un cluster o meno;
* minPoints: il numero minimo di punti necessari per formare un cluster

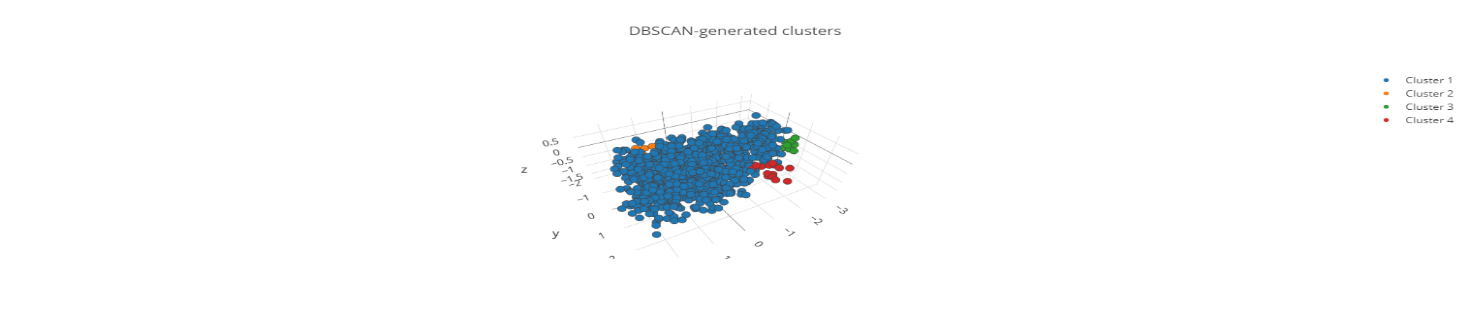
Nel nostro caso abbiamo provato ad eseguire l’algoritmo con il parametro minPoints fissato a 10 e con valori di epsilon che variavano da 0.35 a 0.7.

Come mostrato nel successivo grafico, il numero di elementi esclusi dall’analisi (**noise**) è risultato essere troppo elevato o, in modo contrario, erano creati pochi cluster ciascuno comprendente un numero elevato di elementi e 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 abbiamo avuto 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 ha generato.

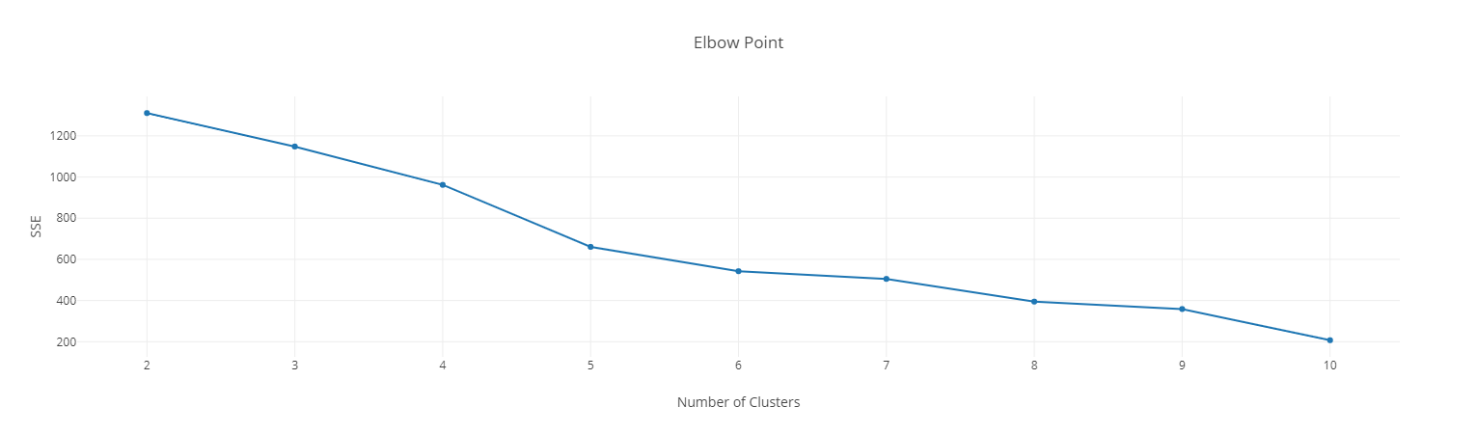


Nell’ultimo grafico mostriamo in uno spazio tridimensionale il risultato del DBScan con ε = 0.54 . Possiamo notare l’esistenza di un cluster predominante (colore blu) il quale fa supporre un risultato del processo di clustering non ottimale.

* 1. **K-means**

Il K-means è un algoritmo partizionale che permette di suddividere un insieme di punti in K gruppi. Ogni gruppo viene identificato mediante un **centroide**. L'algoritmo segue una procedura iterativa: inizialmente crea k partizioni e assegna i punti d'ingresso a ogni partizione casualmente, quindi calcola il centroide di ogni gruppo; costruisce in seguito una nuova partizione associando ogni punto d'ingresso al gruppo il cui centroide è più vicino ad esso; infine vengono ricalcolati i centroidi per i nuovi gruppi e così via finche l’algoritmo non converge o finchè il numero massimo di iterazioni impostato è raggiunto.

Nell’utilizzo dell’algoritmo 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**. Esso è un metodo che consiste nel graficare l’SSE (Somma degli errori quadrati) come funzione del numero di cluster e scegliere come K ottimale il gomito della curva risultante. Per SSE intendiamo la somma delle distanze di ogni punto dal proprio centroide per ogni cluster.

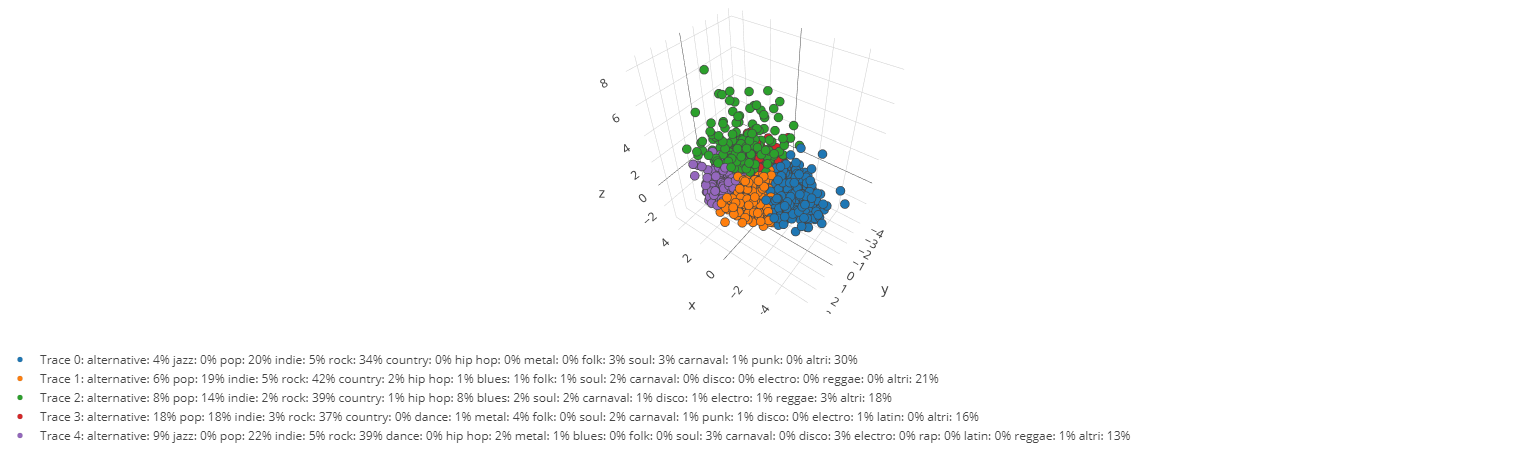


Individuato il gomito all’interno del grafico, abbiamo di conseguenza individuato il k-ottimale per il nostro problema. Di seguito una funzione sviluppata per l’individuazione del k ottimale in modo automatico:



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.

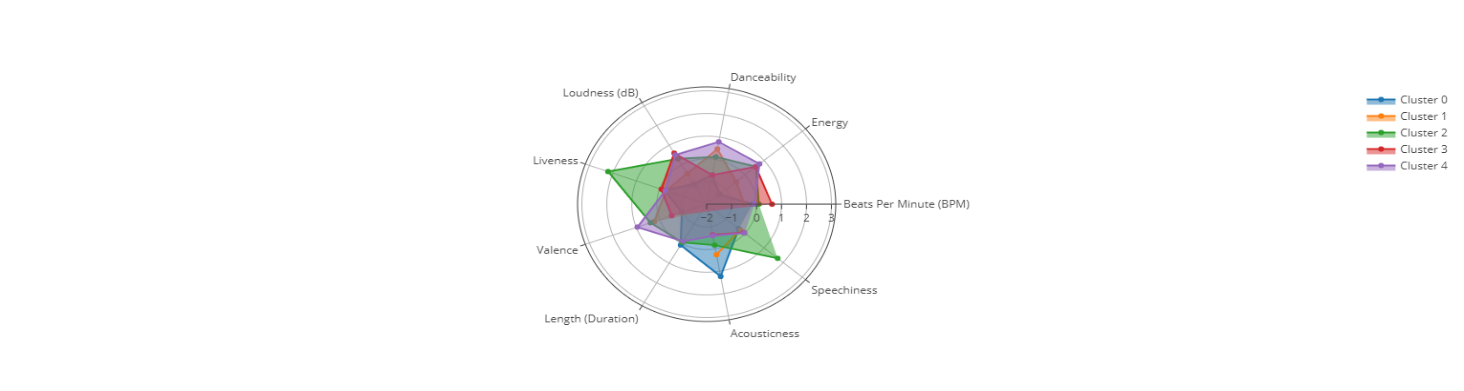
1. **Nuovo obiettivo**

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.

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.

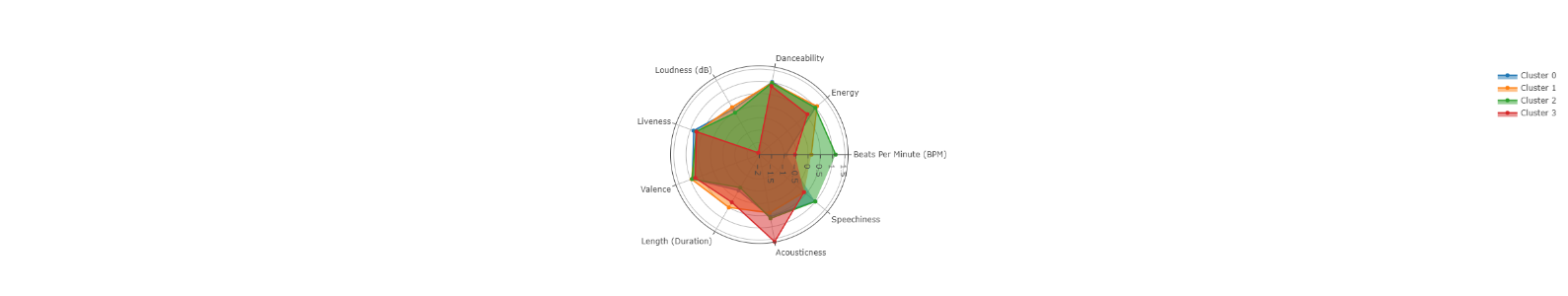
Abbiamo infine rappresentato i valori delle features dei cluster in **bar charts** in modo da avere un ulteriore visione 2D dell’andamento dei valori delle features in ogni singolo cluster. Di seguito ne riportiamo alcuni:

1. **Esempio di interazione con l’utente**

Dopo aver concluso la ricerca di metodi e parametri ottimali per raggiungere il nostro scopo di suddividere una libreria musicale di input in vari gruppi in funzione delle varie caratteristiche delle singole canzoni, abbiamo messo a punto uno script che permettesse di ottenere varie playlists a partire dai propri brani preferiti su Spotify.

Lo script genera un dataset (del tutto simile in features a quello utilizzato in precedenza preso da Kaggle), lo analizza, ne ricava vari cluster e genera poi per ogni cluster una playlist. Di seguito alcuni screenshot che ne dimostrano il funzionamento:

* 1. **Libreria personale utilizzata (1092 brani)**
  2. **Radar ottenuto dal Clustering (k ottimale trovato a 4)**

****

* 1. **Playlist ottenute**

****

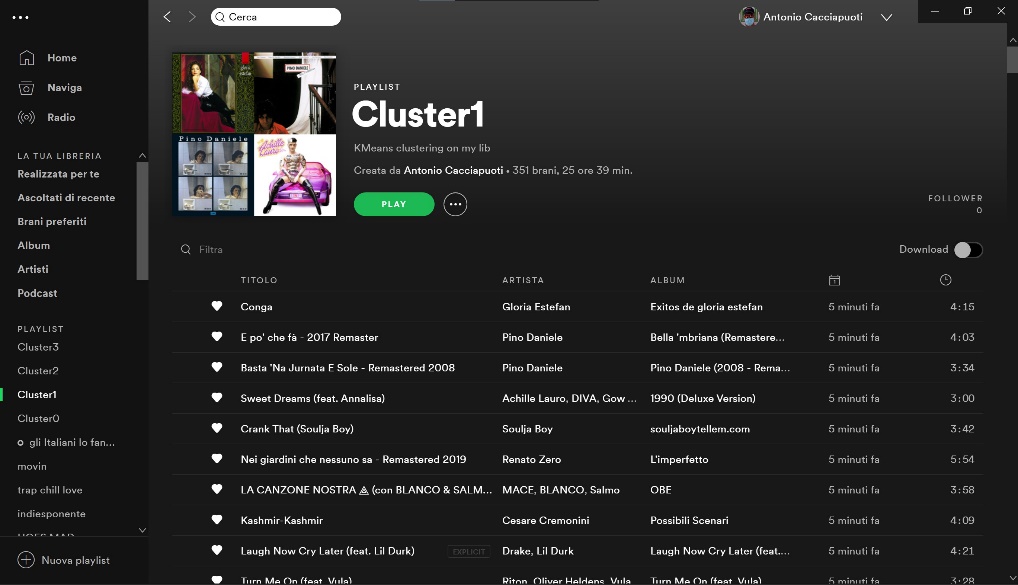
****



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 obsoleto. 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.