Fondamenti di Intelligenza Artificiale

Progetto

SpotifAI

Autore:
David Coccorullo

Matricola:
0512110452

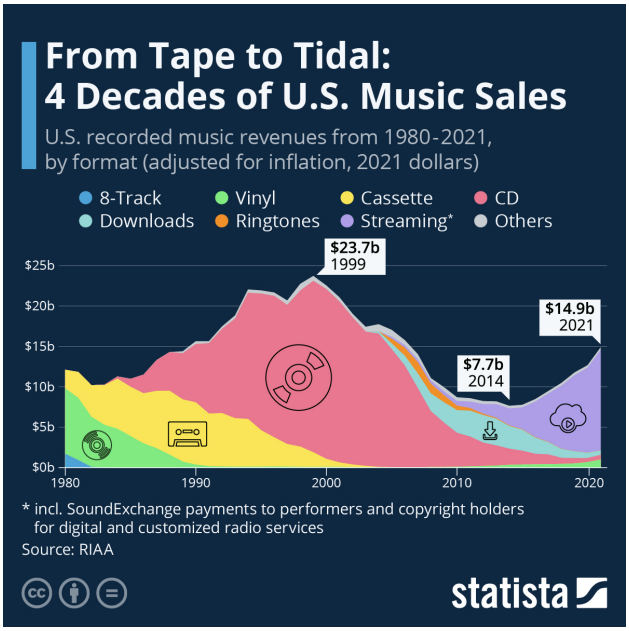
Repository:
https://github.com/davidcocc/SpotifAI

Indice

1 Presentazione del Problema

1.1 Il paradosso della scelta

Il modo di fruire della musica è radicalmente mutato nel corso degli anni: le radio e i supporti fisici (vinili, *tapes*, CD...), fino a pochi anni fa unico mezzo per ascoltare i propri brani preferiti, si stanno sempre più spegnendo, riducendosi a oggetti da collezionismo.
La "causa del decesso" è sicuramente da imputare al rapido espandersi dei servizi di *streaming* musicali che hanno preso piede nei dispositivi di tutti noi.



Ricavi dalla vendita di contenuti musicali nel tempo, divisi per formato.

Il vantaggio principale per il consumatore consiste nel fatto di non essere più "costretto" a spendere una certa cifra per l'acquisto di un singolo album, ma è in grado di pagare una quota mensile per l'accesso ad un intero catalogo grazie alla natura *on demand* di piattaforme come *Spotify*, *Apple Music*, *Deezer* etc.
La possibilità di accedere a una quantità di musica apparentemente infinita (e

in continua espansione!) è sicuramente un ottimo punto a favore dello streaming musicale, ma lascia spazio a una riflessione: quante volte, aprendo servizi di streaming di qualsiasi genere, ci troviamo quasi sopraffatti dall'enorme quantità di **scelta** che ci viene offerta?
Il continuo rilascio di musica sul mercato, e l'immediato accesso ad essa, ci porta inevitabilmente a esserne saturi, con la conseguenza che essa ci rimanga molto meno impressa, venendo dunque dimenticata.
Un buon metodo per ridurre la scelta dei brani da ascoltare è sfruttando le **playlist**, insiemi di brani con caratteristiche comuni, ad esempio il *mood* o il genere del brano.
Tuttavia, per gli ascoltatori più incalliti che fruiscono di migliaia e migliaia di brani diversi, creare delle playlist a partire dalla propria musica preferita non è esattamente impresa semplice.

1.2 Progetto *SpotifAI*

A questo punto, potremmo porci l'obiettivo di sfruttare l'Intelligenza Artificiale per tentare di affrontare il problema presentato sviluppando un sistema che sia in grado di realizzare delle *playlist* coerenti e piacevoli all'ascolto a partire dalla musica preferita di un utente sulla popolare piattaforma di streaming musicale **Spotify**; è qui che entra in gioco il progetto **SpotifAI**, che si propone di aiutare gli utenti più indecisi ad organizzare la propria musica.
Spotify offre già un servizio simile, nella forma dei Daily Mix personalizzati per gli utenti, che hanno però un problema: sono *contaminati* da brani sponsorizzati e mai ascoltati dall'utente, spesso nemmeno vicini ai suoi gusti. Si desidera quindi limitare il sistema alla propria libreria senza coinvolgere brani estranei.
Ciò che sicuramente ci verrà in aiuto sono i parametri con i quali Spotify classifica i brani presenti in catalogo. Infatti, oltre a genere, popolarità, lingua, durata e altre classificazioni banali che descrivono un brano, Spotify mantiene anche delle cosiddette *features*, che caratterizzano un brano in maniera più specifica:

- **Danceability** - descrive quanto un brano è "ballabile", tenendo conto del suo tempo, del ritmo e dall'energia del beat.
- **Valence** - descrive l'umore del brano. Un valore basso indicherà una canzone dal suono triste o negativo, mentre un valore alto descriverà

un brano positivo e allegro.

- **Energy** - indica l'intensità di un brano. Una canzone death metal avrà un'alta energia, mentre una ballad avrà bassa energia.
- **Tempo** - nella teoria musicale, il tempo è la velocità di un brano, e si rappresenta in BPM (beats per minute).
- **Loudness** - rappresenta la "chiassosità" del brano.
- **Speechiness** - indica la quantità di parlato presente nel brano.
- **Instrumentalness** - indica la quantità di strumentale o meno. Un valore basso indicherà un brano ricco di cantato.
- **Liveness** - rileva la presenza di pubblico all'interno di un brano, dunque se esso è una versione *live* o studio.
- **Acousticness** - indica se il brano è in acustico o meno.
- **Key** - la chiave in cui è scritto il brano.
- **Mode** - indica se il brano è in maggiore o in minore.

Conoscendo tali *features*, passiamo ora ad analizzare il problema per realizzare un modello di Machine Learning che possa esserci d'aiuto.

2 Analisi del Problema

2.1 Specifica PEAS

Un ambiente viene descritto identificandone **P**erformance, **E**nvironment, **A**ctuators, **S**ensors.
Nel caso di SpotifAI:

- **P** - una buona performance si avrà nella generazione di playlist coerenti e valide in fase di clustering, e poi nel saper predire a quale playlist già esistente assegnare un brano.
- **E** - l'ambiente è costituito da parte dei brani presenti nella libreria Spotify di quanti eseguiranno il codice.
- **A** - grazie alla libreria Spotipy, SpotifAI è in grado di creare le playlist direttamente sull'account di chi lo usa.
- **S** - una volta effettuato il clustering, i sensori ottengono un nuovo brano in input all'utente e ne identificano il cluster di appartenenza.

2.2 Proprietà dell'ambiente

L'ambiente risulta essere:

- **Completamente Osservabile**, perché i brani su cui lavorare sono tutti visibili dall'agente;
- **Deterministico**;
- **Episodico**;
- **Discreto**;
- **Statico**, sebbene potrebbe essere implementato dinamicamente se il dataset venisse aggiornato alla predizione del cluster di un brano;
- **Single-agent**.

3 Scelta dell'Algoritmo

3.1 Tecniche di Machine Learning

Si potrebbe pensare di affrontare il problema con un algoritmo di Machine Learning, basandoci sui dati dei brani già presenti nella playlist da espandere.

- **Classificazione** - la classificazione è una tecnica di apprendimento supervisionato che costruisce un modello (o classificatore), che, appunto, classifica i nuovi elementi sulla base del training set.
  Tra i classificatori più importanti se ne annoverano due, il *Naive Bayes*, che applica l'omonimo teorema alle istanze da classificare e il *Decision Tree*, basato su entropia, che crea un albero in cui i nodi rappresentano un sotto-insieme di caratteristiche del problema e gli archi rappresentano delle decisioni.

- **Regressione** - la regressione è una tecnica di apprendimento supervisionato, che predice il valore di una variabile numerica sulla base di un training set, usando un regressore, ossia una funzione matematica che cerca di descrivere i dati.
  La regressione può essere singola o multipla in base al numero di regressori che si hanno a disposizione per la predizione. Alla fine, l'obiettivo è quello di tracciare una retta che si adatti ai dati del training set a disposizione.

- **Clustering** - il clustering è una tecnica di apprendimento non supervisionato che classifica i dati senza assegnarvi etichette, creando dunque gruppi di oggetti dalle simili caratteristiche.
  Tali algoritmi, quindi, si basano sul concetto di similarità, che deve essere alta tra gli elementi del gruppo, e bassa tra diversi gruppi.

A questo punto, è chiaro che il problema in questione è più vicino a una soluzione mediante **clustering**: si andranno a realizzare dei cluster, che verranno poi convertiti in playlist e aggiunti automaticamente all'account Spotify dell'utente.

3.2 Algoritmi di clustering

- **k-means** - L'algoritmo ***k-means*** è uno tra i più famosi tra quelli a partizionamento iterativo ad errore quadratico, che sfrutta i **centroidi**, ossia un punto appartenente allo spazio delle features che media le distanze tra tutti i dati appartenenti al cluster ad esso associato.
  L'algoritmo sceglie i centroidi in maniera casuale o pseudo-casuale, genera un partizionamento assegnando ogni campione al centroide più vicino, calcola i nuovi centroidi e ripete fino al cambiamento dei centroidi. Ha una buona efficienza, e con cluster ben fatti restituisce buoni risultati: bisogna però impostare un valore di k cluster a priori, che può essere stabilito su dati empirici ma anche sfruttando alcune metriche.

- **Clustering gerarchico** - il clustering gerarchico cerca di considerare raggruppamenti multi-livello, riconoscendo gruppi di pattern con caratteristiche simili a un certo livello, e forma un **dendrogramma**, che si può navigare per decidere il numero di cluster. Può essere divisivo o agglomerativo in base che si parta dalla similarità più bassa o più alta.

- **Density-based clustering** - il clustering basato sulla densità raggruppa pattern sulla base della loro densità nella distribuzione, basandosi sui parametri *minPts* ed ϵ. Uno degli algoritmi più famosi di questo tipo è il **DBSCAN**.

La scelta dell'algoritmo è ricaduta sul *k-means* grazie alla sua facilità di implementazione e rapidità.
È stato fatto un tentativo anche mediante clustering gerarchico agglomerativo, ma le playlist risultanti si sono rivelate meno piacevoli all'ascolto rispetto a quelle generate tramite k-means.

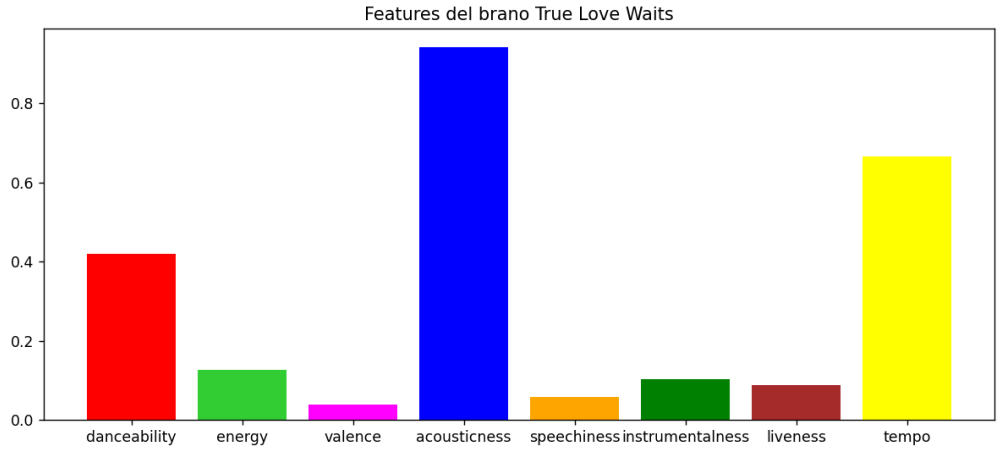
4 Implementazione e Realizzazione Demo

4.1 Raccolta dei Dati

È necessario, adesso, procurarsi i dati sui brani da organizzare in playlist. Siccome il sistema andrebbe basato sugli ascolti del proprio profilo personale, è necessario scrivere una funzione di **scraping** per ottenere alcuni brani già presenti nella nostra libreria.
Il profilo di partenza sarà quello del sottoscritto, e per facilitare le cose è stato impiegato uno script già realizzato trovato in Rete[1], ed è presente nella *repository* del progetto.
Il risultato è un *.csv di circa 2000 righe di brani presi da playlist già presenti nel profilo del sottoscritto, anche questo disponibile nella .
Segue, nella successiva sottosezione, una descrizione più approfondita sul come è stato possibile accedere e operare sul proprio profilo Spotify mediante script.

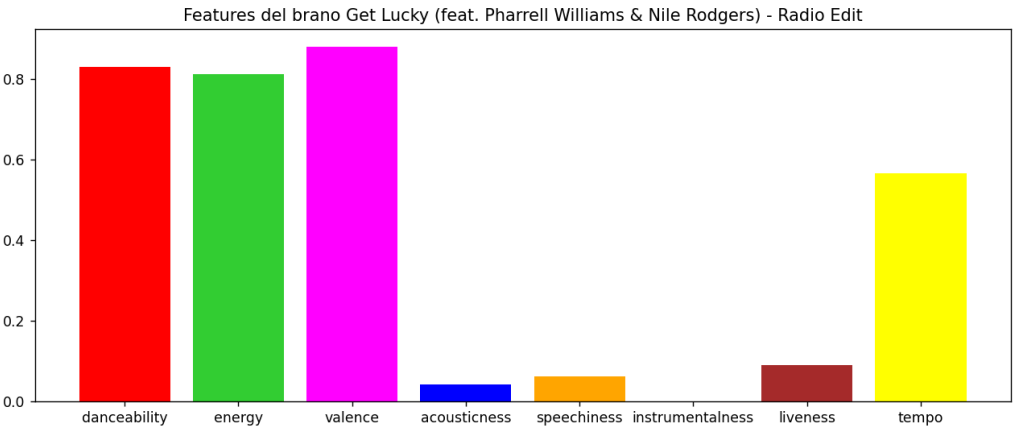
4.2 Introduzione a Spotipy e features di un brano

Per l'implementazione si utilizzerà Python; in particolare, per interfacciarsi con le API di Spotify al fine di accedere ai brani della propria libreria, creare playlist e tanto altro, verrà impiegata la libreria **SpotiPy** che richiede l'accesso alla piattaforma per sviluppatori di Spotify, con validazione tramite login e token segreto.
Informazioni sul come accedere come sviluppatore al proprio account Spotify non verranno trattate in questo documento: *Google is your best friend*!
Tra le tante funzionalità di Spotipy, alcune interessanti riguardano la possibilità di riprodurre brani (riservata a utenti *premium!*), creare ed eliminare playlist, salvare canzoni nel proprio account e tanto altro.
Ciò che ci interessa, per ora, è la possibilità di ricavare le feature di un brano, tramite la funzione `audio_features()`, che ci permetterà di ricavare i parametri elencati precedentemente elencati.
Proviamo subito ad eseguire questa funzione su due brani totalmente diversi tra loro.
Il primo è *True Love Waits*, della band britannica *Radiohead*. Si tratta di un brano notoriamente triste e malinconico, sia nel testo che nella parte musicale.



Analisi delle features del brano *True Love Waits*

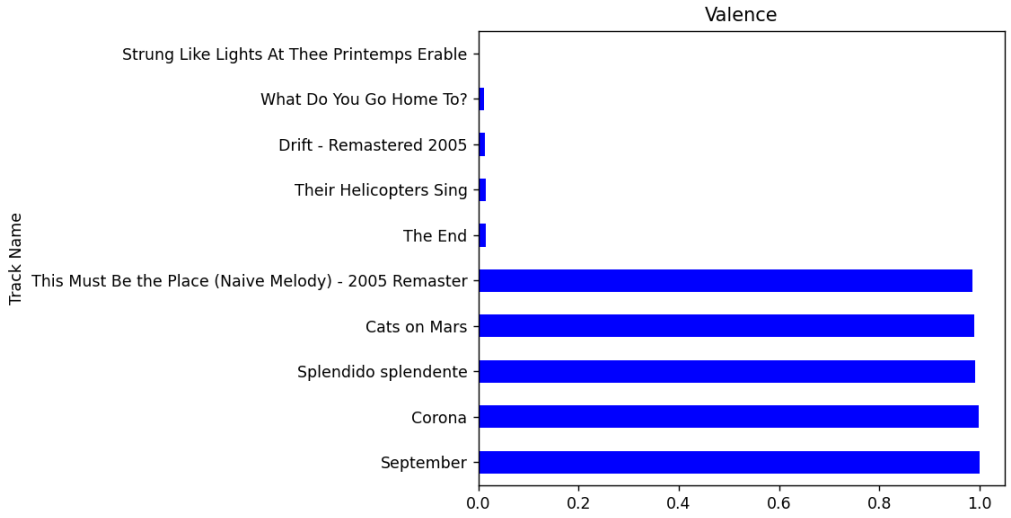
Tra i valori più rilevanti, è interessante notare la bassissima valenza ed energia, mentre si ha un'acustica abbastanza alta.
Riproviamo ad eseguire la funzione su un brano totalmente diverso, come *Get Lucky*, del famosissimo duo *Daft Punk*.



Analisi delle features del brano *Get Lucky*

È immediato che l'energia risulta ora molto più alta, come anche la valenza.

Scende anche il valore di "acusticità", contenendo molti suoni derivati da sintetizzatori.
È stato anche interessante ricavare qualche grafico che confrontasse i brani presenti nella mia libreria, ad esempio qui sono stati rappresentati i brani con minore e maggiore valenza.



Esempio di analisi della valenza

4.3 Preparazione dei dati

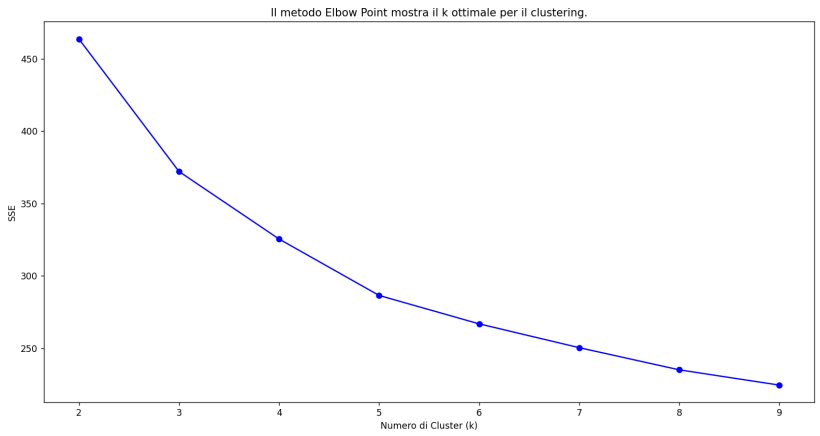
Per assicurare la massima resa del prodotto finale, è stato necessario lavorare sui dati per assicurarne la qualità.
Se non è stata necessaria una fase di data cleaning, in quanto - fortunamente - si è lavorato su un dataset privo di dati rumorosi o mancanti, è stato necessario applicare delle tecniche di **feature scaling**, per consentire di normalizzare o scalare i valori di una certa caratteristica.
In particolare, tra i diversi metodi possibili si è scelto di adoperare la *min-max normalization*:

$$x^i = a + \frac{(x - min(x))(b - a)}{max(x) - min(x)}$$

a e b rappresentano i valori minimo e massimo da ottenere nella normalizzazione. Nel nostro caso, si sono scelti *0* e *1*.
Inoltre, è stato necessario selezionare le features sulle quali basare il modello. Si sono scelte quindi tutte le features illustrate precedentemente (*danceability*, *valence*, *tempo*...), eccezion fatta per la ***mode*** del brano, che distingue i brani in chiave maggiore o minore, in quanto, dopo una serie di tentativi, ci si è resi conto che considerare questo valore andava a condizionare in maniera eccessiva il modello, diventando fuorviante e rendendo la playlist finale meno coerente.

4.4 Realizzazione del modello di Machine Learning

A questo punto, è possibile iniziare a realizzare il modello di Machine Learning.
Una volta ottenuto un *dataframe* normalizzato contenente i brani in esame grazie all'ausilio della libreria *pandas*, è stato necessario compiere due operazioni fondamentali, in risposta a dei problemi che si sono presentati.
Il primo problema riguardava la scelta del **numero di cluster** da realizzare. Come già detto, infatti, l'algoritmo *k-means* richiede una scelta a priori del numero di cluster, che non può essere assolutamente presa sotto gamba o approssimata. Si usa quindi il metodo empirico del **punto di gomito**, implementabile in Python grazie alla libreria *sklearn*.



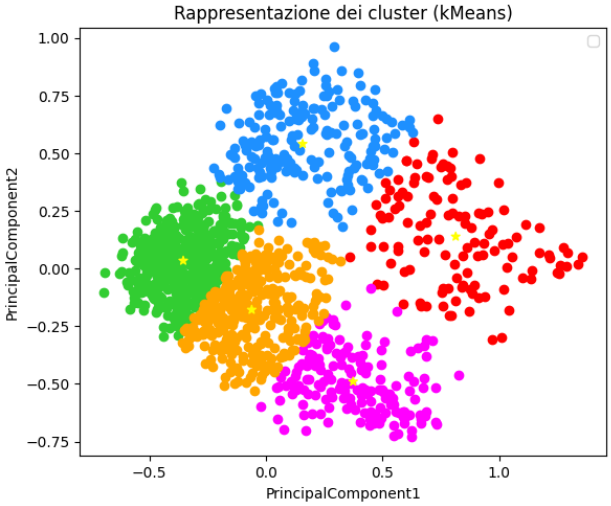
Applicazione del metodo del punto di gomito al problema

Dalla figura, si deduce che un buon compromesso tra errore e cluster è in corrispondenza del valore di k = 4 o 5, dunque si è deciso di effettuare il k-means con 5 cluster.
Un altro problema è legato alla **dimensionalità del dataset**; avere un numero di attributi così elevato, infatti, non permette di realizzare una rappresentazione mediante uno spazio n-dimensionale. È necessario, dunque, applicare una tecnica di **riduzione della dimensionalità**, che risolve anche problemi legati alla performance dell'algoritmo. Nel caso specifico di SpotifAI si è scelto di usare **PCA** (**Principal Component Analysis**), che trasforma un insieme di variabili in un numero inferiore (in questo caso, 2) di cosiddette *componenti principali*. Il funzionamento dell'algoritmo, fornito

da *sklearn* non è stato approfondito: *it just works!*

Prima di applicare l'algoritmo *k-means*, è necessaria un'ultima attenzione: abbiamo detto che tale tecnica inizializza i centroidi in maniera casuale, ma, per come viene gestito da *sklearn*, anche l'ordine in cui si generano viene randomizzato ad ogni esecuzione: siccome a partire dai cluster vanno realizzate delle playlist, non possiamo permetterci di scambiare i centroidi nel nostro spazio ad ogni esecuzione. È stato quindi necessario fissarli, mantenendo quindi l'ordine.
Si applica ora l'algoritmo, e si generano i cluster.

```
kmeans = KMeans(5, init=init, max_iter=100)
assigned_clusters = kmeans.fit_predict(df_train)
print("Cluster creati")
```



Plot dei cluster generati dal kMeans. Notare PC1 e PC2 lungo gli assi.

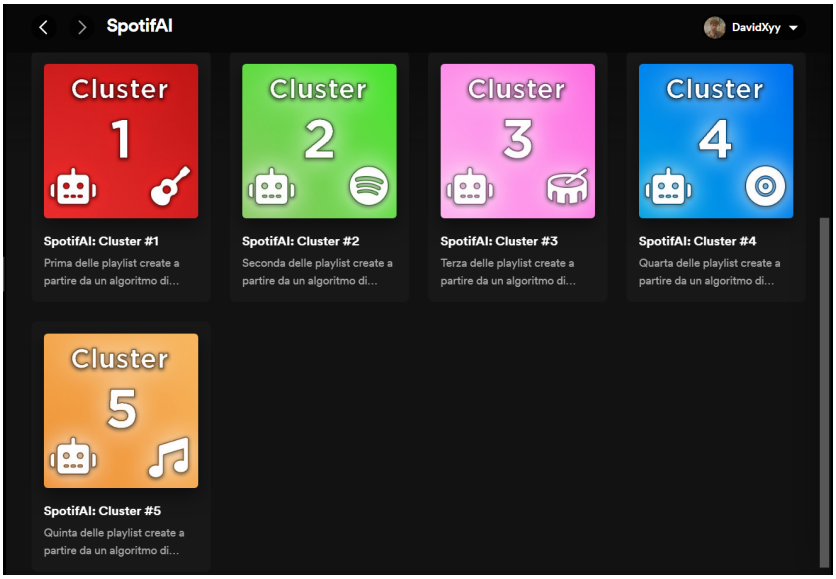
Passando oltre, grazie alla funzione `kmeans.predict()` fornita da sklearn, è stato anche possibile realizzare la predizione di quale playlist fosse adatta per un brano in input.

```
df_test = pca.transform(song_df)
pred = kmeans.predict(df_test)
print("Il brano potrebbe rientrare nella playlist ", pred + 1 )
```

Il gioco è fatto! I cluster con i diversi brani sono stati creati, ora non resta che realizzare le playlist a partire da essi. Per evitare *off-topic*, non verrà trattato il discretamente lungo processo di esportazione in questo documento, il cui focus principale è l'applicazione degli algoritmi di ML: si faccia riferimento, al solito, alla *repository*.
Nell'avviarci verso le conclusioni, segue una fase di analisi dei risultati ottenuti.

5 Prodotto finale

L'idea base di SpotifAI è stata implementata ed è funzionante: accedendo all'app Spotify, sono state create automaticamente le playlist sulla base dei cluster. Per una migliore comprensione del risultato finale, è possibile visu-

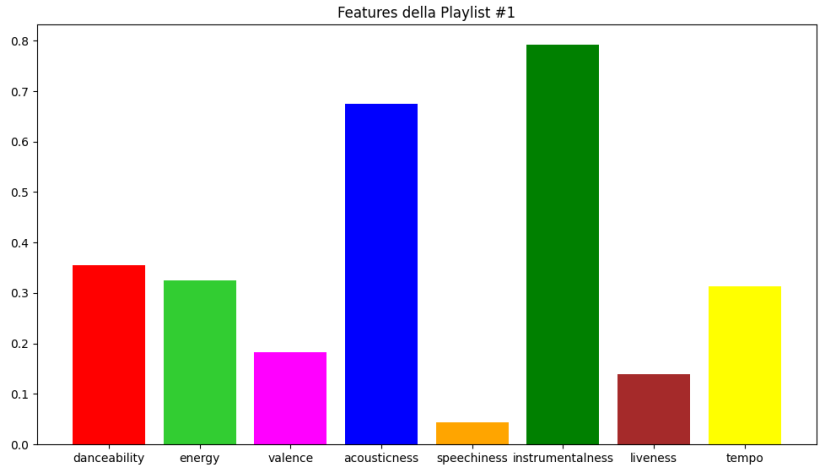


Playlist realizzate da SpotifAI su Spotify

alizzare nel dettaglio le features delle cinque playlist.

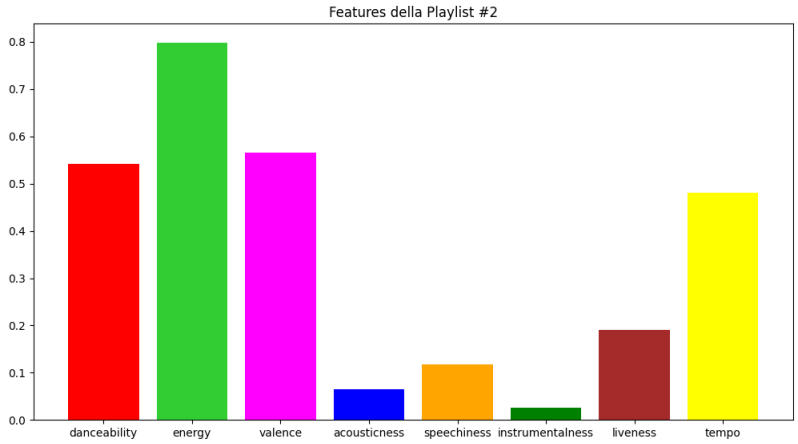
Playlist 1

Nella prima playlist creata, spicca l'elevato livello di instrumentalità e di acusticità: sono presenti molte tracce puramente strumentali, jazz o di generi derivanti, colonne sonore, e generalmente anche brani rilassanti.
Tra gli artisti più presenti spiccano *Brian Eno*, *Godspeed You! Black Emperor*, *The Microphones* e, curiosamente, *Fabrizio de Andrè*.
Una spiegazione possibile risiede nel timbro di voce notoriamente basso del cantautore genovese, dunque potrebbe confondersi, talvolta, con la strumentale, portando alla confusione nel valore.



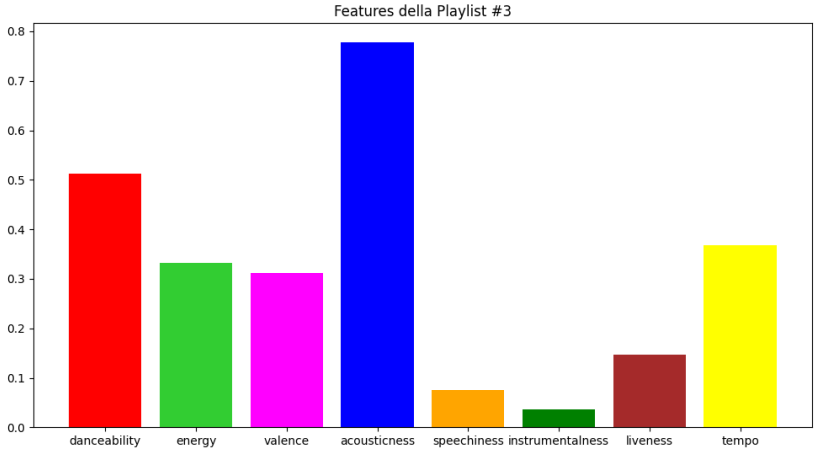
Playlist 2

La seconda playlist è caratterizzata da un'elevatissima energia, con valori di ballabilità, valenza e tempo abbastanza alti, con bassi valori acustici e di strumentale.
Ci troviamo quindi dinanzi a una playlist con brani pieni di parole cantate e di generi vicini al rock, al punk e al rap.
Tra gli altri, presenti un gran numero di brani degli *Afterhours*, *Eminem*, *Arctic Monkeys*.



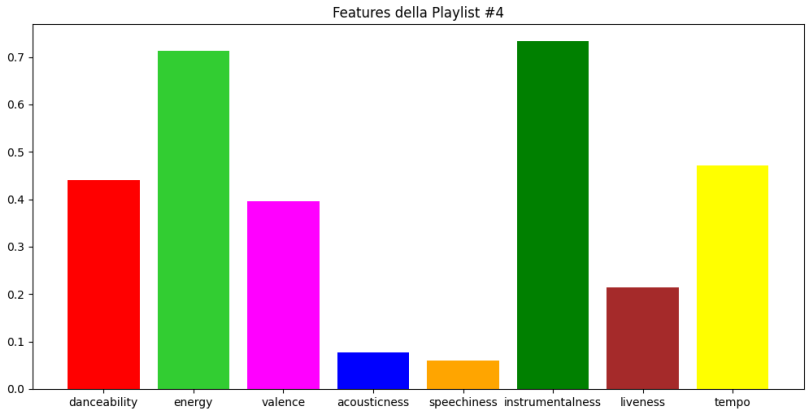
Playlist 3

La terza playlist è caratterizzata da un alto valore di acusticità, con tempo, energia e valenza bassi. Anche qui avremo brani prevalentemente cantati ma dal mood "triste", dunque conterrà brani cantautoriali, di autori come *Elliott Smith*, *I Cani*, *King Krule*.



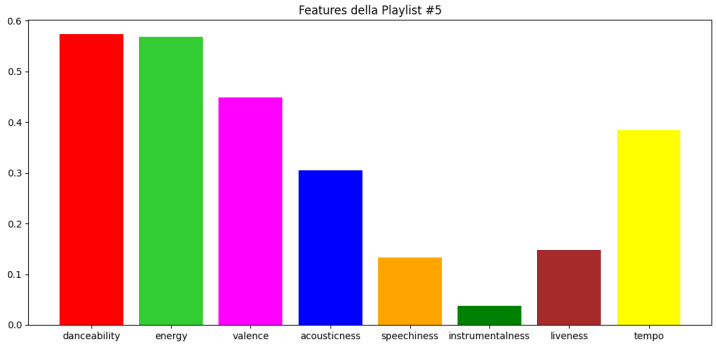
Playlist 4

Nella quarta playlist, l'alta energia e strumentalità fanno da padrone. La bassa acusticità, inoltre, sottintende l'alto numero di numerosi pezzi elettronici o tramite strumenti "effettati", giustificando la presenza di artisti come *Kraftwerk*, *The Smashing Pumpkins*, *Verdena* e, come nel caso di De Andrè per la prima playlist, il timbro vocale di *Ian Curtis*, cantante della band *Joy Division* viene confuso con la strumentale.



Playlist 5

Infine, la quinta playlist è caratterizzata da brani cantati con valori nella media in ballabilità, energia, tempo e valenza. Possiamo aspettarci un alto numero di ballad, ma sembra effettivamente una playlist quasi *jolly*, visto che non si riesce ad identificare un genere preciso, presentando simultaneamente artisti come *Radiohead*, *Kendrick Lamar* e *Calcutta*.



Adesso, non resta che godersi le playlist, disponibili pubblicamente per chi vorrà ascoltare della buona musica! Questa fase non è da sottovalutare, in quanto potrebbe costituire un'operazione di *crowdsourcing* che, per motivi di tempo e di scarsa (se non nulla!) possibilità economica non è stata realizzata. Sarebbe stato tuttavia interessante e utile ai fini della valutazione raccogliere opinioni del pubblico tramite questionari e sondaggi, anche retribuiti.

6 Conclusioni e sviluppi futuri

Finalmente il progetto, che sebbene riassunto in poche pagine ha richiesto tanto tempo e impegno può considerarsi, per il momento, concluso.
Sono evidenti le enormi possibilità di un progetto simile, ad esempio si può pensare di sfruttare altre caratteristiche dei brani, come la lingua, il genere (non banale, infatti i sotto-generi assegnati da Spotify ai brani sono talvolta confusionari se non totalmente errati) o le *lyrics* di un brano, sfruttando il *Natural Language Processing*. Quest'ultima possibile aggiunta ha una difficoltà non da sottovalutare, siccome la musica è composta da testi poetici, per definizione ricchi di figure retoriche e altre forme di linguaggio che possono risultare ambigue per il modello.

Un altro aspetto da rivedere sta nell'impiego del parametro della *speechiness*, effettivamente davvero poco utile, risultando davvero elevato (e dunque significativo) solo in caso di podcast, monologhi e pochi altri contenuti.

Infine, avrei voluto approfondire alcune tecniche per misurare l'accuratezza della corrispondenza tra un nuovo brano aggiunto e una playlist, oltre che sfruttare magari la metrica del *silhouette score* per validare i cluster ottenuti.

Da un punto di vista personale, infine, posso dire che realizzare un lavoro del genere è stato davvero soddisfacente, anche perché il dominio scelto mi è molto vicino e ciò ha reso molto sicuramente più piacevole l'esperienza progettuale (e, soprattutto, ho guadagnato delle ottime playlist da ascoltare!)

Crediti

[1] Ido Lior (2020) *Using Spotify's audio features*, kaggle.com