Studio su dataset generato dai miei ascolti spotify

Alessandro Carella – Matricola: 726006

Email: [a.carella49@studenti.uniba.it](mailto:a.carella49@studenti.uniba.it)

Link repository GitHub: <https://github.com/AlessandroCarella/ingegneria-della-conoscenza>

Requisiti funzionali:

Il progetto è stato realizzato in Python, in quanto si presta maggiormente alla trattazione e l’analisi dei dati, mentre come ambiente di lavoro è stato utilizzato Visual studio code (con annesse relative estensioni).

Librerie utilizzate:

Matplotlib: visualizzazione dei grafici;

Numpy: visualizzazione dei grafici;

Pandas: importazione dei dataset .csv;

Scikit-learn: applicazione concetti di ML;

Networkx: creazione di grafici.

Pgmpy: creazione della rete bayesiana

Spotipy: creazione del dataset e query per la user interface

Tqdm: barre di avanzamento per l’installazione automatica dei requirements

Installazione e avvio:

* Installare [visual studio code](https://code.visualstudio.com/) e l’[estensione per compilare i file .py](https://marketplace.visualstudio.com/items?itemName=ms-python.python) ;
* Installare [python 3.9](https://www.python.org/downloads/release/python-390/) mediante sito ufficiale;
* Aprire l’editor di testo;
* Aprire il file “main.py” e premere il tato F5.
  + Alla riga 2 del file “main.py” è presente l’installazione automatica delle librerie contenute all’interno del file “requirements.txt”, la linea che effettua questa operazione è commentata nella versione del progetto consegnata per evitare di installare packages non necessariamente voluti durante un avvio del programma effettuato per errore ma per funzionare il progetto ha bisogno di utilizzare le versioni delle librerie elencate nel file requirements.txt. Un modo alternativo per installare i packages è entrare nella cartella del progetto attraverso il prompt dei comandi o windows powershell (entrambi se il sistema operativo utilizzato è windows) e eseguire il comando “pip install -r requirements.txt”

Creazione del dataset:

Il dataset è stato da me generato tramite i [dati forniti da Spotify](https://support.spotify.com/us/article/data-rights-and-privacy-settings/) e utilizzando i seguenti campi estratti dal file ‘’StreamingHistory.json” fornito dallo stesso e le chiamate ai servizi API di Spotify tramite la libreria spotipy “spotify.audio\_features(songsUri)” e spotify.search(q=query ', type='track, album'):

Danceability: metrica spotify



Energy: metrica spotify

Key: metrica spotify

Loudness: metrica spotify

Speechiness: metrica spotify

Acousticness: metrica spotify

Instrumentalness: metrica spotify

Valence: metrica spotify

Tempo: metrica spotify

TrackIsexplicit: metrica spotify (campo “explicit” dall’api spotify)

SongIsLiked: campo calcolato attraverso questo calcolo:

numero di millisecondi riprodotti in totale dato dalla sommatoria del campo relativo per ogni istanza della stessa canzone all’interno del file json

/

durata della canzone in millisecondi

Se il risultato di questo calcolo è un numero ≥ 50, il campo SongIsLiked sarà pari ad 1, in caso contrario sarà pari a 0.

(per altre informazioni sulle metriche di spotify visitare la [documentazione](https://developer.spotify.com/documentation/web-api/reference/#/operations/get-several-audio-features) o scorrere in basso fino alla sezione “Feature del dataset”)

Preprocessing del dataset:

Il preprocessing del dataset è stato realizzato nel file per la creazione del dataset (“spotifyDataExtraction\jsonsWithNeededFeaturesToCsv.py”) con successiva normalizzazione dei dati attraverso questa formula:

int((math.cos(value) + 1) / 2 \* 100)

Questa normalizzazione si è dimostrata necessaria in quanto, durante la creazione del modello per la realizzazione della rete bayesiana, mantenere i dati senza normalizzarli portava ad un uso eccessivo di memoria.Immagine che contiene testo

Descrizione generata automaticamente

Feature del dataset:

* danceability: La danzabilità descrive quanto un brano è ‘’adatto’’ al ballo, basandosi su una combinazione di elementi musicali tra cui tempo, stabilità del ritmo, forza del beat e regolarità generale.
* energy: L'energia rappresenta una misura percettiva di intensità e attività. Tipicamente i brani energici appaiono veloci, rumorosi e caotici. Ad esempio, il death metal ha un'alta energia, mentre una preludio di Bach ha un punteggio basso sulla scala.
* key: La chiave in cui si trova il brano.
* loudness: Il volume generale del brano in decibel (dB). I valori di volume sono mediati su tutto il brano e sono utili per confrontare il volume relativo dei brani.
* speechiness: Speechiness rileva la presenza di parole parlate in una traccia audio. Più la registrazione assomiglia a un discorso (ad esempio, un programma radiofonico, un audiolibro, una poesia), più il valore dell'attributo si alza.
* acousticness: Un indicatore di confidenza che indica se la traccia è acustica.
* instrumentalness: Prevede se una traccia non contiene parti vocali. In questo contesto, i suoni come "ooh" e "aah" sono considerati strumentali. Le tracce di rap o di parole parlate sono chiaramente "vocali".
* valence: Descrive la positività della musica. Brani con valore di valence alto suonano più “positivi” (felici, allegri, euforici), mentre quelli con valore di valence basso suonano più “negativi” (ad esempio tristi, depressi, arrabbiati).
* tempo: Il tempo indica la velocità o il ritmo di una canzone, espresso in battiti per minuto (BPM).
* trackIsexplicit: Un valore booleano che indica se il brano contiene contenuti espliciti (come linguaggio forte o contenuti sessualmente espliciti).
* songIsLiked: campo calcolato attravero questo calcolo:
  + numero di millisecondi riprodotti in totale dato dalla sommatoria del campo relativo per ogni istanza della stessa canzone all’interno del file

json

/

la durata della canzone in millisecondi

* + se il risulato di questo calcolo è un numero maggiore o uguale a 50 il campo songIsLiked sarà pari ad 1, altrimenti sarà pari a 0

Panoramica dei dati:

Attraverso la stampa di una tabella viene mostrata a schermo una panoramica delle informazioni contenute nel dataset:

Immagine che contiene testo, elettronica

Descrizione generata automaticamente

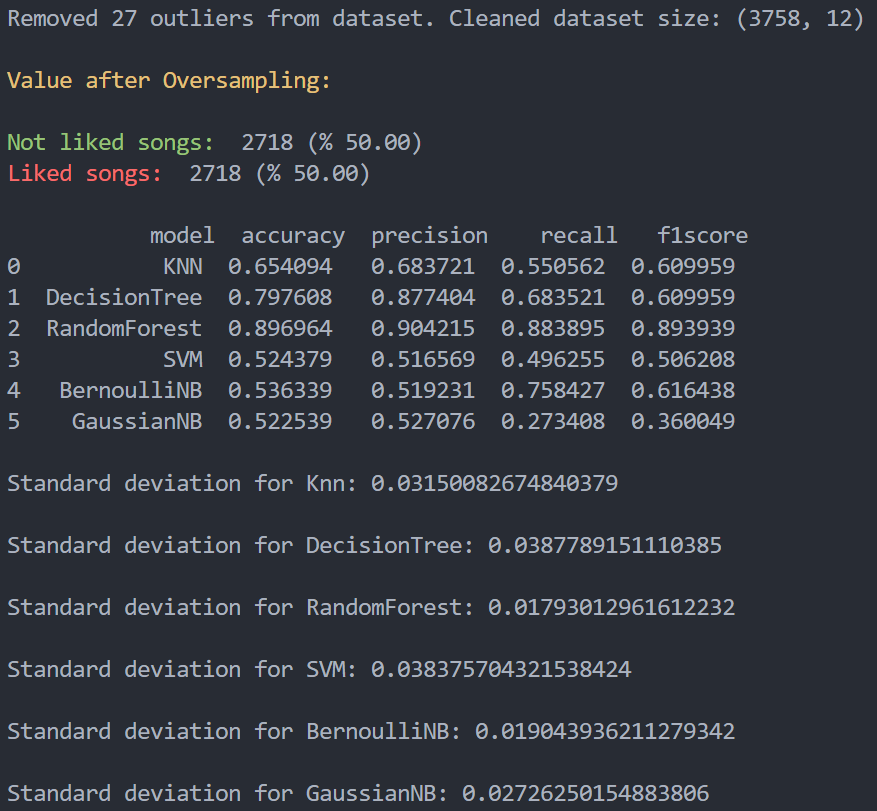
Immagine che contiene testo, elettronica

Descrizione generata automaticamente

Clustering per rimozione outliers:

Per quanto riguarda l’utilizzo dell’apprendimento non supervisionato ho deciso di implementare la ricerca di cluster per la rimozione degli outliers attraverso una tecnica di soft clustering chiamata Expectation-Maximization (EM).

* Utilizzo il modello di Gaussian Mixture per addestrare un cluster sul dataset
* Calcolo le likelihoods, che sono le probabilità che ogni punto del dataset appartenga a ciascun cluster.
* Calcolo una soglia per identificare gli outlier del dataset come i punti con likelihood inferiore a una certa soglia,
* TODO come calcoli la soglia
* Rimuovo gli outlier dal dataset e stampo il numero di outlier rimossi e la dimensione del dataset pulito.



Bilanciamento delle classi:

attraverso un grafico, realizzato con pyplot (matplotlib) vediamo che i dati del dataset non sono bilanciati:

Immagine che contiene grafico, grafico a torta

Descrizione generata automaticamente

Classi squilibrate inficiano sulla precisione. Questo è un problema molto comune nell’apprendimento automatico (in particolare nella classificazione), che si verifica in set di dati con un rapporto sproporzionato di osservazioni per le varie classi di dati coinvolte.

La precisione standard non misura più in modo affidabile le prestazioni, il che rende l’addestramento del modello molto più complicato.

Vi sono diversi modi per poter risolvere il problema dello sbilanciamento delle classi. La soluzione che ho deciso di applicare è l’oversampling. Per far ciò ho individuato la classe con più dati e la classe con meno dati ed ho effettuato un resampling facendo combaciare le occorrenze. Ottenendo così classi bilanciate:

Immagine che contiene grafico, grafico a torta

Descrizione generata automaticamente

TODO Apprendimento supervisionato:

Scelta del modello:

Per questo tipo di apprendimento abbiamo usato vari modelli per poi identificare quale fosse quello più adatto al nostro dataset

Questi sono i modelli valutati:

* KNN (K-Nearest Neighbors)
  + K-Nearest Neighbors è un algoritmo utilizzato per il riconoscimento di pattern e la classificazione di oggetti basandosi sulle caratteristiche degli oggetti vicini al campione in esame.
* Decision Tree
  + Decision Tree è un classificatore strutturato ad albero in cui ogni nodo può essere una foglia o un nodo interno. Se è una foglia, indica il valore della classe assegnata all'istanza. Se è un nodo interno, specifica il test effettuato su un attributo. Per ogni valore dell'attributo in un test, l'algoritmo crea un ramo e il relativo sottoalbero.
* Random Forest
  + Random Forest è un classificatore ottenuto dall'aggregazione di alberi di decisione tramite bagging. Questo metodo minimizza l'overfitting del set di training rispetto agli alberi di decisione.
* SVC (Support-Vector Classification)
  + SVC è un modello di apprendimento per la regressione e la classificazione. Dato un insieme di esempi etichettati con una delle due possibili classi, un algoritmo di addestramento per le SVC costruisce un modello che assegna i nuovi esempi a una delle due classi, ottenendo così un classificatore lineare binario non probabilistico.
* Classificatori Naive Bayes
  + BernoulliNB (Bernoulli Naive Bayes) è simile al multinomiale naive bayes, ma i predittori sono variabili booleane. I parametri utilizzati per prevedere la variabile di classe possono assumere solo i valori sì o no.
  + GaussianNB (Gaussian Naive Bayes) è una variante di Naive Bayes che segue la distribuzione normale gaussiana e supporta dati continui.

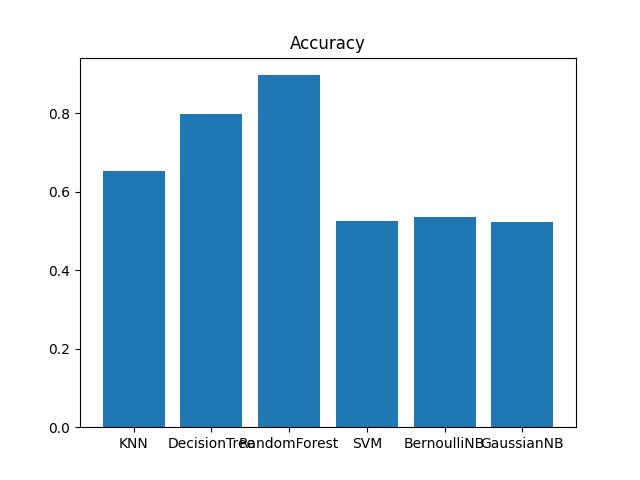
Su questi ho eseguito il K-Fold Cross Validation (in particolare il RepeatedKFold con 5 ripetizioni) per capire quale di questi sia il più attendibile. TODO check se stanno da aggiungere informazioni

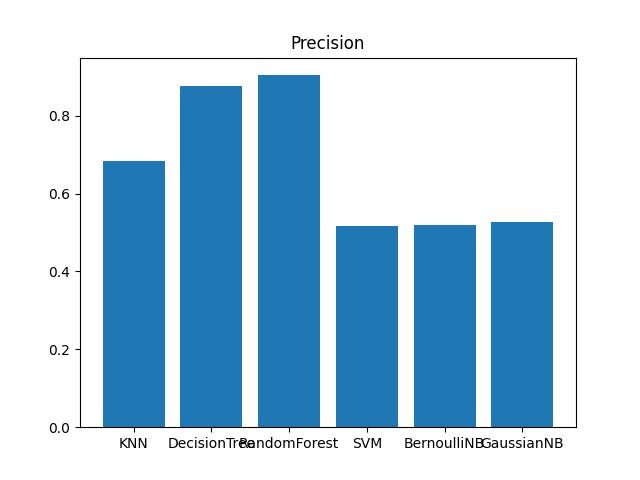
Le metriche utilizzate per la valutazione e i loro relativi risultati sono:

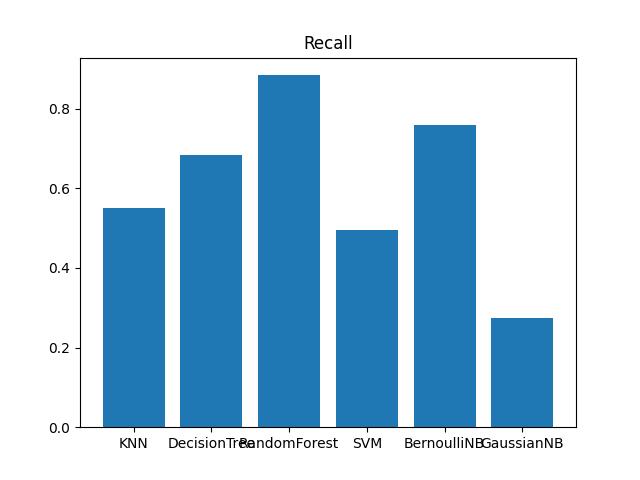
Immagine che contiene testo

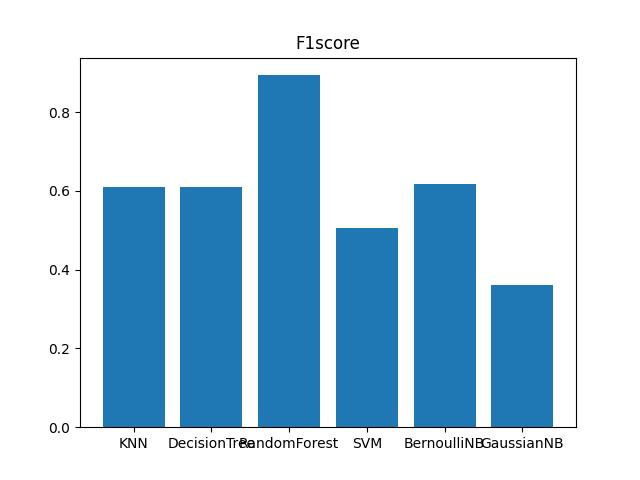
Descrizione generata automaticamente

In particolare:









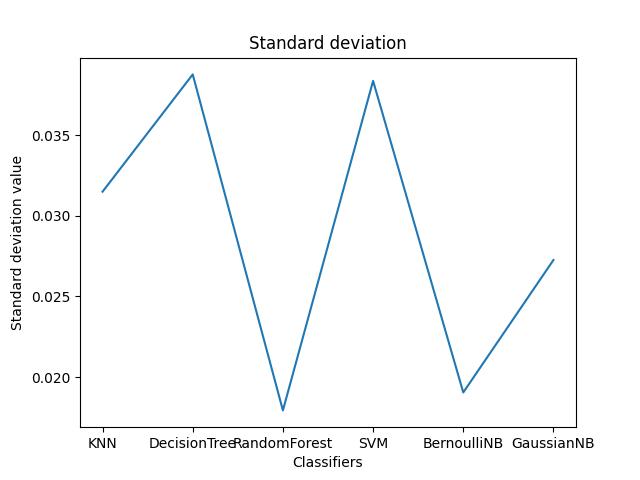


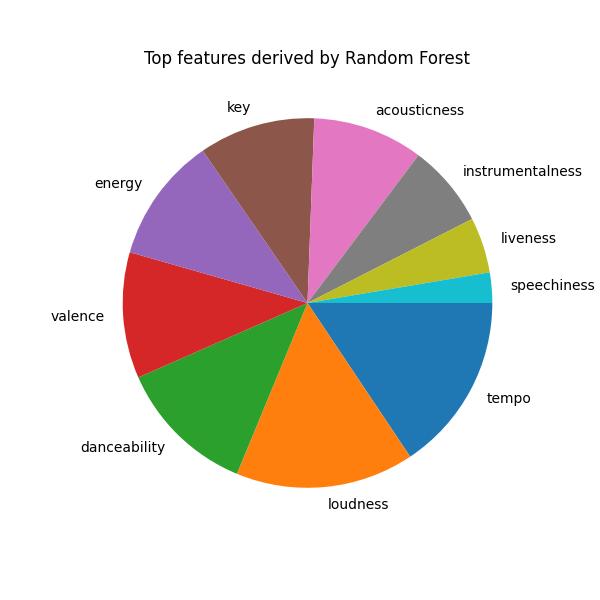
Immagine che contiene testo

Descrizione generata automaticamente

Il classificatore migliore risulta essere il random forest.

Verifica dell’importanza delle features:

Ho successivamente generato un grafico, basato sulle analisi effettuate in precedenza, che estrae le features più importanti basandosi sul random forest:



Creazione della rete bayesiana

Ho scelto di implementare una rete bayesiana per poter effettuare delle interrogazioni per verificare le probabilità delle features, utilizzando come metodo di scoring, il K2score.

Le tabelle CPD vengono generate con il dataset e andando ad usare il MaximumLikeliHoodEstimator creiamo una rete bayesiana completa, con le probabilità apprese dal dataset.

In questo modo la predizione avverrà in base alle features del dataset.

