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 esso risulta più idoneo per la trattazione e l’analisi dei dati e come ambiente di lavoro è stato utilizzato visual studio code con le estensioni relative.

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

#TODO aggiornare

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/) tramite il sito ufficiale
* aprire l’editor di testo
* aprire il file “main.py” e premere il tato F5

Creazione del dataset:

il dataset è stato da me generato tramite i dati forniti da spotify ed utilizzando i seguenti campi estratti dal file fornito da spotify “StreamingHistory.json” e le chiamate ai servizi api di spotify “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 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

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

Preprocessing del dataset:

Il preprocessing del dataset è stato realizzato nel file per la creazione del dataset (“spotifyDataExtraction\jsonsWithNeededFeaturesToCsv.py”) e i dati sono stati tutti normalizzati attraverso questo codice:

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

questo è stato necessario in quanto durante la creazione del modello per la realizzazione della rete bayesiana mantenere i dati senza normalizzarli portava ad un uso eccessivo di memoriaImmagine che contiene testo

Descrizione generata automaticamente

Feature del dataset:

* danceability: La danzabilità descrive quanto è adatto un brano per ballare basandosi su una combinazione di elementi musicali tra cui tempo, stabilità del ritmo, forza del beat e regolarità generale.
* energy: L'energia è una misura che rappresenta una misura percettiva di intensità e attività. Tipicamente, i brani energici si sentono veloci, rumorosi e caotici. Ad esempio, il death metal ha un'alta energia, mentre una preludio di Bach ha un punteggio basso sulla scala. Le caratteristiche percettive che contribuiscono a questo attributo includono la gamma dinamica, la percezione della sonorità, il timbro, il tasso di inizio e l'entropia generale.
* key: La chiave in cui si trova il brano. Gli interi si mappano in note utilizzando la notazione standard di Pitch Class.
* 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. Il volume è la qualità di un suono che è il correlato psicologico primario della forza fisica (ampiezza).
* 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 valence alto suonano più positivi (ad esempio felici, allegri, euforici), mentre quelli con 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). In termini musicali, il tempo è la velocità o il ritmo di una data composizione e deriva direttamente dalla durata media del battito.
* 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

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 abbiamo 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:

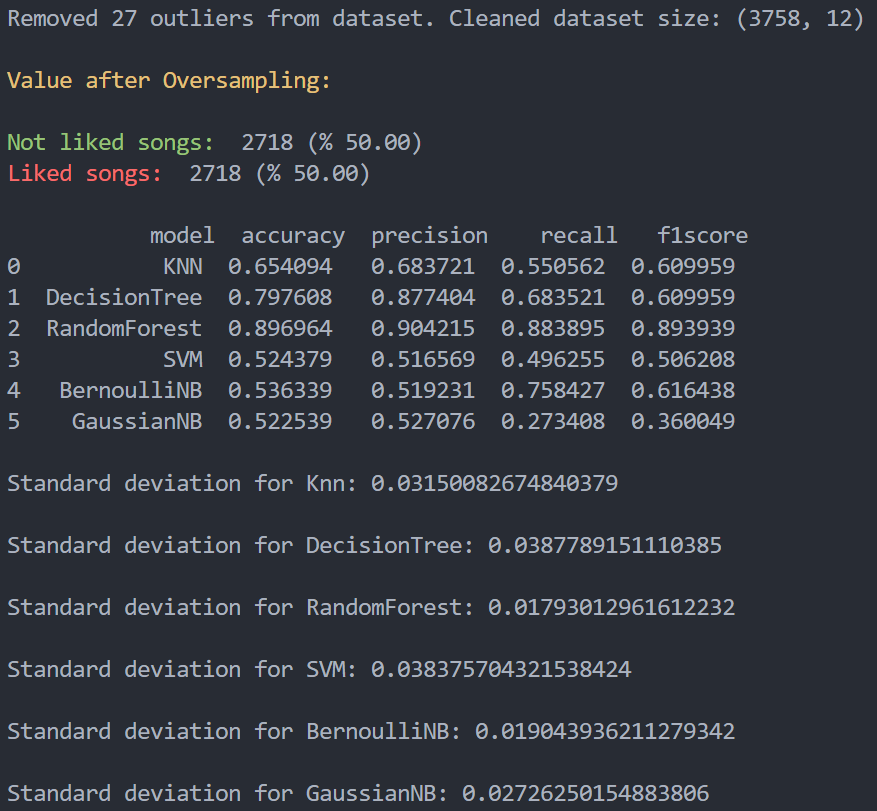
Immagine che contiene grafico, grafico a torta

Descrizione generata automaticamente

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.



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.

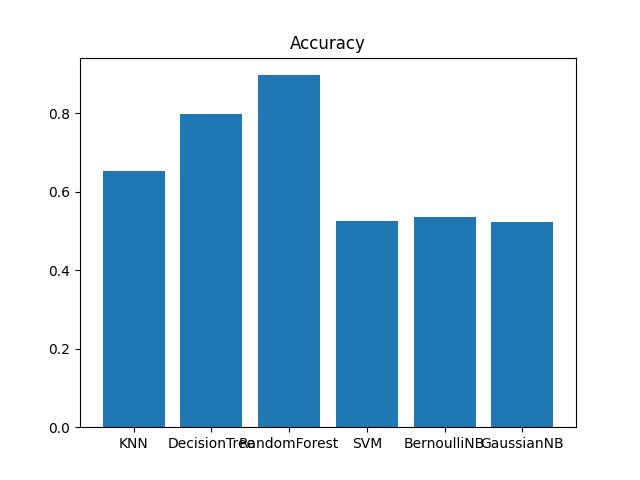
Su questi ho eseguito il K-Fold Cross Validation (in particolare il RepeatedKFold con 5 ripetizioni) per capire quale di questi sia il più attendibile. TODO check se stanno da aggiungere informazioni

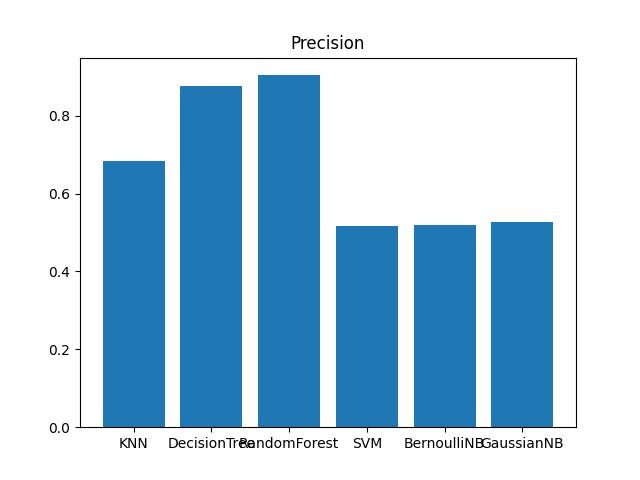
Le metriche utilizzate per la valutazione e i loro relativi risultati sono:

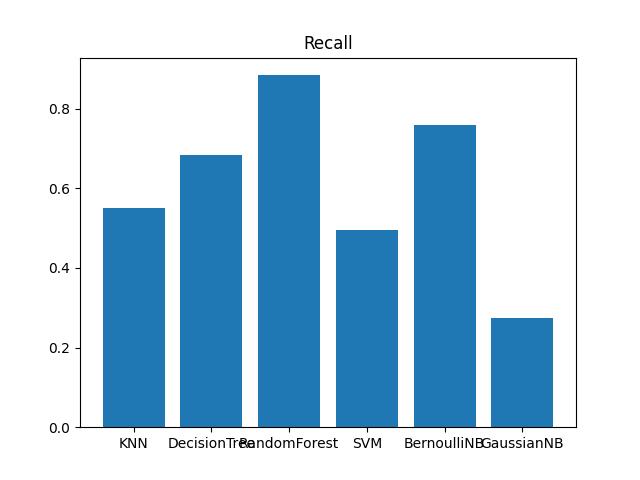
Immagine che contiene testo

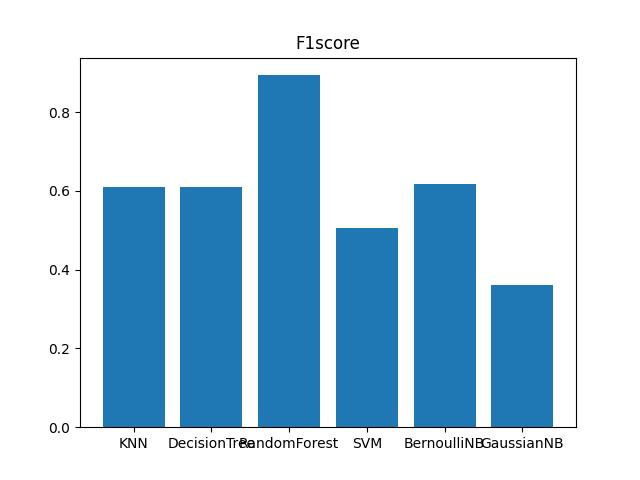
Descrizione generata automaticamente

In particolare:









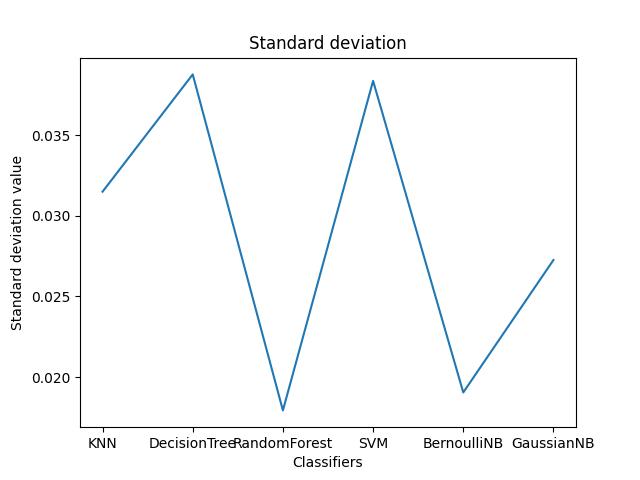


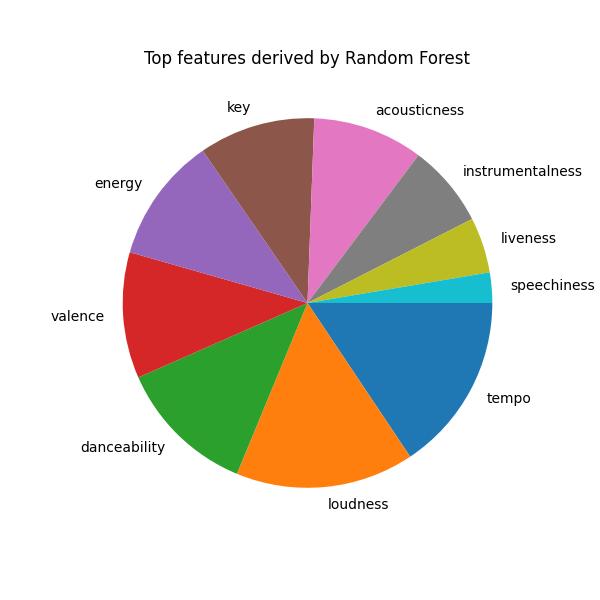
Immagine che contiene testo

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 interrogazione 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.

Immagine che contiene grafico

Descrizione generata automaticamente