Genres Classificator

Struttura dei file di progetto

Il file csv “track_genres_mio.csv” (o “track_genres_mio2.csv”) è stato creato filtrando i metadati offerti da Free Music Archive, per tenere solamente le informazioni che mi interessano (cioè il genere di appartenenza delle tracce audio del dataset fma_small).

I file ipynb i cui nomi finiscono con “Extraction” sono quelli che, quando eseguiti, analizzano i file mp3 contenuti nella cartella fma_large (scaricabile qui) e producono come output file .csv contenenti le feature.

(Nella cartella di progetto caricata su github non sono presenti i file csv con le feature perché troppo pesanti, quindi per far funzionare i classificatori è necessario scaricare fma_large).

I file ipynb i cui nomi finiscono con “Classifier”, utilizzano i csv delle feature estratte per addestrare i classificatori e testarli.

Il file “ClassifyExternalSong.ipynb” è una demo in grado di classificare le tracce audio mp3 presenti sul pc.

Per quanto riguarda le librerie necessarie, oltre alle classiche librerie utilizzate in progetti di analisi di dati (pandas, numpy, scipy, sklearn, matplotlib) è necessaria librosa, per eseguire elaborazioni sulle tracce audio. È stata necessaria anche l’installazione di ffmpeg, che permette di lavorare con file mp3 invece che wav.

Il file ClassifyExternalSong utilizza la libreria ipyfilechooser che fornisce un widget per la selezione di file.

Genres Classificator

Ricerca dei dati

Per svolgere questo progetto è necessario un dataset contenente un sufficiente numero di tracce musicali. Queste tracce inoltre devono avere assegnato il genere di appartenenza.

Spesso una canzone appartiene a più di un solo genere musicale, per questo progetto si è scelto di tenerne in considerazione solamente uno, quello "prevalente", per canzone.

Ho utilizzato un dataset chiamato FMA (Free Music Archive).

https://github.com/mdeff/fma

Paper: https://arxiv.org/pdf/1612.01840.pdf

In particolare, ho utilizzato un sottoinsieme del dataset chiamato "fma_large": contenente circa 28000 tracce audio mp3 da 30 secondi ciascuna. Questo compromesso diminuisce notevolmente lo spazio di archiviazione richiesto, nell'ipotesi che 30 secondi siano rappresentativi dell'intera canzone.

Il dataset è già sbilanciato.

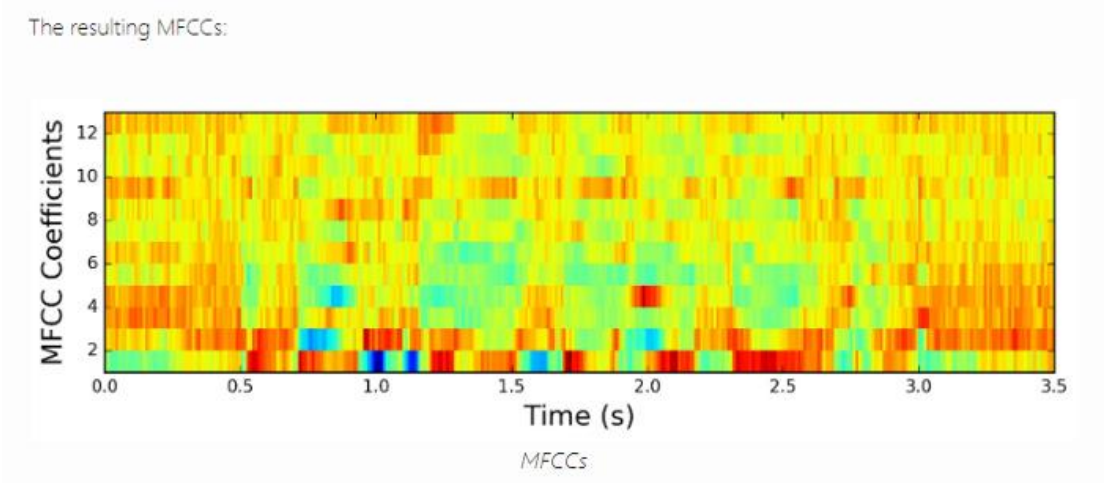
Classi: Electronic, Experimental, Folk, Hip Hop, Instrumental, International, Pop, Rock

Preprocessing e analisi dei dati

Le informazioni più significative che si possono estrarre da tracce musicali hanno a che fare con lo spettrogramma: tutte le feature listate di seguito (ad eccezione di ZCR) vengono calcolate a partire da esso.

MFCC (Mel Frequency Cepstral Coefficients)

Gli MFCC sono di un segnale sono un set di feature che descrivono in generale la forma dello spettrogramma. Essi si sono rivelati essere le feature più influenti nel buon funzionamento del classificatore.



Nel calcolo degli MFCC, viene eseguita la "trasformata di fourier veloce" (FFT), assumendo che l'audio sia un segnale stazionario per brevi periodi di tempo. Per questo esso viene diviso in frame.

Lo spettrogramma risultante viene diviso sull'asse delle ordinate in bande di frequenza, utilizzando la Mel scale: le bande non sono lineari ma sono basate su come l'orecchio umano percepisce i suoni.

Utilizzando mfcc con Librosa:

mfccs = librosa.feature.mfcc(x, sr=sr, n_mfcc=12, hop_length=512, n_fft=2048)

Viene creato un array annidato (matrice) con shape (12, 1292)

Questo perché:

- abbiamo scelto n_mfcc = 12, è il numero di bande di frequenza in cui è suddiviso lo spettrogramma
- ci sono 1292 frame

Con frequenza di campionamento = 22050, in una traccia audio di 30 sec ci sono in totale (22050*30) campioni.

Usando una finestra contenente 2048 campioni (n_ftt) e facendo avanzare la finestra di 512 campioni per passare da ogni frame al successivo (hop_lenght), nella traccia audio ci saranno 1292 frame in totale.

In maniera approssimata:

Numero frame = numero totale campioni / hop_lenght

Vogliamo estrarre informazioni per ogni banda di frequenza del mfcc: non ci interessa il dominio del tempo.

Ad esempio, facciamo la media dell'intensità per ogni banda di frequenza: otterremo un array (vettore) di n_mfcc elementi.

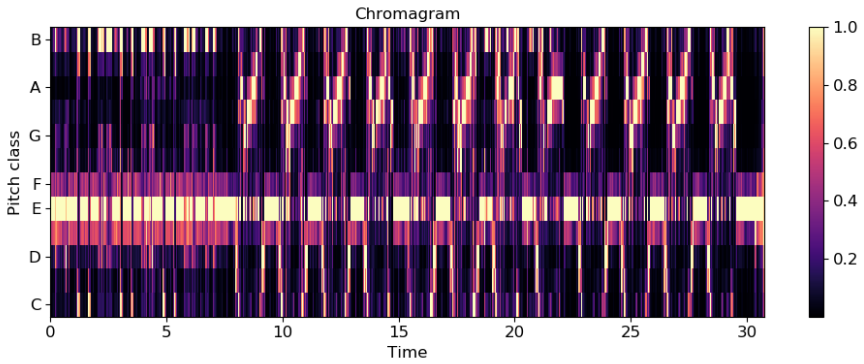
Gli operatori statistici utilizzati sono:

Media, deviazione standard, massimo, minimo, mediana, skewness (misura la simmetria della distribuzione dei valori rispetto alla media), kurtosis (misura il peso delle code rispetto al resto della distribuzione dei valori).

Vengono dati al classificatore i risultati di queste operazioni.

Chroma

Misura l'intensità delle frequenze relative ad ognuna delle 12 note musicali.



La funzione che calcola il chroma crea uno spettrogramma e somma le intensità delle frequenze relative ad ogni nota. Quindi la funzione restituisce un array annidato (matrice) con shape (12, 1292), dove 12 sono le note musicali e 1292 è il numero di frame.

Vengono quindi fatti calcoli statistici sull'intensità di ogni nota: media, deviazione standard, minimo, massimo, mediana, skewness e kurtosis.

ZCR (Zero Crossing Rate)

È una feature che non utilizza lo spettrogramma.

Indica, per ogni frame, il numero di volte in cui la traccia audio ha valore di ampiezza pari a zero. Un valore più elevato indica che la traccia è più percussiva.

Spectral Centroid

Indica, per ogni frame temporale, la frequenza media (media pesata in base all'intensità di ogni frequenza nello spettrogramma).

Utilizzando la funzione di librosa si ottiene un array con numero di elementi pari al numero di frame (ad esempio 1292). Vengono poi utilizzati gli operatori statistici visti sopra, ad eccezione di skewness e kurtosis.

Questi valori aiutano il classificatore dando una vaga idea della frequenza che tende a dominare, e di come essa vari nel tempo.

Spectral Rolloff

Indica un percentile di frequenza, ovvero il valore minimo di frequenza sotto al quale ricade una certa percentuale (di default 85%) dell'intensità totale dello spettrogramma.

Come per lo spectral centroid, si ottiene un valore di rolloff per ogni frame della traccia, ma vengono dati al classificatore solamente la loro media, deviazione standard, massimo, minimo e mediana.

Questa feature aiuta il classificatore mostrando fino a dove si estende la maggior parte di intensità dello spettro, e come il valore limite vari nel tempo.

Spectral Bandwidth

Viene calcolata lo spectral bandwidth di ordine p = 2, che è come una deviazione standard pesata.

$$\left(\sum_k S(k)(f(k) - f_c)^p\right)^{\frac{1}{p}}$$

where $S(k)$ is the spectral magnitude at frequency bin k, $f(k)$ is the frequency at bin k, and f_c is the spectral centroid.

Spectral contrast

Ogni frame dello spettrogramma è suddiviso in sottobande (come per il calcolo di mfcc). Per ogni sottobanda viene calcolato il contrasto, cioè il rapporto tra il quantile superiore (intensità massima) e il quantile inferiore (intensità minima). Quindi un contrasto alto significa che nella sottobanda c'è un segnale chiaro e di banda stretta, mentre un contrasto basso indica un segnale rumoroso su tutta la sottobanda.

Questa feature può aiutare il classificatore: ad esempio, una strumentazione elettronica ha in genere un segnale più chiaro mentre una strumentazione acustica ha un segnale più rumoroso.

Di default ci sono 6 sottobande, per ognuna di queste si fanno media, deviazione standard, massimo, minimo, mediana, skewness e kurtosis.

Tecniche di ML utilizzate

Il DataFrame di feature ottenuto ha 230 colonne.

Ho provato ad aumentare il numero di feature, in particolare aumentando il numero di n_mfcc (numero di bande di frequenza per i mfcc), per avere una migliore rappresentazione dello spettrogramma delle tracce audio. Il risultato è però peggiorato con tutte le tecniche di ML, mostrando un problema di overfitting.

SVM

Sceglie un "decision boundary" massimizzando la distanza tra i punti più vicini appartenenti a classi diverse. Questi punti sono chiamati "support vector".

Nel progetto ci sono più di due classi, viene utilizzato quindi l'approccio OVO (one versus one). Per ogni traccia da classificare, vengono svolti [n_classi * (n_classi-1) / 2] problemi di classificazione binaria (28 nel nostro caso).

Si è scelto l'approccio OVO rispetto a quello OVR (one versus rest) perché, anche se più dispendioso, ha dato risultati migliori.

Logistic regression

A differenza della linear regression, che non viene utilizzata nei classificatori ma per prevedere valori continui, la logistic regression può essere usata.

Anch'essa è una regression perché prevede un valore continuo, in particolare l'output è una probabilità. Impostando un valore limite a questa probabilità (come h(x) > 0.5) si può ottenere un classificatore. In caso di classificatore multiclasse, utilizza gli stessi approcci di SVM, cioè OVO o OVR (in questo progetto ho utilizzato OVO).

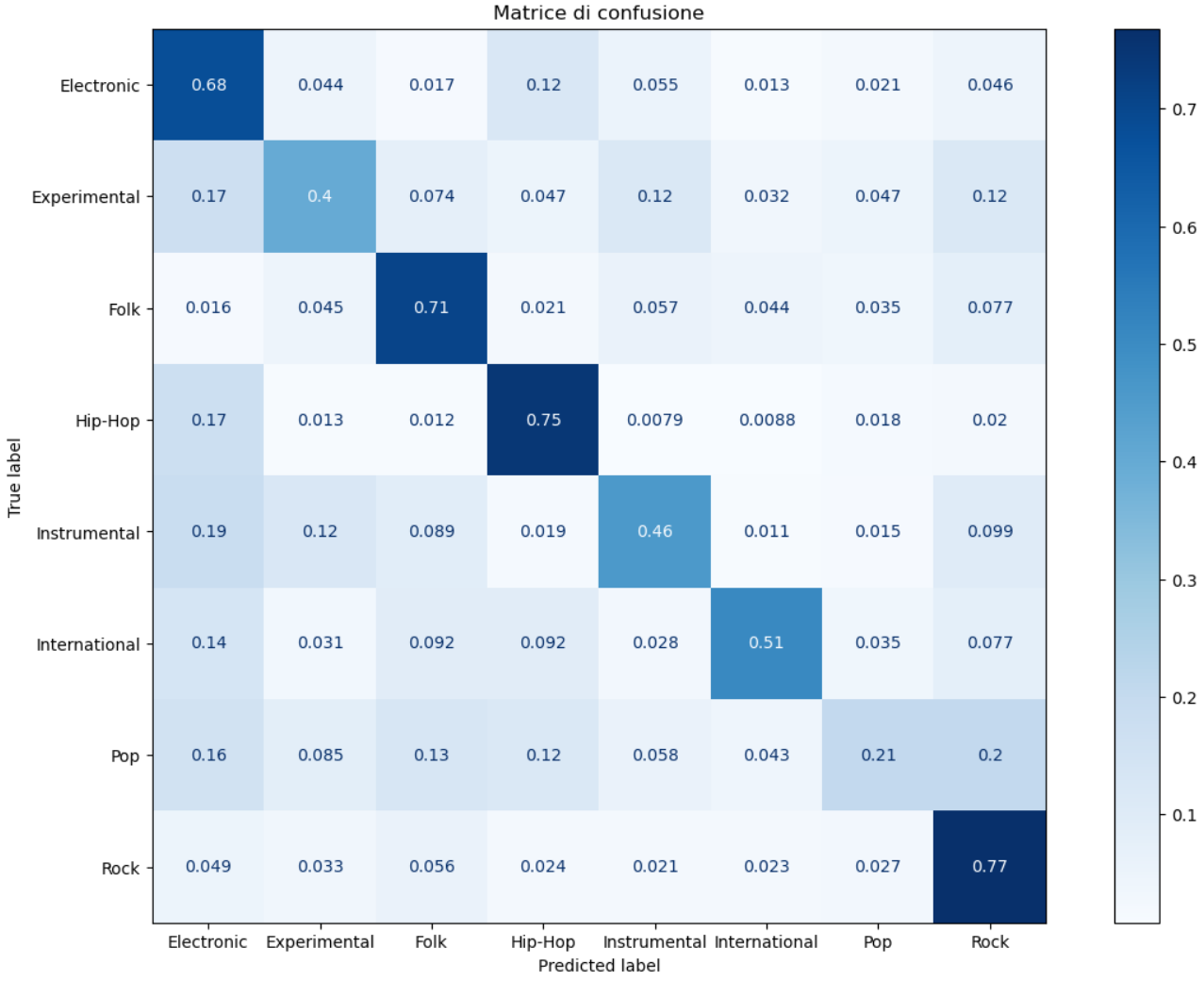
kNN (K-Nearest Neighbor)

Algoritmo per la classificazione. A differenza degli altri due modelli visti prima, è lazy, cioè non costruisce un modello di predizione sulla base dei dati di training in quanto non c'è una fase di training: i dati di training vengono soltanto memorizzati. Essi vengono usati solo quando viene richiesta la classificazione di una traccia.

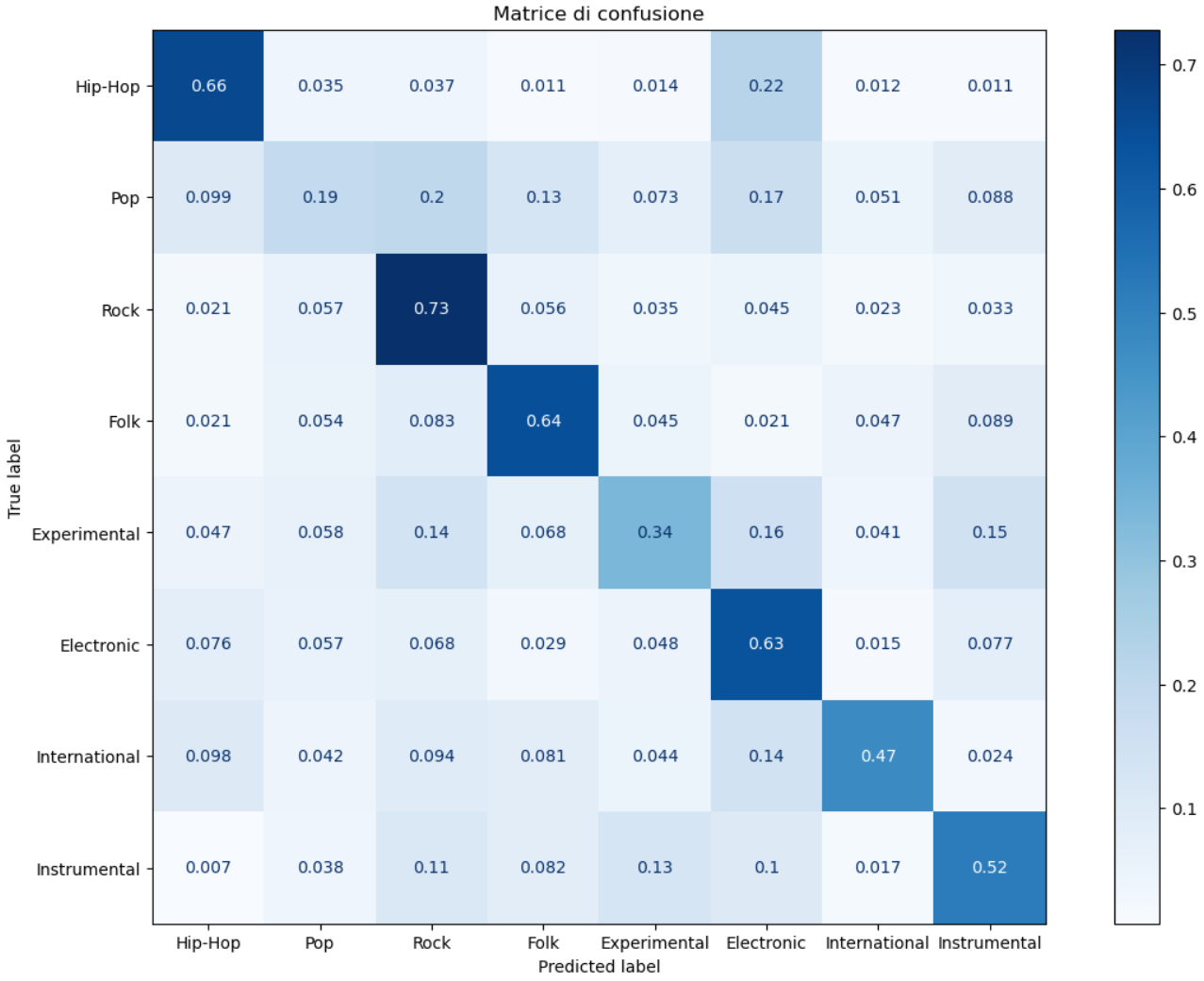
Ho addestrato classificatori kNN variando il valore di k da 1 a 100, per trovare il valore di k che massimizza l'accuracy. Il valore ottimale di k varia in base a come i dati di training e di testing vengono mischiati, un valore che è generalmente buono è k=1.

Risultati

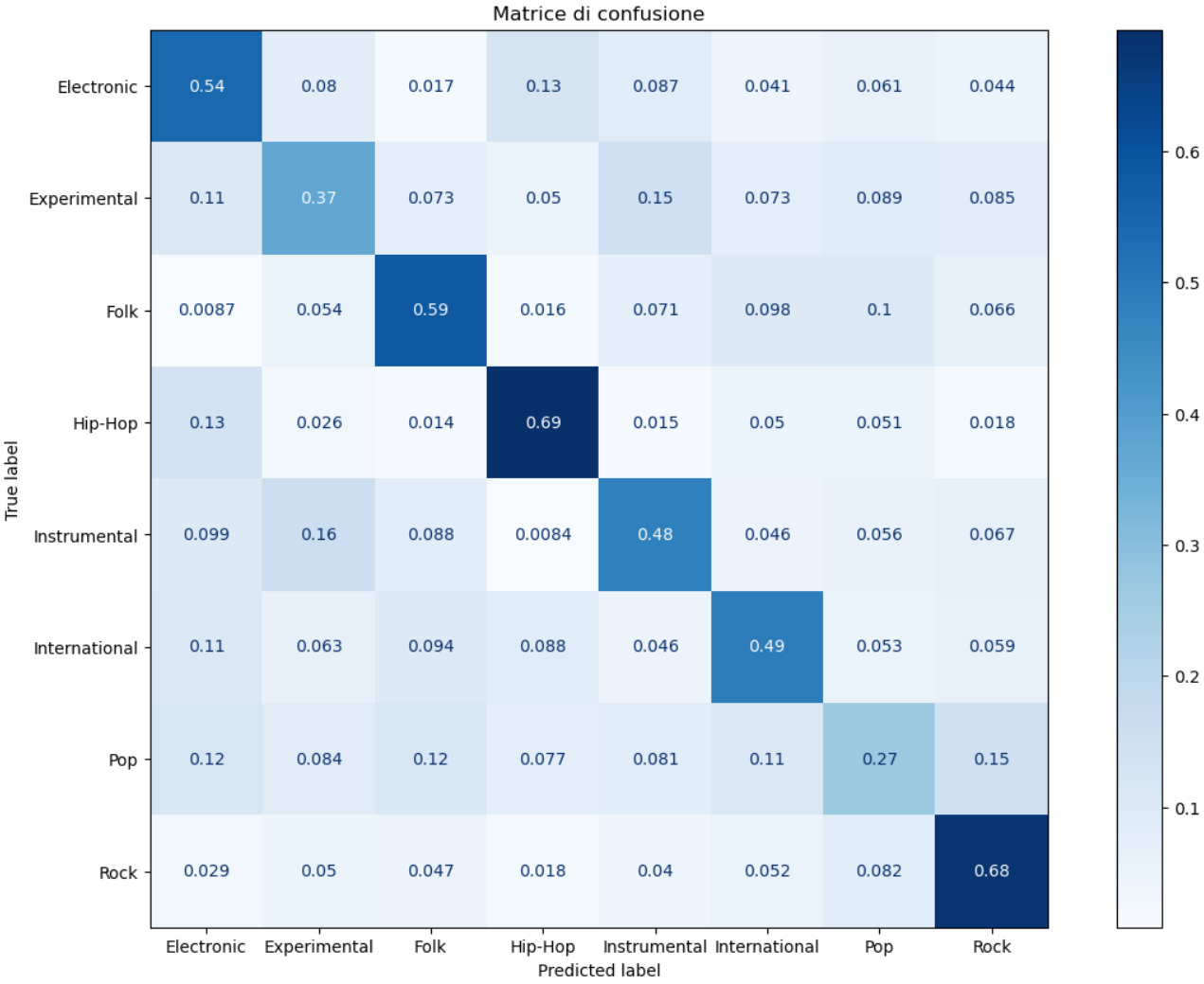
SVM



Random Forest



Logistic Regression



kNN (con 1<k<100 che massimizza l'accuracy, si è trovato k=1)