



UNIVERSITÀ DI PARMA

Dipartimento di Ingegneria e Architettura
Corso di Laurea in Ingegneria dei Sistemi Informativi

Analisi e classificazione automatica di generi musicali tramite machine learning

Analysis and automatic classification of musical genres using
machine learning

Relatore:

Prof. Michele Tomaiuolo

Correlatore:

Prof. Giulio Angiani

Tesi di Laurea di:
Luca Bonelli

ANNO ACCADEMICO 2019-2020

Ai miei genitori.

Ringraziamenti

Ringrazio innanzitutto il Prof. Tomaiuolo ed il Prof. Angiani per il sostegno e per avermi spinto a continuare lo sviluppo di un progetto di mio interesse, che ha ora preso forma in questa tesi.

Voglio ringraziare la mia famiglia, che è sempre molto unita: grazie a mio papà, che è il miglior ingegnere (ad honorem) che conosca; a mia mamma, che più di tutti mi ha supportato negli ultimi 21 anni; a Silvia e Ste, che mi hanno viziato fin da piccolo, e che mi hanno reso zio di Mattia e Celeste: ora ho qualcuno con cui giocare a Super Mario. Grazie ad Ale, che più che un cugino è un fratello. Grazie a mio zio Claudio per le cene a casa sua e per avermi, forse, passato qualche gene di musicista. Grazie ai nonni Rina, Armando e Franca, che hanno costruito tutto questo. Grazie a tutti gli altri membri della mia famiglia, parenti di sangue e acquisiti, mi accorgo ora che siamo in troppi.

Voglio poi ringraziare tutti gli amici che mi sono vicini, anche in tempi di pandemia. Grazie al gruppo di amici di Roccabianca, tra cui Anichini, i Botta, Nizzo, Nicco, Matte e tutti gli altri. Con voi si sta bene anche quando non si fa niente. Bastano due chiacchiere in mezzo alla nebbia in piazza.

Grazie al gruppo che si è formato in università: Lorenzo, Riccardo, Cosimo, Mario, Adin, Luca. Siete il motivo per cui le lezioni in presenza sono le migliori.

Grazie agli amici dai tempi delle superiori: Leo, Damian, Carra, Ippo, Seba, Rubens e tutti gli altri.

Non vedo l'ora di poter festeggiare con tutti voi.

Indice

Introduzione	1
1 Stato dell'arte	2
1.1 Music Information Retrieval	2
1.2 Machine Learning	4
1.3 Problema di classificazione	4
1.3.1 Classificazione supervisionata	5
1.3.2 Support Vector Machine	6
1.3.3 Logistic Regression	8
1.3.4 k-Nearest Neighbors	9
1.3.5 Random Forest	10
1.4 Python	11
1.5 Free Music Archive	13
2 Selezione di dati	15
2.1 Selezione dei dati	15
2.1.1 Attenuazione dello sbilanciamento delle classi	18
3 Preprocessing audio	21
3.1 Ricampionamento	21
3.2 Short-time Fourier transform	22
3.2.1 Creazione della Short-time Fourier transform con Librosa	25

4 Estrazione di feature	27
4.1 MFCC (Mel Frequency Cepstral Coefficients)	27
4.1.1 Calcolo degli MFCC	28
4.1.2 Estrazione di feature dagli MFCC	30
4.2 Chroma Feature	32
4.2.1 Creazione del cromagramma	33
4.2.2 Estrazione di feature dal cromagramma	34
4.3 Zero-crossing rate	35
4.3.1 Calcolo dei zero-crossing rate	35
4.3.2 Estrazione di feature dai ZCR	36
4.4 Spectral Centroid	36
4.4.1 Calcolo dello spectral centroid	37
4.4.2 Estrazione di feature dallo spectral centroid	38
4.5 Spectral Rolloff	38
4.5.1 Estrazione di feature dallo spectral rolloff	38
4.6 Spectral Bandwidth	39
4.6.1 Calcolo della spectral bandwidth	39
4.6.2 Estrazione di feature dallo spectral bandwidth	39
4.7 Spectral Contrast	40
4.7.1 Estrazione di feature dallo spectral contrast	41
5 Addestramento dei classificatori	43
5.1 Preparazione del dataset	43
5.1.1 Normalizzazione del dataset	44
5.1.2 Oversampling del training set	45
5.2 Creazione dei classificatori	46
5.3 Feature selection	47
6 Risultati	51
6.1 Risultati della classificazione	51
6.1.1 Osservazioni	52
6.1.2 Risultati con SVM	53

6.1.3	Risultati con Logistic Regression	54
6.1.4	Risultati con k-nearest neighbors	55
6.1.5	Risultati con Random Forest	56
6.2	Visualizzazione grafica del dataset in uno spazio a due dimensioni (embedding)	57
	Conclusioni	60
	Bibliografia	61

Introduzione

L’industria musicale è radicalmente cambiata negli ultimi decenni a causa della rivoluzione digitale: le nuove piattaforme hanno permesso ai consumatori un accesso facilitato ad una mole enorme di tracce musicali.

Sempre più spesso, gli artisti utilizzano piattaforme di streaming come Spotify, SoundCloud o Youtube per pubblicare i propri contenuti, senza aver necessariamente bisogno di etichette discografiche per la distribuzione. Tutte queste piattaforme si trovano quindi oggi a dover gestire miliardi di tracce. Un compito fondamentale di cui queste piattaforme si devono occupare è la categorizzazione delle tracce, che ha come obiettivo l’ottenimento di un archivio maggiormente documentato e facilmente navigabile da parte degli utenti. Questo compito non si limita alla conoscenza, per ogni traccia, di artista, album e data di pubblicazione. È anche necessario sapere quali tracce possano essere considerate simili ad altre: questo aiuterebbe gli utenti nel scegliere canzoni che rispecchino i propri gusti. Una caratteristica in grado di descrivere la similarità tra canzoni è il genere musicale di appartenenza.

L’obiettivo di questa tesi è capire se sia possibile stabilire in maniera automatica il genere musicale di canzoni, quando esso non si conosce a priori. Per fare questo si estrarranno informazioni di basso livello dall’audio, facendo analisi di tipo fisico sul segnale, e si utilizzeranno tecniche di machine learning per la decisione del genere musicale da assegnare. Si avrà così modo di verificare se caratteristiche oggettive e misurabili siano correlate al genere musicale, ovvero una caratteristica ad alto livello di astrazione e non descrivibile in maniera scientifica.

Capitolo 1

Stato dell'arte

1.1 Music Information Retrieval

Music Information Retrieval, spesso abbreviato in MIR, è un campo di ricerca che utilizza psicoacustica, teoria dei segnali, informatica e machine learning. [1] MIR si dedica a:

- estrazione di informazioni da segnali audio, rappresentazioni simboliche o fonti esterne, come pagine web
- indicizzazione della musica utilizzando queste informazioni
- sviluppo di diversi schemi di ricerca e recupero (ad esempio, ricerche basate sul contenuto, sistemi di raccomandazione musicale o interfacce utente per la navigazione in grandi raccolte musicali)

È importante notare come nell'ambito del MIR si ponga l'attenzione sulla percezione che le persone hanno della musica: viene infatti utilizzata la psicoacustica, insieme alla teoria dei segnali, per ottenere informazioni significative per gli utenti.

Esistono diversi tipi di compiti pratici tra gli obiettivi del MIR [2]:

- Ricerca di tracce: ha lo scopo di aiutare gli utenti a trovare musica da grandi database sulla base di criteri di somiglianza. Un esempio

di applicazione che fa questo è Shazam: da un frammento di audio può riconoscere l'intera traccia. Altri tipi di applicazioni fanno ricerche sulla base di filtri semanticci: un utente può cercare una canzone descrivendone le caratteristiche con parole chiave.

- Raccomandazione di tracce: vengono proposte liste di canzoni, scelte in base alle preferenze dell'utente. Spotify è una delle tante applicazioni che utilizzano sistemi di raccomandazione. La scelta delle canzoni da raccomandare deve essere accurata e trasparente verso l'utente.
- Generazione di playlist: ha molte similarità con la generazione di raccomandazioni. In questo caso è importante anche l'ordine in cui le canzoni vengono presentate, e la lista di tracce deve anche essere diversificata, per non annoiare l'utente con canzoni troppo simili.



Figura 1.1: Logo di Shazam



Figura 1.2: Logo di Spotify

1.2 Machine Learning

Machine Learning, o apprendimento automatico, è una branca dell'intelligenza artificiale. [3] L'obiettivo principe dell'apprendimento automatico è che una macchina sia in grado di generalizzare dalla propria esperienza, ossia che sia in grado di svolgere ragionamenti induttivi. In questo contesto, per generalizzazione si intende l'abilità di una macchina di portare a termine in maniera accurata esempi o compiti nuovi, che non ha mai affrontato, dopo aver fatto esperienza su un insieme di dati di apprendimento.

1.3 Problema di classificazione

La classificazione è un problema tipico nel machine learning. In un problema di classificazione viene utilizzato un algoritmo, in grado di assegnare una categoria (detta classe) ad ogni osservazione che gli viene presentata. L'algoritmo di classificazione è detto classificatore.

Il classificatore deve essere preventivamente addestrato con un set di dati composto un numero elevato di osservazioni, chiamate anche istanze o campioni. Ogni istanza del set di dati è rappresentata utilizzando lo stesso insieme di caratteristiche, chiamate feature, che possono essere valori numerici continui o discreti.

Se la classe a cui le istanze del set di training appartengono sono note, l'apprendimento è chiamato supervisionato, altrimenti si dice che l'apprendimento è non supervisionato.

Lo scopo di applicare algoritmi per la classificazione non supervisionata (detti algoritmi di clustering) è di scoprire classi non conosciute a priori sulla base di pattern presenti sui dati di addestramento.

L'obiettivo dell'apprendimento supervisionato è costruire un modello conciso della distribuzione delle classi in relazione alle feature del set di training. Una volta ottenuto, il modello può essere utilizzato per assegnare classi ad altre istanze in cui i valori delle feature sono noti, ma la classe di appartenenza è sconosciuta. [4]

1.3.1 Classificazione supervisionata

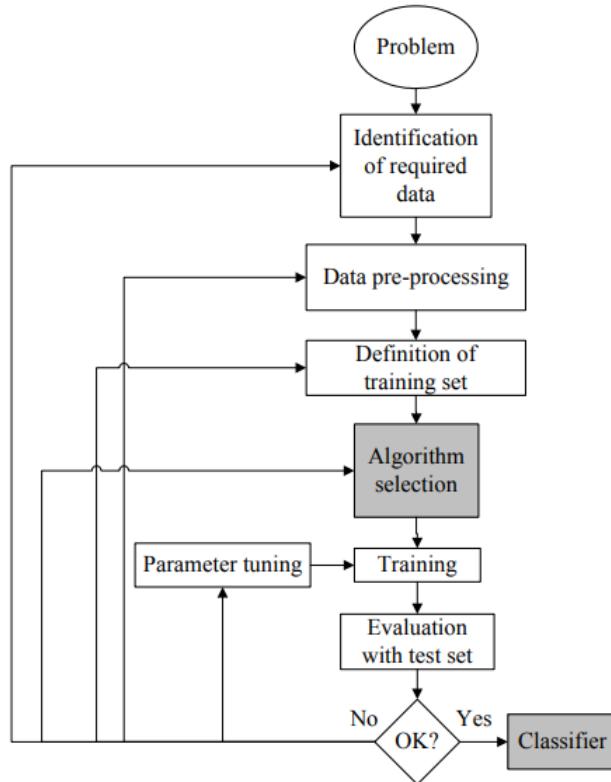


Figura 1.3: Processo per l'applicazione della classificazione supervisionata a problemi reali

Il primo passo verso la creazione di un classificatore per problemi reali è la raccolta del set di dati, che verrà utilizzato per l'addestramento e per la valutazione delle prestazioni del classificatore. Per scegliere quali attributi e caratteristiche prendere in considerazione, è in genere necessario avere buone conoscenze nell'ambito del problema che si vuole affrontare, è quindi consigliabile consultare un esperto del settore. Altrimenti, il metodo più semplice è quello della "forza bruta", ovvero includere tutto ciò che è disponibile nella speranza che le caratteristiche giuste (informative, rilevanti) possano essere isolate. Questo metodo porterà a risultati peggiori, in quanto c'è un forte rischio di includere rumore ed escludere alcune caratteristiche significative.

La valutazione di un classificatore preventivamente addestrato è spesso basata sul valore di accuratezza delle previsioni effettuate su di un set di test, calcolata come la percentuale di previsioni corrette diviso il numero totale di previsioni. Una tecnica è dividere il dataset utilizzandone due terzi per il training ed un terzo per la stima delle prestazioni. Con un'altra tecnica, chiamata cross-validation, il dataset è suddiviso in subset mutualmente esclusivi e di dimensioni uguali. A turno, per ogni subset il classificatore viene addestrato con l'unione di tutti gli altri subset. Il subset non utilizzato nel training viene utilizzato per la valutazione. La media dell'accuratezza ottenuta con ogni subset è una stima dell'accuratezza del classificatore.

1.3.2 Support Vector Machine

SVM è un algoritmo di apprendimento automatico supervisionato che può essere utilizzato sia per scopi di classificazione che di regressione. [5]

Le istanze del set di training vengono considerate punti in uno spazio n-dimensionale (\mathbb{R}^n), dove n è il numero di feature. L'algoritmo SVM definisce un “decision boundary”, ovvero un iperpiano che separa le istanze di classi diverse. L'iperpiano viene scelto massimizzando la distanza tra esso e le istanze di classi diverse. Le istanze più vicine all'iperpiano sono chiamate “support vectors”.

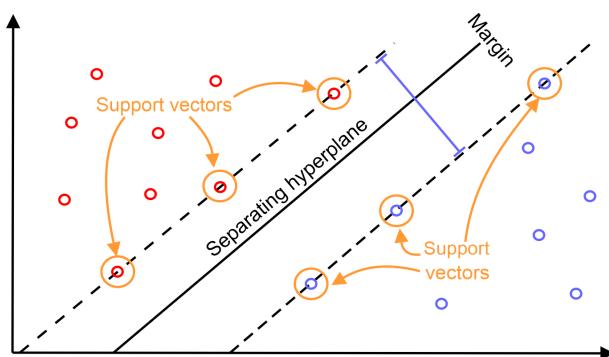


Figura 1.4: SVM lineare su dati a due dimensioni

Se non è possibile trovare un iperpiano che possa suddividere le classi in maniera efficace, si dice che il dataset non è linearmente suddividibile. Si utilizza quindi una funzione, detta kernel, che mappa le istanze del dataset in punti appartenenti ad uno spazio avente un numero di dimensioni superiore al numero di feature.

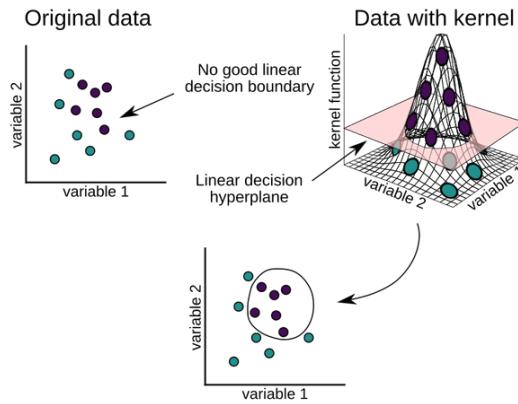


Figura 1.5: Applicazione di una funzione di kernel su dati non linearmente separabili

Esistono diverse funzioni kernel:

Kernel polinomiale, in cui un valore elevato di d renderà il limite decisionale più complesso.

$$k(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j)^d$$

Kernel RBF (Radial Basis Function), anche chiamato il kernel gaussiano. Utilizza un metodo di mappatura più complesso.

$$k(\vec{x}_i, \vec{x}_j) = \exp(-\gamma \|\vec{x}_i - \vec{x}_j\|^2)$$

SVM è un algoritmo per la classificazione binaria, in cui sono presenti solamente due classi, ma può essere adattato per problemi di classificazione multiclasse. Con l'approccio OVR (one versus the rest), per ogni classificazione vengono affrontati n_classi problemi di classificazione binaria.

Con l'approccio OVO (one versus one), per ogni traccia da classificare, vengono affrontati $\frac{n_classi(n_classi-1)}{2}$ problemi di classificazione binaria.

1.3.3 Logistic Regression

La regressione logistica è un metodo di classificazione rientrante nella famiglia degli algoritmi di apprendimento supervisionato. [6] Avvalendosi di metodi statistici, la regressione logistica permette di generare un risultato che rappresenta una probabilità che un dato valore di ingresso appartenga a una determinata classe.

Nei problemi di regressione logistica binomiale, la probabilità che l'output appartenga ad una classe sarà P , mentre che appartenga all'altra classe sarà $1-P$ (dove P è un numero compreso tra 0 e 1 perché esprime una probabilità). Impostando un valore limite a questa probabilità, come $P > 0.5$, si può ottenere un classificatore.

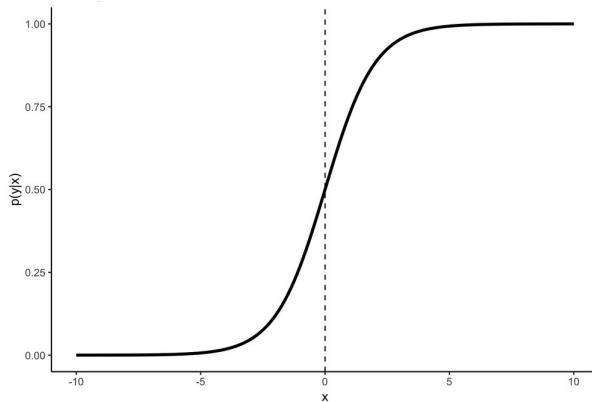


Figura 1.6: Funzione logistica

Le probabilità che un campione appartenga ad una classe vengono calcolate utilizzando la seguente formula, ricavata dalla funzione logistica e dal teorema di Bayes:

$$\mathbb{E}[Y | \mathbf{X}] = Pr(Y = 1 | X_1, \dots, X_k) == \frac{e^{\beta_0 + \beta_1 X_1 + \dots + \beta_k X_k}}{1 + e^{\beta_0 + \beta_1 X_1 + \dots + \beta_k X_k}} = p$$

Dove $\boldsymbol{\beta}$ è il vettore dei parametri β_0, \dots, β_k . \mathbf{X} è il vettore di variabili indipendenti X_1, \dots, X_k , ovvero le feature. Y è la variabile dipendente, che identifica la classe

Il vettore β è di norma stimato con il metodo della massima verosimiglianza, con il quale si ottengono stimatori efficienti, consistenti e distribuiti normalmente nel caso in cui il campione statistico sia abbastanza grande. [7]

Per utilizzare la regressione logistica in un problema di classificazione multiclassa, vengono utilizzati gli stessi approcci descritti con SVM, cioè OVO o OVR.

1.3.4 k-Nearest Neighbors

Il k-nearest neighbors (k-NN) è un algoritmo utilizzato nel riconoscimento di pattern per la classificazione di oggetti basandosi sulle caratteristiche degli oggetti vicini a quello considerato. [8] Nella classificazione k-NN, tutti i dati sono rappresentati come punti in uno spazio multidimensionale. Un oggetto è classificato da un voto di pluralità dei suoi vicini: viene assegnato alla classe più comune tra i k oggetti a lui più vicini (k è un numero intero positivo, tipicamente piccolo). Se $k = 1$, l'oggetto viene semplicemente assegnato alla classe di quel singolo vicino più prossimo.

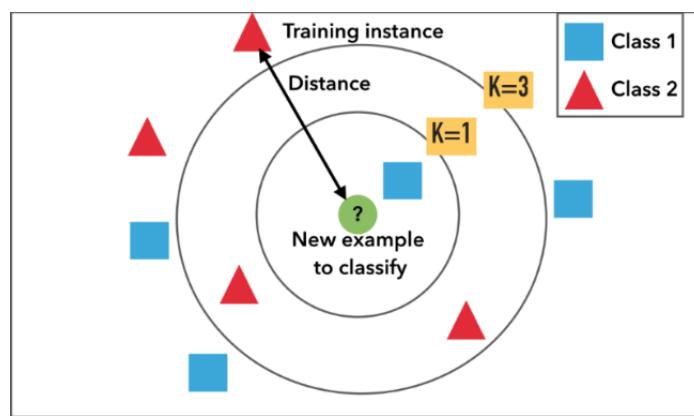


Figura 1.7: Classificazione con algoritmo kNN

A differenza degli altri algoritmi trattati in questa tesi, kNN viene definito "lazy": non costruisce un modello di predizione sulla base dei dati di training, essi vengono soltanto memorizzati. Nella fase di classificazione vengono calcolate le distanze tra l'istanza da classificare e i dati di training. Di

solito viene usata la distanza euclidea, ma anche altri tipi di distanza sono ugualmente utilizzabili, ad esempio la distanza Manhattan. Un punto è assegnato alla classe C se questa è la più frequente fra i k esempi più vicini all'oggetto sotto esame.

1.3.5 Random Forest

Una foresta casuale combina molti alberi decisionali in un unico modello. Individualmente, le previsioni fatte dagli alberi decisionali potrebbero non essere accurate, ma combinate insieme, le previsioni saranno in media più vicine al risultato. [9]

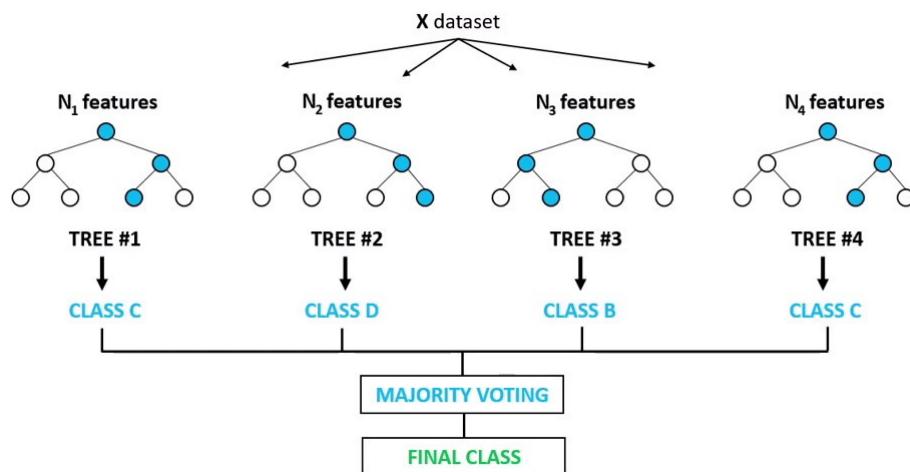


Figura 1.8: Classificatore Random Forest

Per fare questo, viene utilizzato il bagging, una tecnica in grado di combinare le previsioni di più algoritmi di apprendimento automatico per fare previsioni più accurate rispetto a qualsiasi singolo modello.

Il bagging rappresenta una procedura generale che può essere utilizzata per ridurre la varianza di quegli algoritmi che hanno una varianza elevata. In questo caso, viene utilizzato sugli alberi decisionali.

Ogni albero decisionale impara da un set casuale di istanze prese dai dati di training. In ciascun albero decisionale viene considerato solo un sottinsieme di tutte le caratteristiche per suddividere ciascun nodo.

Il risultato finale restituito dal Random Forest quando viene applicato ad un oggetto è la classe restituita dal maggior numero di alberi.

1.4 Python



Figura 1.9: Logo di Python

Python è un linguaggio di programmazione interpretato e orientato a oggetti. [10] È considerato di alto livello perché caratterizzato da una grande astrazione dai dettagli del funzionamento del calcolatore e dal linguaggio macchina, ed è quindi più vicino alla logica umana. Python ha tra i principali obiettivi dinamicità, semplicità e flessibilità. La velocità di esecuzione non è un punto di forza, ma diverse librerie tra quelle più utilizzate hanno parti di codice scritte nei linguaggi C o C++, per aumentare le performance.

È uno dei linguaggi di scripting più utilizzati nell'ambito del data science e data analysis. Esistono per questo scopo diverse librerie:

- NumPy è una libreria che offre concetti di vettorializzazione e indicizzazione più versatili e completi rispetto a quelli contenuti nelle librerie standard di Python. I vettori N-dimensionali di numpy sono un tipo di dato utilizzato molto comunemente. Inoltre offre funzioni matematiche e relative alla generazione di numeri casuali. NumPy ha diverse parti di codice scritte in linguaggio C, per rendere più veloce e performante il suo utilizzo.

- SciPy offre routine per integrazione numerica, interpolazione, ottimizzazione, algebra lineare e statistica.
- Scikit-learn è una libreria di apprendimento automatico (machine learning) che contiene algoritmi di classificazione, regressione, clustering, regressione logistica, classificatore bayesiano, k-mean e DBSCAN.
- Pandas è uno strumento per la manipolazione di dati ad alto livello. Offre una struttura chiamata DataFrame, utile per la gestione dei dataset, che vengono memorizzati in forma tabulare, in cui solitamente le righe sono osservazioni e le colonne sono feature.
- Seaborn è una libreria per la visualizzazione di dati, basata su matplotlib
- Librosa è una libreria python che offre strumenti per l'analisi di audio.



Figura 1.10: Logo di Librosa

1.5 Free Music Archive

Free Music Archive [11] è un dataset aperto e accessibile, che contiene 106574 brani musicali in formato audio MP3, creati da 16341 artisti.



Figura 1.11: Logo di Free Music Archive

Questo dataset è adatto all'elaborazione dei diversi compiti nell'ambito del MIR (Music Information Retrieval). Le tracce audio sono ricche di metadati, che includono informazioni sulla traccia, sull'artista e sull'album di appartenenza. I metadati di ogni traccia vengono inseriti dall'artista stesso quando egli pubblica il file MP3 nell'archivio.

100% track_id	100% title	93% number
2% information	14% language_code	100% license
4% composer	1% publisher	1% lyricist
98% genres	98% genres_all	47% genre_top
100% duration	100% bit_rate	100% interest
100% #listens	2% #comments	61% #favorites
100% date_created	6% date_recorded	22% tags
100% album_id	100% title	
94% type	96% #tracks	
76% information	16% engineer	18% producer
97% #listens	12% #comments	38% #favorites
97% date_created	64% date_released	18% tags
100% artist_id	100% name	25% members
38% bio	5% associated_labels	
43% website	2% wikipedia_page	
37% location	5% related_projects	
11% #comments	23% longitude	23% latitude
99% date_created	48% #favorites	10% tags ¹
	8% active_year_begin	
	2% active_year_end	

Figura 1.12: Metadati disponibili con Free Music Archive. Le percentuali indicano la quantità di brani per cui il dato è disponibile

Sono presenti 161 possibili generi musicali.

Essi sono organizzati in una struttura ad albero che definisce 16 generi base (le radici dell'albero) e 145 sottogeneri.

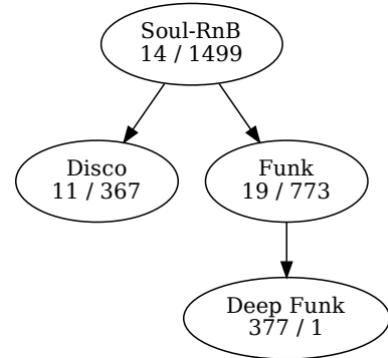


Figura 1.13: Organizzazione del genere base Soul-RnB e dei suoi sottogeneri

Capitolo 2

Selezione di dati

2.1 Selezione dei dati

Per lo sviluppo del progetto è necessario avere un buon numero di tracce audio in file MP3 di cui si conosce il genere musicale di appartenenza.

Free Music Archive permette il download di `fma_large.zip`, un archivio contenente 30 secondi estratti da ogni canzone presente sulla piattaforma. La lunghezza ridotta dei file permette di dover utilizzare meno spazio di archiviazione e ridurrà il tempo necessario ad eseguire l'analisi audio delle tracce. I 30 secondi sono estratti dalla parte centrale di ogni canzone, si presuppone che essi possano essere rappresentativi dell'intera traccia. Ogni canzone ha assegnato un ID univoco, con cui sono denominati i file MP3.

 000002	3	Food	AWOL	AWOL - A Way Of Life
 000003	4	Electric Ave	AWOL	AWOL - A Way Of Life
 000005	6	This World	AWOL	AWOL - A Way Of Life
 000010	1	Freeway	Kurt Vile	Constant Hitmaker
 000020	3	Spiritual Level	Nicky Cook and C...	Niris
 000026	4	Where is your Love?	Nicky Cook and C...	Niris
 000030	5	Too Happy	Nicky Cook and C...	Niris

Figura 2.1: Estratto di contenuto del dataset `fma_large`. Ogni file è denominato con un codice univoco

All'interno dell'archivio `fma_metadata.zip` è presente il file `tracks.csv`, contenente i metadati relativi a ogni traccia.

L'unica informazione necessaria per lo sviluppo di questo progetto riguarda il genere musicale di appartenenza. Esistono due attributi inerenti ad esso all'interno dei metadati:

- L'attributo "genres" specifica, per ogni traccia, uno o più generi musicali di appartenenza tra i 161 generi possibili. Esiste quindi una relazione molti a molti tra le tracce ed i generi: una traccia può appartenere a più di un genere musicale, e, ovviamente, ogni genere musicale è assegnato a più tracce.
- L'attributo "genre_top" specifica, per ogni traccia, un solo genere musicale tra i 16 considerati "generi base". In questo caso, esiste una relazione uno a molti tra le tracce e i generi base: una traccia può appartenere ad un solo genere base, ed ogni genere base può essere ovviamente assegnato a più tracce.

Siccome lo scopo del progetto è quello di creare un classificatore che assegna un solo genere musicale ad una traccia, si è scelto di utilizzare esclusivamente l'attributo "genre_top".

```
#Leggo il file tracks.csv, presente tra i metadati di FMA
path = '/fma_metadata/tracks.csv'
tracks = pd.read_csv(path, index_col=0, header=[0, 1])
#Mi interessa avere l'id della traccia e il rispettivo genere
#musicale base
tracks_genres = tracks.loc[tracks.index, ('track', 'genre_top')]
#Elimino le righe riguardanti le tracce che non hanno l'
#attributo genre_top
tracks_genres = tracks_genres.dropna()
```

Questo porta a due conseguenze:

- si prendono in considerazione soltanto 16 generi musicali

- siccome l'attributo genre_top è presente solamente in circa il 47% delle tracce, si passa dalle 106574 tracce dell'intero archivio di FMA a 49598 tracce utilizzabili.

Osservando la distribuzione delle 49598 tracce nei 16 generi base, si nota che ci sono grandi differenze in termini di quantità.

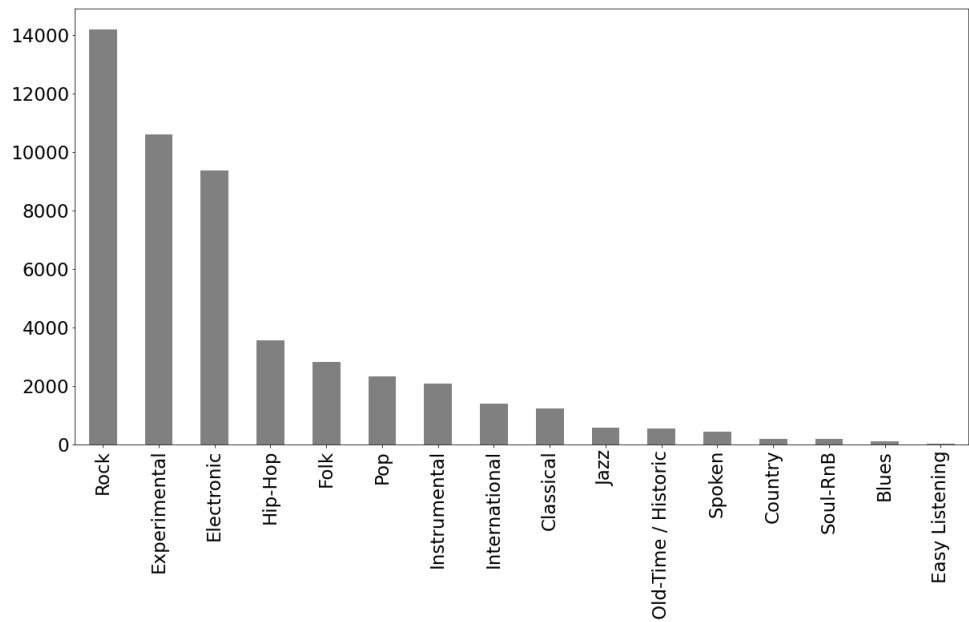


Figura 2.2: Numero di tracce per top-genre nel dataset fma.large

Alcuni di questi generi non hanno un numero di tracce sufficienti alla loro rappresentazione. Si sono quindi scelti di prendere in considerazione soltanto gli 8 generi musicali base aventi il maggior numero di tracce disponibili:

- Electronic
- Experimental
- Folk
- Hip Hop
- Instrumental

- International
- Pop
- Rock

```
wanted_genres = np.array(["Hip-Hop", "Rock", "International",
                           "Pop", "Experimental", "Folk", "Instrumental", "Electronic"])
#Tengo solo le tracce appartenenti agli 8 generi musicali che
#mi interessano
tracks_genres['is_wanted_genre'] = tracks_genres.genre_top.
    isin(wanted_genres)
tracks_genres = tracks_genres[tracks_genres.is_wanted_genre
    == True]
tracks_genres = tracks_genres.drop(columns='is_wanted_genre')
```

2.1.1 Attenuazione dello sbilanciamento delle classi

I generi musicali scelti per l'utilizzo hanno un numero di tracce molto vario. Nel linguaggio utilizzato in ambito di Machine Learning, si dice che le classi, in questo caso i generi musicali, sono sbilanciate. Questa è una caratteristica non gradita quando si vuole affrontare un problema di classificazione: un classificatore che si addestra su campioni appartenenti a classi sbilanciate tenderà a non riconoscere a dovere le caratteristiche delle classi meno rappresentate.

Per risolvere questo problema si sono utilizzati diversi metodi di bilanciamento, che verranno trattati successivamente.

In fase di selezione dei dati si è cercato comunque di attenuare la differenza nel numero di tracce. Per le tre classi più popolose, in particolare Rock, Experimental e Electronic, si è ridotto il numero di tracce utilizzate. FMA definisce nei metadati alcune tracce "più rappresentative del proprio genere musicale". Utilizzando solo queste tracce si è eseguito una sorta di "undersampling deterministico", dato dalla conoscenza a priori dei campioni meno significativi, che vengono così esclusi. Le tracce definite più rappresentative

sono state selezionate dai creatori di FMA e fanno parte di un dataset chiamato fma_medium, in cui è presente un sottoinsieme delle tracce del dataset fma_large utilizzato finora.

```
#Estraggo la lista di canzoni appartenenti ai generi
    electronic, experimental e rock dal medium set
medium_dataset_tracks = tracks[tracks['set', 'subset'] <= ,
    'medium']
medium_dataset_genres = medium.loc[medium.index, ('track', ,
    'genre_top')]
electronic_songs_df = medium_dataset_genres.loc[
    medium_dataset_genres['genre_top'] == "Electronic"]
experimental_songs_df = medium_dataset_genres.loc[
    medium_dataset_genres['genre_top'] == "Experimental"]
rock_songs_df = medium_dataset_genres.loc[
    medium_dataset_genres['genre_top'] == "Rock"]
```

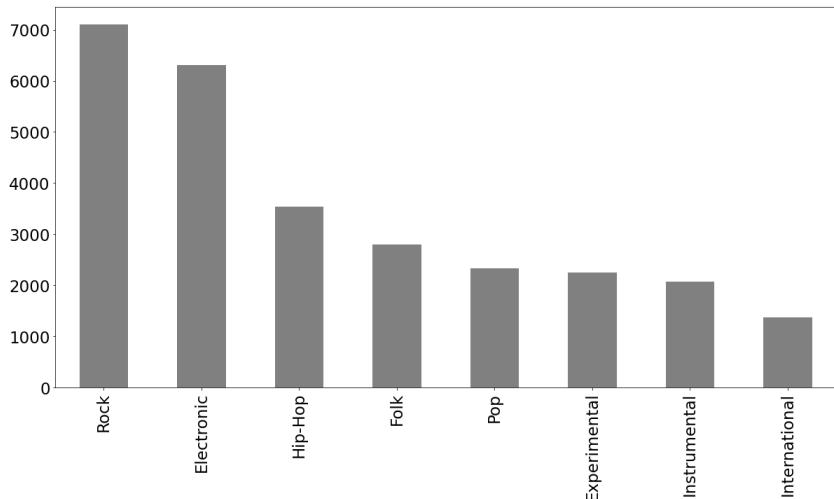


Figura 2.3: Numero di tracce per genere nel dataset definitivo

Il dataset definitivo, su cui si faranno analisi, contiene 27789 tracce audio. Si è dunque creato un file in formato csv contenente i codici delle tracce scelte ed i relativi generi musicali, che verrà utilizzato in seguito.

	A
1	track_id,genre
2	2,Hip-Hop
3	3,Hip-Hop
4	5,Hip-Hop
5	10,Pop
6	134,Hip-Hop
7	136,Rock
8	139,Folk
9	140,Folk
10	141,Folk
11	142,Folk
12	148,Experimental

Figura 2.4: File tracks_genres.csv contenente i codici identificativi delle tracce scelte per il progetto ed i relativi generi musicali di appartenenza, estratti dai metadati di FMA

Capitolo 3

Preprocessing audio

È possibile accedere ai file audio scelti nella fase di selezione dei dati e presenti in fma_large in questo modo, utilizzando il file .csv precedentemente creato:

```
audio_dir = '/fma_large'  
tracks_genres = pd.read_csv('tracks_genres.csv', index_col =  
    0, header = [0])  
for track_id in tracks_genres.index:  
    #Nome del file audio da caricare  
    tid_str = '{:06d}'.format(track_id)  
    filename = os.path.join(audio_dir, tid_str[:3], tid_str +  
        '.mp3')  
    #...  
    #Analisi sull'audio del file "filename"  
    #...
```

Il ciclo for mostrato nell'estratto di codice scorre tutte le tracce scelte per il progetto, per cui si vogliono eseguire analisi audio. La variabile filename verrà utilizzata per aprire i file MP3.

3.1 Ricampionamento

I file MP3 utilizzano una frequenza di campionamento di 44100 Hz, inoltre i file utilizzati in questo progetto hanno un bit rate di 192 Kbps.

Per avere un risparmio in termini di memoria e tempi di esecuzione, i file audio vengono ricampionati ad una frequenza di 22050 Hz.

```
sr = 22050
x, sr = librosa.load(filename, sr=sr)
```

Il teorema del campionamento di Nyquist-Shannon dice che la minima frequenza di campionamento necessaria per evitare ambiguità nella ricostruzione di un segnale con una determinata larghezza di banda è pari al doppio della sua massima componente in frequenza. [12]

$$f_{campionamento} > 2f_{max}$$

Un file audio campionato a 22050 Hz perde quindi informazioni sulle frequenze maggiori a 11025 Hz.

L'orecchio umano è in grado di percepire suoni fino ad una frequenza di 20000 Hz (anche se solitamente un adulto non è in grado di sentire suoni a frequenze maggiori di 16000 Hz), quindi il cambio di qualità è rilevabile ad un orecchio allenato.

In termini di prestazione dei classificatori, le due frequenze di campionamento non hanno però portato a significative differenze nei risultati, quindi si è deciso di mantenere il modello il più veloce e semplice.

3.2 Short-time Fourier transform

Short-time Fourier transform è una trasformata che permette un'analisi tempo-frequenza dei segnali audio, ovvero di come i parametri spettrali del segnale cambino rispetto al tempo. La STFT può essere interpretata come una trasformata di Fourier che "scorre" sul segnale.

La procedura per computare la STFT prevede:

1. suddivisione del segnale audio in segmenti temporali di breve durata
2. applicazione della funzione di windowing su ogni segmento

3. calcolo della trasformata di Fourier discreta (DFT) su ogni segmento

Per la suddivisione in segmenti è necessario utilizzare dei buoni parametri, in quanto esiste un principio di indeterminazione secondo il quale non è possibile stimare con precisione arbitraria e simultaneamente i parametri temporali e frequenziali di un segnale. [13] Ciò significa che, suddividendo il segnale in tanti segmenti di brevissima durata, si otterrà uno spettrogramma con ottima risoluzione temporale ma scarsa risoluzione frequenziale, utilizzando invece segmenti lunghi si otterrà ottima risoluzione frequenziale ma scarsa risoluzione temporale. Una dimensione adatta dei segmenti per poter effettuare delle analisi musicali sullo spettrogramma risultante può essere quella di circa 100 millisecondi.

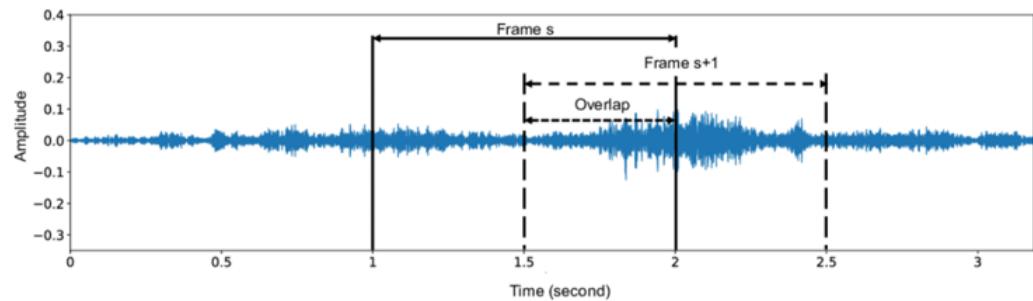


Figura 3.1: Framing del segnale audio

Il segnale presente in ogni segmento temporale deve essere trattato in maniera indipendente dal resto del segnale. Per isolare il segnale presente all'interno di ogni segmento dal resto del segnale viene utilizzata una funzione di windowing. La funzione di windowing effettua un tapering (deformazione): mantiene inalterata l'ampiezza del segnale al centro del segmento, mentre viene azzerata l'ampiezza ai due estremi. L'obiettivo del tapering è di evitare infiltrazione spettrale (leakage), influenze che alcune componenti spettrali potrebbero avere su altre porzioni dello spettro.

Per mitigare la perdita di informazioni agli estremi della finestra, i segmenti adiacenti sono in parte sovrapposti (overlapping).

Nel progetto si è utilizzata la finestra di Hann, una delle diverse funzioni-finestra esistenti.

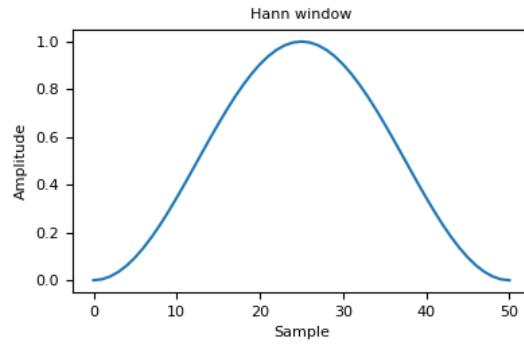


Figura 3.2: Funzione finestra di Hann

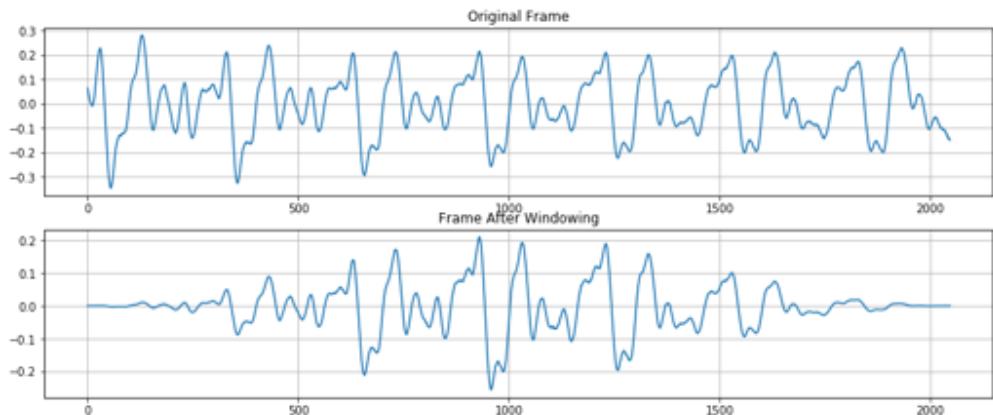


Figura 3.3: Segmento di un segnale audio prima e dopo l'applicazione della finestra di Hann

Una volta che tutti i segmenti sono stati modificati dalla finestra, è necessario applicare la trasformata di Fourier ad ogni segmento in maniera indipendente. In questo modo si ottiene il segnale presente in ogni segmento espresso in termini frequenziali (spettro). Confrontando lo spettro dei diversi

segmenti, si può osservare come le informazioni spettrali cambino rispetto al tempo.

Dato che il segnale audio è digitale, ovvero rappresentato come serie di campioni, viene utilizzata la trasformata discreta di Fourier, che ha come ingresso una funzione discreta i cui valori sono in generale complessi e non nulli, a durata limitata.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-ik\frac{2\pi}{N}n} \quad k = 0, \dots, N-1$$

3.2.1 Creazione della Short-time Fourier transform con Librosa

```
n_FTT = 2048
hop_length = 512
S = librosa.stft(x, n_FTT = n_FTT, hop_length = hop_length)
```

- Il parametro `n_FTT` indica la dimensione della finestra, espressa in numero di campioni che essa contiene.

Data una frequenza di campionamento uguale a 22050 Hz, con un valore di `n_FTT` uguale a 2048, la finestra conterrà 93 millisecondi della traccia audio. Questa dimensione è adatta per l'analisi di segnali musicali.

- Il parametro `hop_length` indica il numero di campioni di cui viene fatta avanzare la finestra. E' necessario utilizzare un valore di `hop_length` minore al valore di `n_FTT`. In questo modo si avrà la sovrapposizione (overlap) delle finestre: uno stesso campione sarà presente in più frame adiacenti, e non verranno perse le informazioni dei campioni agli estremi della finestra.

Il risultato della Short-time Fourier transform è uno spettrogramma a valori discreti, in cui:

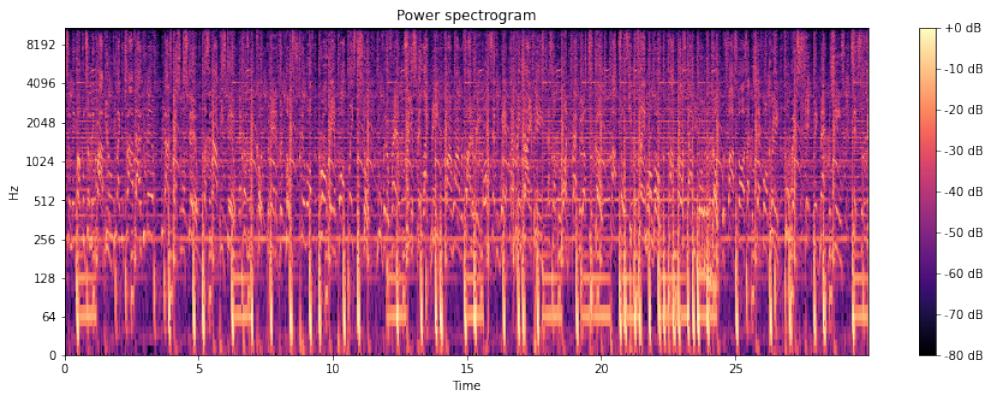


Figura 3.4: Spettrogramma ottenuto con STFT. L’asse delle frequenze è in scala logaritmica e le ampiezze sono espresse in dB

- l’asse temporale è diviso in frame. Il numero di frame è calcolabile come

$$n_frame = \frac{n_samples}{hop_length}$$

dove

$$n_samples = sample_rate * track_duration$$

Utilizzando un valore di `hop_length` uguale a 512 su una traccia audio di 30 secondi campionata a 22050 Hz, si otterranno 1292 frame.

- l’asse delle frequenze diviso in bande. Il numero di bande di frequenza è calcolabile come

$$n_bands = 1 + \frac{n_FFT}{2}$$

Utilizzando un valore di `n_FFT` uguale a 2048, ci saranno 1025 bande di frequenza.

Capitolo 4

Estrazione di feature

4.1 MFCC (Mel Frequency Cepstral Coefficients)

Gli MFCC sono una rappresentazione compatta dello spettro di un segnale audio. Sono feature molto utilizzate nell'ambito del riconoscimento vocale automatico (ASR), perché in grado di mettere in evidenza le differenze tra diversi fonemi linguistici.

Gli MFCC sono utilizzati in questo progetto perché riescono a descrivere il timbro sonoro, una caratteristica molto importante nelle tracce musicali. [14] Le tre caratteristiche che distinguono i suoni sono la frequenza fondamentale, l'intensità (o ampiezza) ed il timbro. Le prime due sono facilmente ottenibili, perché sono grandezze fisiche direttamente misurabili. Ad esempio, nello spettrogramma creato con STFT, le frequenze sono mostrate sull'asse delle ordinate e le intensità sono mostrate con il colore. Le frequenze fondamentali in genere sono quelle aventi una maggiore intensità. Il timbro è invece una grandezza multidimensionale, determinato dalle armoniche. Dato un suono fondamentale avente una certa frequenza, le armoniche sono frequenze che si sviluppano insieme alla fondamentale, e sono multiple ad essa. Il timbro è determinato da quali armoniche si sviluppino e da quanto esse siano ampie. La differenza nelle armoniche è ciò che rende il suono di una chitarra diverso

dal suono di un sassofono, quando essi suonano la stessa nota (la frequenza fondamentale).

4.1.1 Calcolo degli MFCC

Per calcolare i valori degli MFCC, si utilizza lo spettrogramma ottenuto con STFT come punto di partenza. Sono necessari tre passaggi:

1. Applicazione del filtro di mel
2. Applicazione di una scala logaritmica alle ampiezze
3. Applicazione della trasformata discreta del coseno

Il filtro di Mel è utilizzato per rendere lo spettrogramma più simile a come un orecchio umano percepisce l'audio. La coclea, parte dell'organo uditivo, percepisce meglio le variazioni di tonalità nelle basse frequenze rispetto a variazioni nella zona delle alte frequenze. [15] Il filtro di Mel è formato da tanti filtri triangolari. Il primo filtro è molto stretto e indica quanta energia è presente attorno agli 0 Hz. Andando verso le frequenze più alte, i filtri triangolari diventano sempre più larghi, in quanto le variazioni diventano per noi meno significative. In questo progetto sono stati utilizzati 128 filtri triangolari per formare il filtro di Mel.

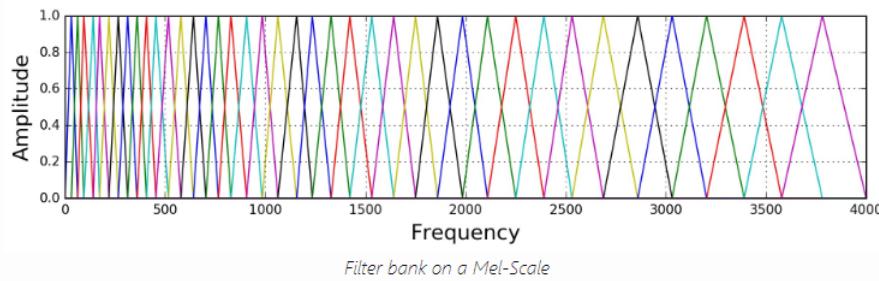


Figura 4.1: Filtro di Mel

Facendo il prodotto scalare tra lo spettro di ogni frame della STFT ed il filtro di Mel, si ottengono, per ciascun frame, tanti coefficienti quanti

sono i filtri triangolari utilizzati (128 in questo progetto). Questi coefficienti indicano l'ampiezza del segnale alle frequenze relative a ciascun filtro triangolare.

Librosa, per rappresentare graficamente lo spettrogramma di Mel, riconverte i coefficienti di Mel nelle rispettive frequenze, misurate in hertz. Si noti però che nei successivi passaggi vengono utilizzati i coefficienti così come sono stati calcolati, quindi rappresentati in una matrice di forma 128 x n_frame

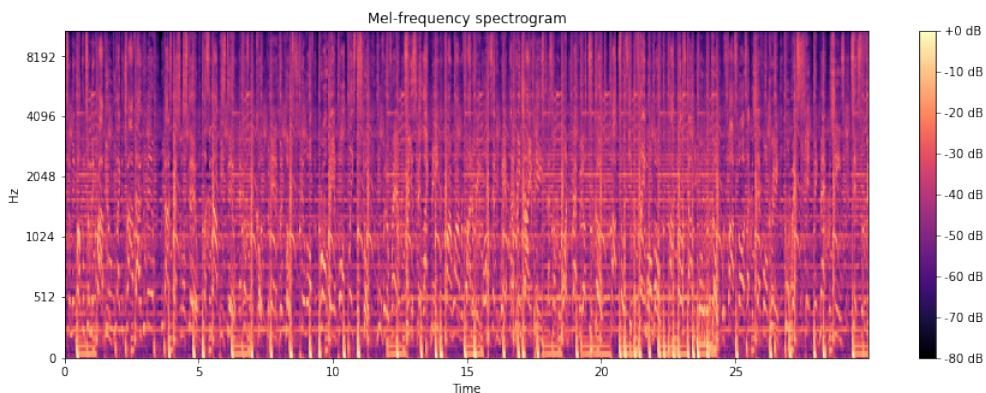


Figura 4.2: Spettrogramma di Mel. Per la rappresentazione, Librosa converte le ampiezze in dB, e l'asse y mostra le frequenze in scala logaritmica

Ottenuti i coefficienti di Mel, viene a loro applicato il logaritmo. Anche questa operazione è motivata dall'udito umano, in quanto non percepiamo il volume in maniera lineare. Generalmente per raddoppiare il volume percepito, bisogna aumentare di 8 volte l'energia utilizzata. Questa compressione delle intensità è quindi utile per rendere le feature più simili a ciò che sentiamo. Il logaritmo viene applicato convertendo i valori di ampiezza in decibel:

$$\left(\frac{N_1}{N_2} \right)_{\text{dB}} = 10 \log_{10} \left(\frac{N_1}{N_2} \right)$$

L'operazione finale è l'applicazione della trasformata discreta del coseno (DCT-II), una trasformata simile alla trasformata di Fourier discreta.

$$X_k = \sum_{n=0}^{N-1} x_n \cos \left[\frac{\pi}{N} \left(n + \frac{1}{2} \right) k \right] \quad k = 0, \dots, N-1.$$

Vengono quindi ottenuti nuovi coefficienti (sempre in una matrice di forma 128 x n_frame) che rappresentano lo "spettro dello spettro" del segnale, chiamato cepstrum.

Il cepstrum serve ad analizzare le velocità di cambiamento del contenuto spettrale del segnale. Nel cepstrum, i picchi indicano la presenza di un pitch (altezza di una nota) e sono visibili perché le armoniche, nello spettro, sono periodiche, con periodo corrispondente all'altezza della nota. Il cepstrum è una rappresentazione dello spettro che mette in evidenza le armoniche (e quindi il timbro sonoro)

I coefficienti così calcolati sono gli MFCC.

Dei 128 coefficienti, vengono mantenuti solo i primi 12, considerati i più significativi. [16]

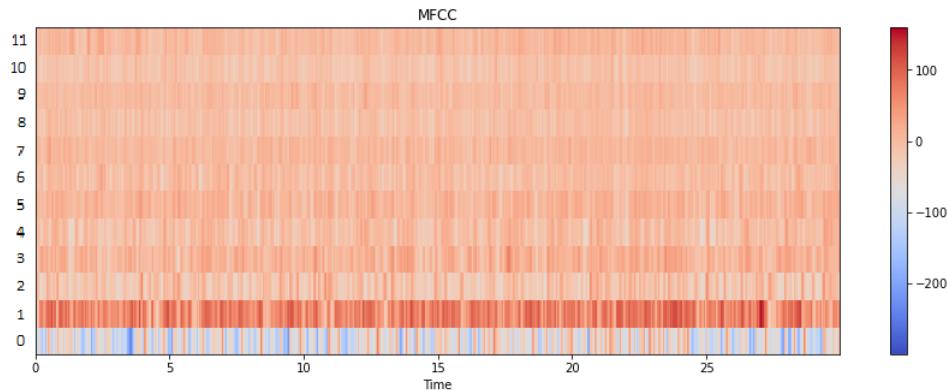


Figura 4.3: MFCC della traccia "Food" di AWOL

L'asse y del grafico suddivide i 12 coefficienti, relativi ai primi 12 filtri triangolari del filtro di Mel. L'asse x mostra il tempo, suddiviso discretamente in frame. Il colore rappresenta il valore dei coefficienti.

4.1.2 Estrazione di feature dagli MFCC

```
S_mel = librosa.feature.melspectrogram(S=np.abs(S)**2, sr=22050, n_mels=128)
S_mel_dB = librosa.power_to_db(S_mel)
mfccs = librosa.feature.mfcc(S=S_mel_dB, n_mfcc=12, sr=sr)
```

Si ottiene una matrice di dimensione 12 x n_frame per ogni traccia, con n_frame approssimativamente uguale a 1292 a causa dei parametri precedentemente utilizzati per la STFT.

Avere 15504 valori numerici che descrivano solamente gli MFCC di una traccia è eccessivo: usarli come feature su cui addestrare un classificatore sarebbe computazionalmente molto impegnativo e potrebbe dare pessimi risultati, a causa del problema dell'overfitting.

Si è quindi rivelato necessario "condensare" le informazioni.

Per fare ciò si sono utilizzate funzioni matematiche e statistiche, applicabili a set di valori. I set di valori presi in considerazione sono i coefficienti. Sebbene possa essere interessante conoscere il modo in cui il segnale si evolve nel tempo, si è deciso in questo progetto di eliminare le informazioni temporali.

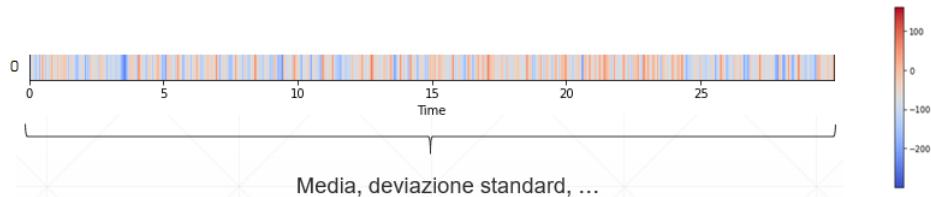


Figura 4.4: Estrazione di feature dall'evoluzione nel tempo di un coefficiente MFCC

Per ogni coefficiente vengono calcolati:

- Media aritmetica
- Deviazione standard
- Minimo
- Massimo

- Mediana
- Skewness (simmetria)
- Kurtosis (curtosi)

Si sono quindi aggiunti al dataset 7 valori numerici per ognuno dei 12 coefficienti: 84 feature numeriche estratte dagli MFCC.

4.2 Chroma Feature

Il cromagramma, detto anche chroma feature, è uno strumento per l’analisi musicale. Esso mostra, per ogni frame di una traccia musicale, le intensità delle classi di altezze (pitch classes).

Le frequenze vengono mappate alle note musicali seguendo il 12-TET, il temperamento equabile utilizzato nella quasi totalità della musica occidentale. [17] Il 12-TET definisce una scala musicale suddividendo le ottave in 12 intervalli di frequenza tra di loro uguali: C (Do), C# (Do#), D (Re), D# (Re#), E (Mi), F (Fa), F# (Fa#), G (Sol), G# (Sol#), A (La), A# (La#), B (Si).

Le ottave sono identificate nella notazione scientifica da un numero. Ogni ottava parte da una frequenza doppia rispetto alla precedente, la scala musicale si ripete nello stesso ordine.

Una classe di altezze è l’insieme di tutte le altezze che sono presenti ad un numero intero di ottave di distanza: per esempio, la classe di altezze C (Do in italiano) consiste di tutti i C in tutte le ottave. Le ottave esistono in numero infinito, ma nelle applicazioni pratiche si tengono in considerazione solo quelle udibili dagli umani.

Nel cromagramma non vengono mantenute le informazioni del timbro. Il timbro è determinato dall’intensità delle armoniche, le frequenze multiple alla nota fondamentale. Per la costruzione del cromagramma, le intensità di tutte le armoniche di una nota musicale vengono raggruppate nella stessa classe di altezze, e non sono più distinguibili.

	C	C#	D	Eb	E	F	F#	G	G#	A	Bb	B
0	16.35	17.32	18.35	19.45	20.60	21.83	23.12	24.50	25.96	27.50	29.14	30.87
1	32.70	34.65	36.71	38.89	41.20	43.65	46.25	49.00	51.91	55.00	58.27	61.74
2	65.41	69.30	73.42	77.78	82.41	87.31	92.50	98.00	103.8	110.0	116.5	123.5
3	130.8	138.6	146.8	155.6	164.8	174.6	185.0	196.0	207.7	220.0	233.1	246.9
4	261.6	277.2	293.7	311.1	329.6	349.2	370.0	392.0	415.3	440.0	466.2	493.9
5	523.3	554.4	587.3	622.3	659.3	698.5	740.0	784.0	830.6	880.0	932.3	987.8
6	1047	1109	1175	1245	1319	1397	1480	1568	1661	1760	1865	1976
7	2093	2217	2349	2489	2637	2794	2960	3136	3322	3520	3729	3951
8	4186	4435	4699	4978	5274	5588	5920	6272	6645	7040	7459	7902

Figura 4.5: Note musicali e le corrispondenti frequenze in hertz secondo 12-TET. Ogni riga è un'ottava, ogni colonna è una classe di altezze.

Dal cromagramma non si hanno nemmeno informazioni sulle ottave delle note: ad esempio, se si ha una grande intensità nella classe di altezza D (Re nella notazione italiana), non si può sapere se questa intensità è concentrata alla nota D_1 (36.7 Hz, può essere suonata da un basso elettrico) o alla nota D_6 (1174.6 Hz, può essere suonata da un violino).

Il cromagramma è però utile nel mostrare le note e gli accordi (armonia) utilizzati nelle tracce musicali, senza fare distinzioni riguardo al tono e agli strumenti musicali utilizzati. L'armonia di una traccia è una caratteristica molto correlata al genere musicale di appartenenza.

4.2.1 Creazione del cromagramma

Il cromagramma viene creato a partire dallo spettrogramma ottenuto con STFT.

Le frequenze attorno a quelle definite da 12-TET vengono raggruppate in bande (bin). Ogni banda rappresenta una nota musicale. Vengono quindi raggruppate le bande in classi di altezze, ottendendo così 12 valori.

L'asse y del grafico mostra le 12 classi di altezze. L'asse x mostra il tempo, suddiviso discretamente in frame. Il colore rappresenta l'intensità.

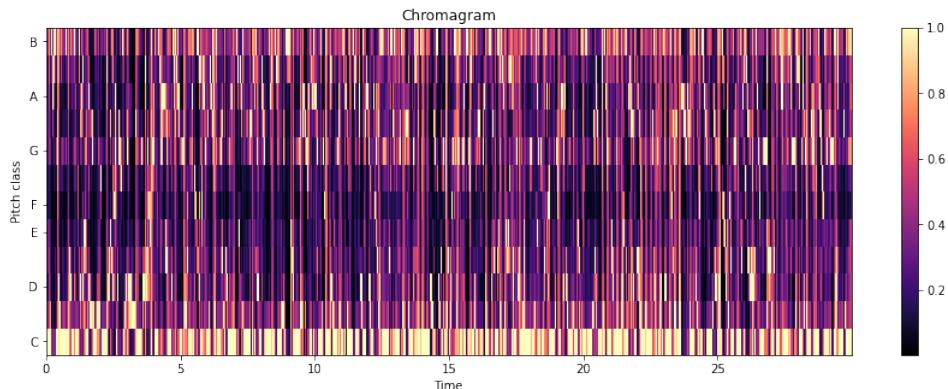


Figura 4.6: Cromagramma della traccia "Food" di AWOL

4.2.2 Estrazione di feature dal cromagramma

```
chroma = librosa.feature.chroma_stft(S = np.abs(S)**2, sr
=22050)
```

La funzione di librosa restituisce un array numpy di forma (12, n_frame). Come nel caso degli MFCC, si vuole diminuire il numero di valori rappresentativi del cromagramma. Si è quindi deciso di applicare queste funzioni su ogni classe di altezze:

- Media aritmetica
- Deviazione standard
- Minimo
- Massimo
- Mediana
- Skewness
- Kurtosis

In questo modo le informazioni temporali vanno perse. Da queste informazioni aggregate si possono comunque dedurre le note o gli accordi principalmente utilizzati nell'armonia della traccia musicale.

4.3 Zero-crossing rate

Il zero-crossing rate è la frequenza con cui un segnale cambia segno, da negativo a positivo o viceversa, in un dato intervallo di tempo. Viene considerata una feature di basso livello, in quanto semplice da calcolare. Viene molto utilizzata nell'ambito della Voice Activity Detection (VAD), per definire se la voce umana è presente in un segmento audio. Il ZCR viene anche utilizzato, come nel caso di questo progetto, nell'ambito del MIR (Music Information Retrieval) perché in grado di determinare la presenza dei suoni percussivi.

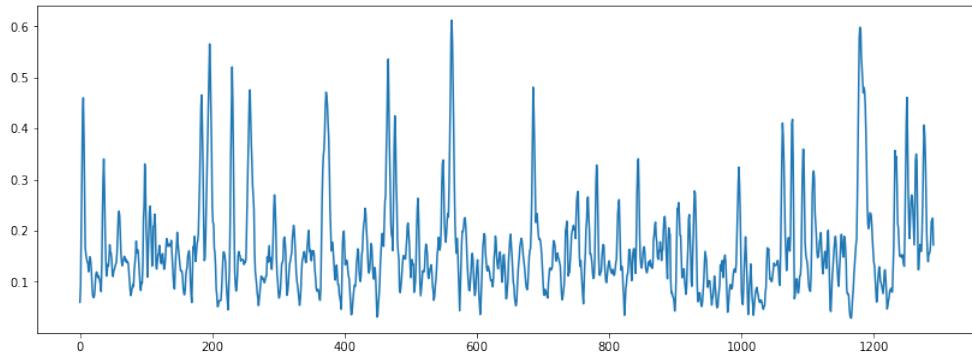


Figura 4.7: Evoluzione del ZCR nel tempo della traccia "Food" di AWOL

4.3.1 Calcolo dei zero-crossing rate

Per il calcolo dei ZCR non è necessario avere lo spettrogramma della traccia audio. Viene utilizzato direttamente il campionamento.

Il segnale campionato viene suddiviso in frame, nello stesso modo utilizzato per il calcolo della STFT. Non viene però applicata alcuna funzione di windowing, dato che non è necessario calcolare la trasformata.

Per ogni frame, contenente N campioni, viene calcolato il ZCR:

$$zcr = \frac{1}{N} \sum_{k=1}^N | s(k) - s(k-1) |$$

Dove $s(k) = 1$ se il segnale ha ampiezza positiva nel campione k, $s(k) = 0$ altrimenti.

4.3.2 Estrazione di feature dai ZCR

```
zero_crossing_rates = librosa.feature.zero_crossing_rate(x,
    frame_length = 2048, hop_length = 512)
```

La funzione librosa restituisce un array contenente n_frame valori. Ogni valore indica la frazione di zero-crossings nell'i-esimo frame. Si è deciso di condensare le informazioni utilizzando le seguenti funzioni:

- Media aritmetica
- Deviazione standard
- Minimo
- Massimo
- Mediana

Così facendo, si perdono le informazioni temporali. Il valore medio dei ZCR può indicare il ritmo della traccia musicale, dato dalle percussioni, mentre la deviazione standard può mostrare se ci sono grosse variazioni nel pattern delle percussioni lungo la traccia.

Si sono dunque aggiunti al dataset 5 valori numerici rappresentativi del ZCR.

4.4 Spectral Centroid

Lo spectral centroid indica il centro di massa dello spettro di un segnale.

Se gli MFCC descrivono in maniera piuttosto precisa la struttura dello spettrogramma, lo spectral centroid mostra la frequenza che viene percepita come prevalente.

Inoltre, ha una forte connessione con la "brillantezza del suono". La brillantezza del suono è un concetto che ha a che fare con il timbro, ma che non può essere fisicamente calcolata in quanto difficilmente descrivibile dal punto di vista fisico. Essa è considerata una caratteristica soggettiva, percepita dall'orecchio umano. Uno studio ha mostrato che in genere una sensazione di brillantezza sonora è correlata all'intensità delle armoniche più alte. [18]

Quindi un maggiore valore di spectral centroid può indicare una maggiore brillantezza percepita.

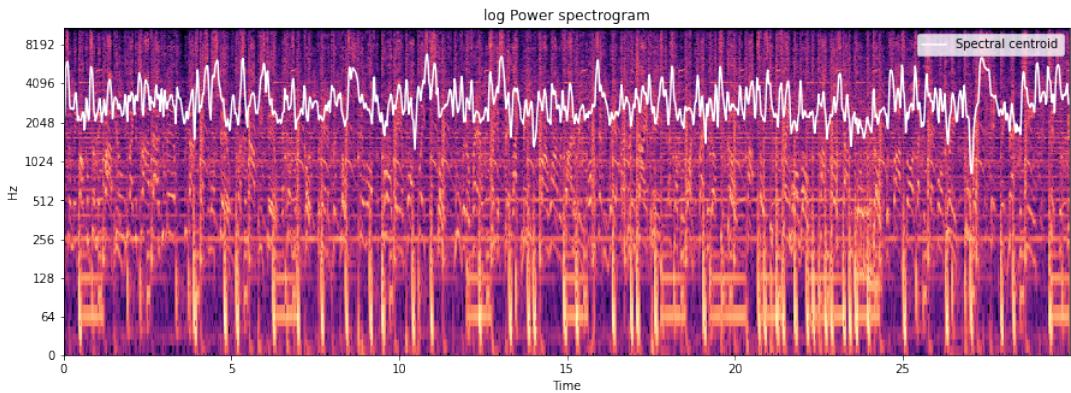


Figura 4.8: Spectral centroid, in bianco, visualizzato sullo spettrogramma della traccia "Food" di AWOL

4.4.1 Calcolo dello spectral centroid

Lo spectral centroid viene calcolato a partire dallo spettrogramma ottenuto con STFT. La formula viene applicata per ogni frame dello spettrogramma.

Essa è equivalente alla media pesata delle frequenze, dove il peso è l'intensità spettrale.

$$\text{Centroid} = \frac{\sum_{i=1}^N x_i f_i}{\sum_{i=1}^N x_i}$$

Dove f_i rappresenta la frequenza i-esima e x_i rappresenta l'intensità della frequenza.

4.4.2 Estrazione di feature dallo spectral centroid

```
spec_centroid = librosa.feature.spectral_centroid(x, sr=sr)
media_centroid = np.mean(spec_centroid)
deviaz_centroid= np.std(spec_centroid)
massimo_centroid = npamax(spec_centroid)
minimo_centroid = npamin(spec_centroid)
mediana_centroid = np.median(spec_centroid)
```

Vengono utilizzate le cinque funzioni statistiche già viste nell'estrazione di feature dagli ZCR. Si ottengono così 5 feature da aggiungere al dataset.

4.5 Spectral Rolloff

Spectral rolloff è la frequenza sotto la quale una certa percentuale dell'energia spettrale (ad esempio 85%) risiede. Quindi, scegliendo il valore di 85%, come è stato fatto per il progetto, lo spectral rolloff è l'85-esimo percentile di frequenza.

Esso indica se l'energia è concentrata nelle basse frequenze. Spesso viene usato per distinguere audio in cui è presente la voce umana.

4.5.1 Estrazione di feature dallo spectral rolloff

```
spec_rolloff = librosa.feature.spectral_rolloff(x, sr=sr,
    roll_percent=0.85)
media_rolloff = np.mean(spec_rolloff)
deviaz_rolloff= np.std(spec_rolloff)
massimo_rolloff = npamax(spec_rolloff)
minimo_rolloff = npamin(spec_rolloff)
mediana_rolloff = np.median(spec_rolloff)
```

Vengono utilizzate le cinque funzioni statistiche già viste nell'estrazione di feature dagli ZCR. Si ottengono così 5 feature da aggiungere al dataset.

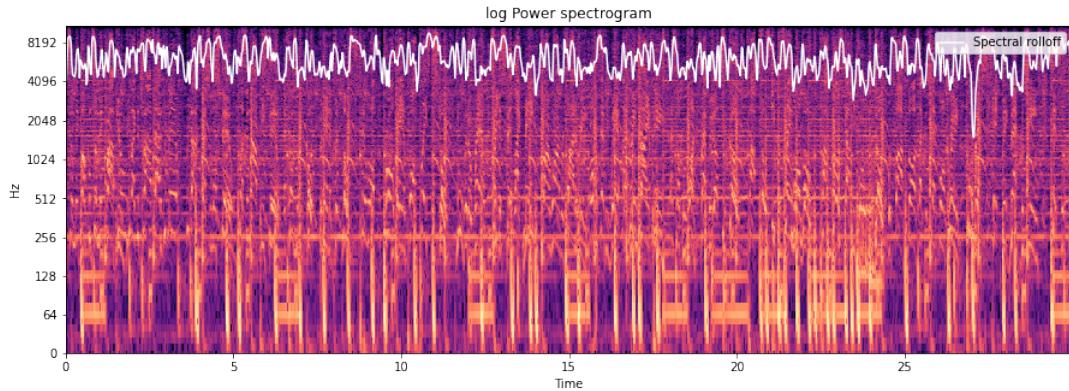


Figura 4.9: Spectral rolloff, in bianco, visualizzato sullo spettrogramma della traccia "Food" di AWOL

4.6 Spectral Bandwidth

Spectral bandwidth indica quanto sia larga la banda di frequenze in cui l'intensità spettrale ha valori rilevanti.

Un valore alto indica che l'energia è distribuita anche in frequenze lontane dallo spectral centroid, un valore basso, al contrario, indica che l'energia è concentrata in prossimità di esso.

4.6.1 Calcolo della spectral bandwidth

Viene calcolata come una deviazione standard pesata, in cui il peso è l'intensità spettrale e la media è lo spectral centroid.

$$\text{Bandwidth} = \left(\sum_{i=1}^N x_i (f_i - f_c)^2 \right)^{\frac{1}{2}}$$

Dove f_i rappresenta la i -esima frequenza e x_i rappresenta la relativa intensità. f_c è lo spectral centroid.

4.6.2 Estrazione di feature dallo spectral bandwidth

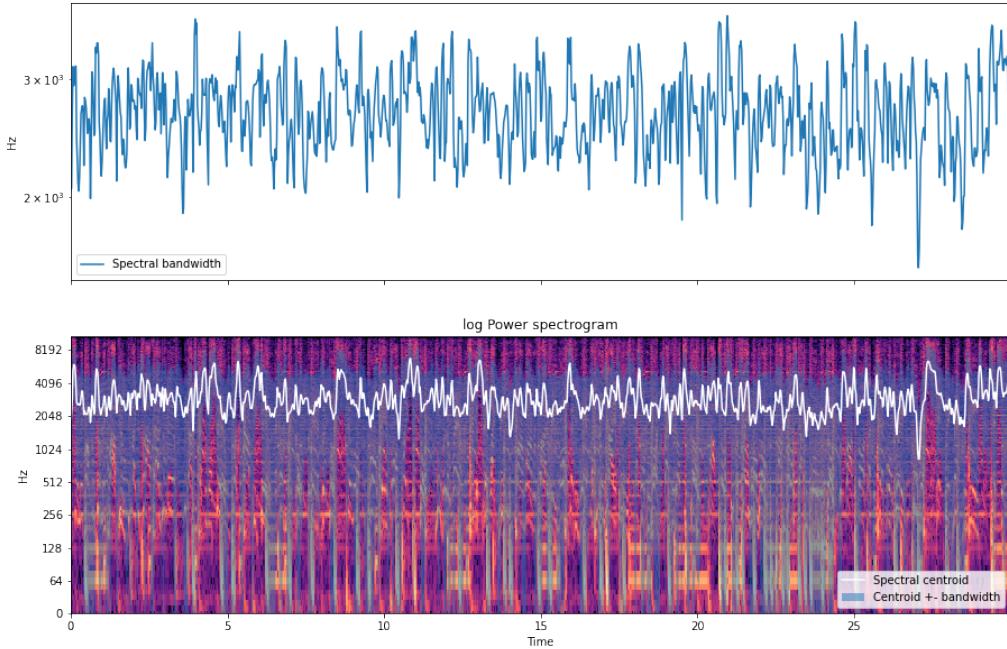


Figura 4.10: Spectral bandwidth della traccia "Food" di AWOL. Nella seconda immagine viene evidenziata in blu la banda indicata dalla spectral bandwidth, centrata sullo spectral centroid

```
spec_bandwidth = librosa.feature.spectral_centroid(x, sr=sr)
media_bandwidth = np.mean(spec_bandwidth)
deviaz_bandwidth= np.std(spec_bandwidth)
massimo_bandwidth = np.amax(spec_bandwidth)
minimo_bandwidth = np.amin(spec_bandwidth)
mediana_bandwidth = np.median(spec_bandwidth)
```

Vengono utilizzate le cinque funzioni statistiche già viste nell'estrazione di feature dagli ZCR. Si ottengono così 5 feature da aggiungere al dataset.

4.7 Spectral Contrast

Lo spectral contrast di una banda di frequenze indica il rapporto tra l'intensità massima e l'intensità minima presenti in quella banda.

Lo spettrogramma ottenuto con STFT viene suddiviso in 6 bande di frequenza. Per ognuna viene calcolato il contrasto: il rapporto tra il quantile superiore (intensità massima) ed il quantile inferiore (intensità minima).

Un contrasto alto significa che nella banda c'è un segnale chiaro e stretto, mentre un contrasto basso indica un segnale rumoroso su tutta la banda. 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.

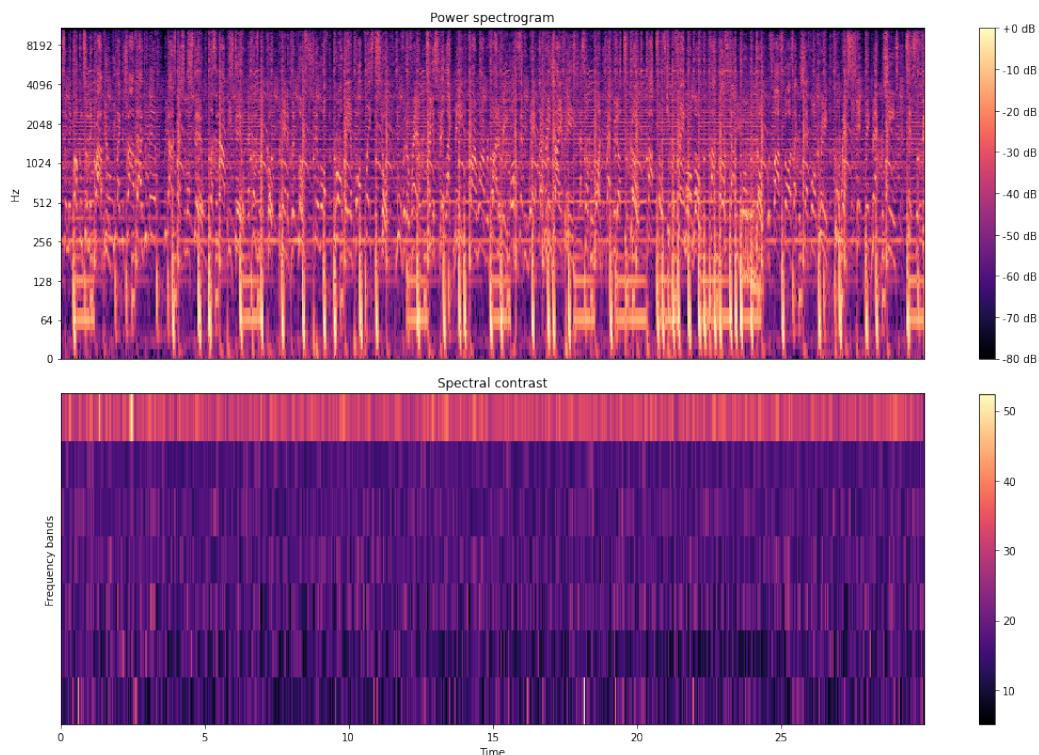


Figura 4.11: Spectral contrast della traccia "Food" di AWOL

4.7.1 Estrazione di feature dallo spectral contrast

```
spec_contrasts = librosa.feature.spectral_contrast(x, sr=sr,
    n_bands = 6, quantile=0.02)
```

```
for frequency_band in spec_contrasts:  
    media_array.append(np.mean(frequency_band))  
    deviaz_array.append(np.std(frequency_band))  
    max_array.append(np.amax(frequency_band))  
    min_array.append(np.amin(frequency_band))  
    mediana_array.append(np.median(frequency_band))  
    skew_array.append(stats.skew(frequency_band))  
    kurtosis_array.append(stats.kurtosis(frequency_band))
```

Per ognuna delle 6 bande di frequenza vengono utilizzate le 7 funzioni statistiche viste per gli MFCC. Si ottengono quindi 42 feature.

Capitolo 5

Addestramento dei classificatori

5.1 Preparazione del dataset

Unendo tutte le feature estratte dalle tracce musicali, otteniamo un dataset rappresentabile come una tabella. Ogni riga identifica una traccia audio, ogni colonna identifica una feature. La tabella ha quindi $27789 \text{ righe} \times 230 \text{ colonne}$

track_id	('mfcc', 'mean', '1')	('mfcc', 'mean', '2')	('mfcc', 'mean', '3')	('mfcc', 'mean', '4')	...	('contrast', 'kurtosis', '3')	('contrast', 'kurtosis', '4')	('contrast', 'kurtosis', '5')	('contrast', 'kurtosis', '6')
2	-67.384201	65.155075	-10.669196	12.176932	...	0.073019	0.147885	0.219722	0.612398
3	-97.992485	72.889534	-3.973132	30.070477	...	0.136208	-0.548991	-0.074470	0.075461
5	-106.587730	87.239357	12.156827	24.140398	...	0.069869	0.671903	0.269572	2.383789
10	-17.238174	94.442024	-48.614548	32.370766	...	0.366582	0.777638	1.636699	0.900061
134	-84.152870	84.352127	-2.286359	34.442200	...	0.284433	0.538790	0.211476	0.279040
...
155304	-79.091148	86.971321	13.332952	42.796810	...	0.299717	0.083636	-0.160900	-0.292233
155305	-192.851105	133.374039	30.840197	33.762676	...	1.141640	-0.064294	-0.147984	0.391955
155306	-217.283554	162.253387	-1.447354	32.816235	...	0.539851	0.134641	0.157030	0.926083
155307	-167.747559	87.483261	6.906166	19.528233	...	-0.262105	-0.664807	-0.387809	-0.170939
155314	-82.922188	110.042313	-19.682865	35.809292	...	0.876661	0.258984	0.556226	0.568842

27789 rows \times 230 columns

Figura 5.1: Dataset completo

Il dataset viene suddiviso in due parti: training set e test set. Il training set contiene il 67% delle righe, il test set il 33%.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.33, random_state=42)
```

5.1.1 Normalizzazione del dataset

La maggior parte degli algoritmi di Machine Learning necessita di un dataset normalizzato. Data una matrice X in cui ogni riga è un campione (una traccia musicale) e ogni colonna identifica una feature, bisogna normalizzare ogni colonna individualmente, in modo che la distribuzione dei valori di quella colonna abbia media = 0 e varianza = 1.

Per ogni valore x:

$$z = \frac{x - \mu}{\sigma}$$

Dove μ è la media della distribuzione e σ è la deviazione standard.

Per far questo in Python viene utilizzato il metodo StandardScaler()

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Lo scaler viene "fittato" sui soli dati di training, il che significa che media e deviazione standard di ogni feature vengono calcolati esclusivamente su questi dati. Utilizzare anche i dati di test per il calcolo di media e deviazione standard non sarebbe corretto, in quanto essi non devono in alcun modo contaminare la fase di addestramento del classificatore.

Vengono quindi normalizzati sia il training che il test set, quest'ultimo sarà utilizzato solo dopo la creazione del classificatore.

5.1.2 Oversampling del training set

Per bilanciare le classi nel training set si sono utilizzati metodi di oversampling, che aumentano il numero di campioni appartenenti alle classi sottorappresentate. Vengono creati campioni sintetici utilizzando uno dei possibili algoritmi esistenti. In genere, per la creazione di ogni campione sintetico, questi algoritmi cercano due o più istanze simili (utilizzando una misura di distanza) e ne creano una nuova che si trova nel mezzo, perturbata casualmente. I campioni sintetici quindi non aggiungono informazioni reali al dataset, ma fanno in modo che gli algoritmi di classificazione in fase di training si trovino un numero di campioni uguali per ogni classe.

Sono stati testati diversi algoritmi, dato che non ne esiste uno strettamente migliore degli altri. L'algoritmo migliore varia in base al dataset ed al tipo di classificatore che si andrà ad utilizzare. Infatti, i classificatori addestrati (SVM, Logistic Regression, kNN e Random Forest) hanno dato i propri risultati migliori utilizzando algoritmi diversi.

Con il classificatore SVM si è scelto l'algoritmo SVM SMOTE:

```
#SVM SMOTE resampling
from imblearn.over_sampling import SVMSMOTE
X_train_resampled, y_train_resampled = SVMSMOTE().
    fit_resample(X_train_scaled, y_train)
```

Con il classificatore kNN si è scelto l'algoritmo k-means SMOTE:

```
#k means SMOTE resampling
from imblearn.over_sampling import KMeansSMOTE
X_train_resampled, y_train_resampled = KMeansSMOTE().
    fit_resample(X_train_scaled, y_train)
```

Con i classificatori Random Forest e Logistic Regression si è scelto l'algoritmo SMOTE:

```
#SVM SMOTE resampling
from imblearn.over_sampling import SVMSMOTE
X_train_resampled, y_train_resampled = SVMSMOTE().
    fit_resample(X_train_scaled, y_train)
```

5.2 Creazione dei classificatori

Il classificatore SVM utilizza la tecnica one-versus-one (ovo) ed un kernel rbf, che ha dato risultati migliori rispetto ad un kernel lineare.

```
from sklearn import svm
clf = svm.SVC(kernel='rbf', decision_function_shape='ovo',
               gamma = 'auto')
# Fit data
clf = clf.fit(X_train_resampled, y_train_resampled)
```

Il classificatore Logistic Regression ha un numero massimo di iterazioni fissato a 2000.

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=0, max_iter = 2000).fit
(X_train_resampled, y_train_resampled)
```

Il classificatore k-nearest neighbours utilizza un valore di k uguale ad uno

```
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors=1)
clf.fit(X_train_resampled, y_train_resampled)
```

Il classificatore Random Forest utilizza 100 stimatori

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators = 100, criterion =
                             'gini', random_state = 42)
clf.fit(X_train_resampled, y_train_resampled)
```

Tutti i classificatori e lo scaler contenente media e deviazione standard dei dati di training vengono memorizzati come file .joblib, per poter essere successivamente utilizzati.

```
joblib.dump(scaler, "Scaler.joblib")
joblib.dump(clf, "SVMClassifier.joblib")
```

5.3 Feature selection

Le feature estratte non contengono tutte la stessa quantità di informazione. Avere un grande numero di feature, specie se alcune di queste non sono particolarmente significative, può avere effetti controproducenti:

- Overfitting (adattamento eccessivo): all'aumentare del numero di dimensioni dello spazio delle feature, il numero di possibili configurazioni aumenta esponenzialmente, di conseguenza il numero di queste configurazioni coperte da ogni osservazione diminuisce. [19] Viene a mancare la flessibilità necessaria nella classificazione di nuovi campioni non usati in fase di training. Il metodo migliore per combattere l'overfitting è quindi quello di aumentare il numero di osservazioni, ma nella pratica non è sempre possibile.
- Maggior tempi di training e di predizione: un modello con tante feature è molto complesso.

Per valutare l'importanza delle feature ed eventualmente ridurne il numero si è utilizzato l'algoritmo RFE (Recursive Feature Elimination). È un algoritmo di tipo wrapper-based: la selezione del set di feature è considerato come un problema di ricerca.

```
rfe_selector = RFECV(estimator=LogisticRegression(), step=1)
rfe_selector.fit(X_scaled, y)
rfe_support = rfe_selector.get_support()
rfe_feature = X.loc[:,rfe_support].columns.tolist()
new_complete_unscaled_dataset = X.loc[ : , rfe_feature ]
```

La funzione RFECV.fit:

- utilizza il metodo di cross-validation con 5 fold sul dataset per l'addestramento e la validazione degli stimatori
- estraе il parametro `_coef` di ogni feature
- elimina la feature con la minore importanza

- ripete il procedimento ricorsivamente, utilizzando il sottoinsieme di feature

Una volta terminata l'esecuzione dell'algoritmo, l'attributo support conterrà il set di feature con il quale lo stimatore ha dato il miglior risultato.

Utilizzando come stimatore il modello Logistic Regression si nota che, eliminando qualsiasi feature, si andrebbe a peggiorare i risultati del classificatore. Si può dedurre quindi che tutte le feature sono abbastanza significative e che non sono presenti in numero eccessivo, data anche la grande quantità di campioni presenti nel dataset.

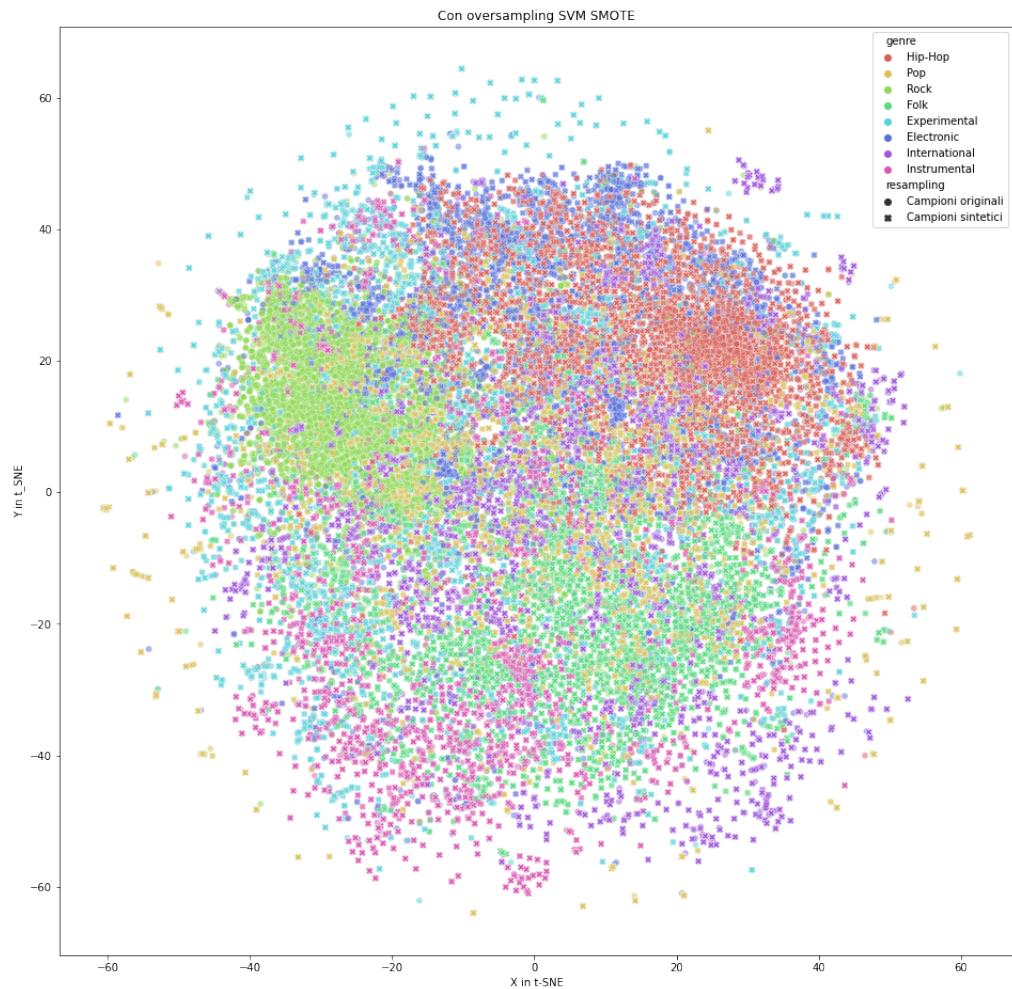


Figura 5.2: Visualizzazione grafica tramite t-SNE del dataset bilanciato con oversampling SVM SMOTE. Si può notare come i campioni siano distribuiti in maniera piuttosto densa in corrispondenza dei cluster delle diverse classi, mentre ci sono relativamente pochi outliers ai bordi del grafico.

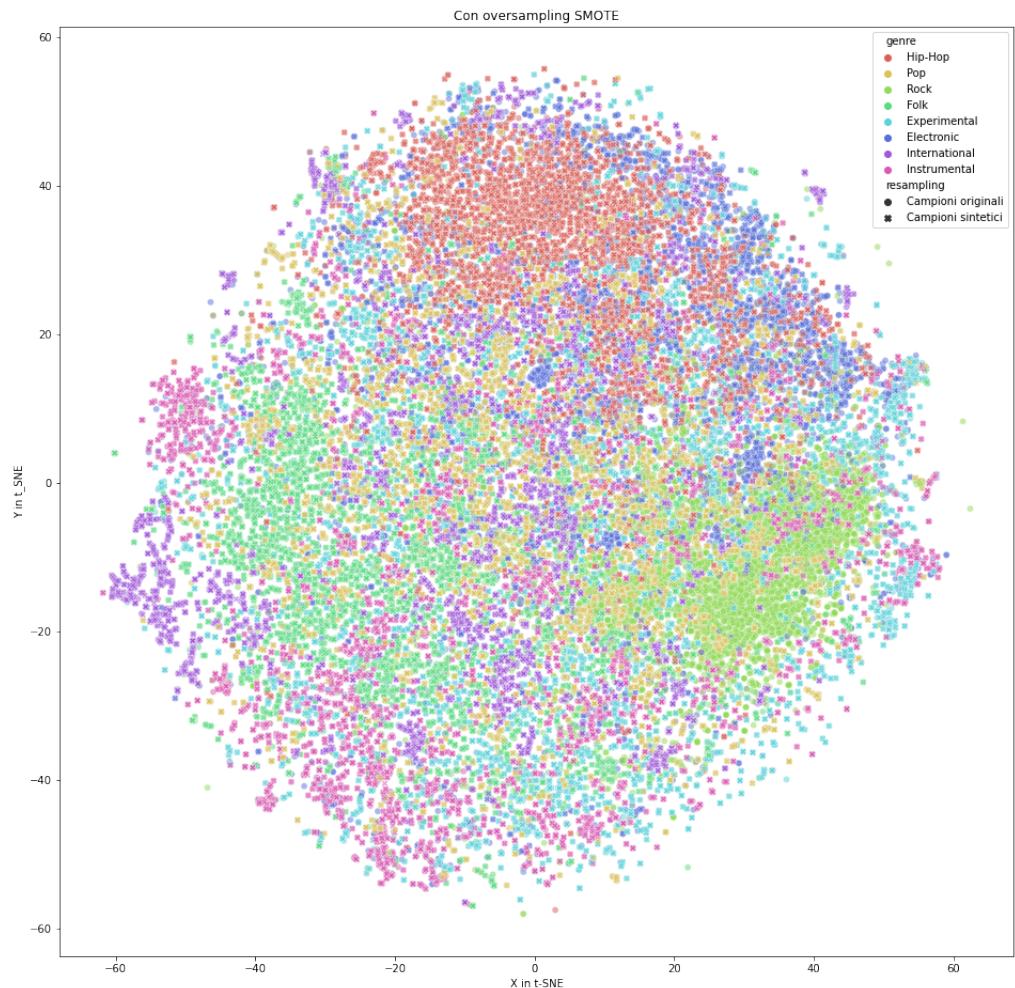


Figura 5.3: Visualizzazione grafica tramite t-SNE del dataset bilanciato con oversampling SMOTE. Si può notare come la distribuzione dei campioni sia piuttosto omogenea nello spazio.

Capitolo 6

Risultati

6.1 Risultati della classificazione

Per valutare l'efficacia dei classificatori si possono utilizzare diversi parametri. Uno dei parametri più significativi è l'accuratezza:

$$Accuracy = \frac{\text{Numero di predizioni corrette}}{\text{Numero totale di predizioni}}$$

Il set di test di questo progetto è però non bilanciato. Questa metrica non è quindi sufficiente a descrivere l'efficacia del classificatore: se essoassegnasse a tutti i campioni solamente la classe più rappresentata, si potrebbe ottenere un valore di accuracy buono quando in realtà il classificatore non è in grado di distinguere le classi.

Bisogna quindi utilizzare anche altre metriche:

- Matrice di confusione: sulla diagonale della matrice sono mostrate le predizioni corrette, nelle altre posizioni sono mostrati falsi positivi e falsi negativi.
- Precisione (precision): è un valore calcolato per ogni classe.

$$Precisione = \frac{\text{Veri positivi}}{\text{Veri positivi} + \text{Falsi positivi}}$$

- Recupero (recall): è un valore calcolato per ogni classe.

$$Recupero = \frac{\text{Veri positivi}}{\text{Veri positivi} + \text{Falsi negativi}}$$

- F1-Score: è un valore calcolato per ogni classe, è la media pesata di precisione e recupero.

$$F1\text{-Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

6.1.1 Osservazioni

Dai risultati si nota come l'algoritmo di classificazione più performante per questo progetto sia SVM.

Il genere Pop è il più difficile da classificare, seguito dal genere Experimental. Ciò può essere giustificato dal fatto che questi generi sono molto ampi e abbiano caratteristiche varie.

I generi che nel dataset iniziale sono più rappresentati, in termini di quantità di campioni, sono quelli che hanno dato i migliori risultati: Hip-Hop, Rock e Electronic. Una causa è sicuramente il fatto che, avendo a disposizione più esempi reali, il classificatore riesce a comprendere meglio le caratteristiche di queste classi.

Le classi che erano originariamente meno rappresentate tendono ad avere risultati peggiori, nonostante l'utilizzo di tecniche di oversampling sul training set abbia influenzato positivamente il classificatore.

6.1.2 Risultati con SVM

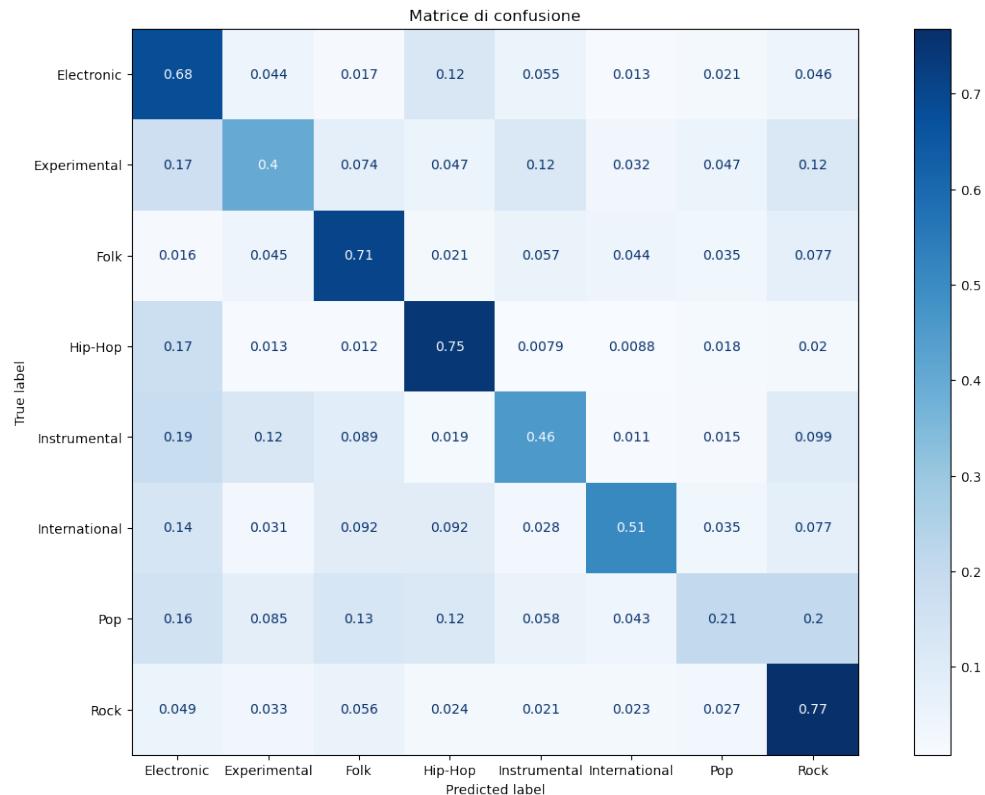


Figura 6.1: Matrice di confusione della validazione con SVM

Accuratezza: 0.6253

SVM					
Classe	Precisione	Recupero	F1-Score	Support	
Electronic	0.65	0.68	0.67	2104	
Experimental	0.43	0.40	0.41	739	
Folk	0.60	0.71	0.65	915	
Hip-Hop	0.62	0.75	0.68	1136	
Instrumental	0.47	0.46	0.46	718	
International	0.54	0.51	0.52	457	
Pop	0.42	0.21	0.28	765	
Rock	0.77	0.77	0.77	2337	
Media pesata	0.62	0.63	0.62	9171	

6.1.3 Risultati con Logistic Regression

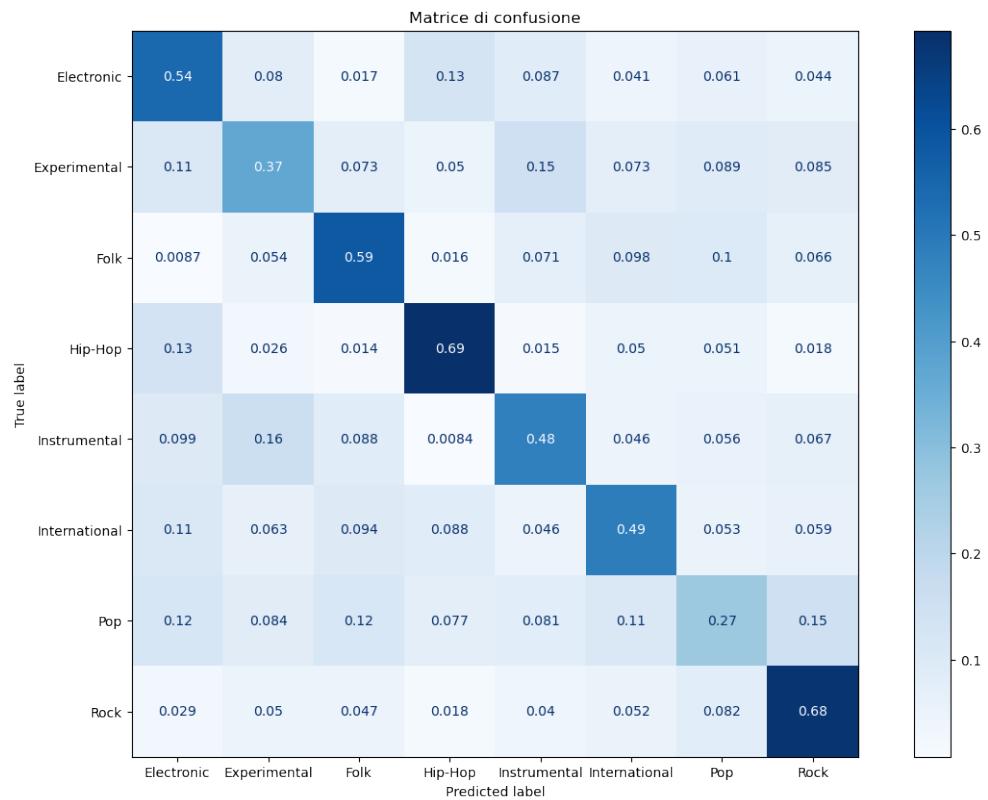


Figura 6.2: Matrice di confusione della validazione con Logistic Regression

Accuratezza: 0.5562

Logistic Regression				
Classe	Precisione	Recupero	F1-Score	Support
Electronic	0.69	0.54	0.61	2104
Experimental	0.32	0.37	0.35	739
Folk	0.57	0.59	0.58	915
Hip-Hop	0.63	0.69	0.66	1136
Instrumental	0.38	0.48	0.43	718
International	0.30	0.49	0.37	457
Pop	0.25	0.27	0.26	765
Rock	0.79	0.68	0.73	2337
Media pesata	0.59	0.56	0.57	9171

6.1.4 Risultati con k-nearest neighbors



Figura 6.3: Matrice di confusione della validazione con k-nearest neighbors

Accuratezza: 0.5007

k-nearest neighbors				
Classe	Precisione	Recupero	F1-Score	Support
Electronic	0.63	0.41	0.49	2104
Experimental	0.31	0.25	0.28	739
Folk	0.51	0.60	0.56	915
Hip-Hop	0.51	0.71	0.59	1136
Instrumental	0.41	0.34	0.37	718
International	0.43	0.46	0.44	457
Pop	0.20	0.33	0.25	765
Rock	0.68	0.63	0.65	2337
Media pesata	0.53	0.50	0.50	9171

6.1.5 Risultati con Random Forest

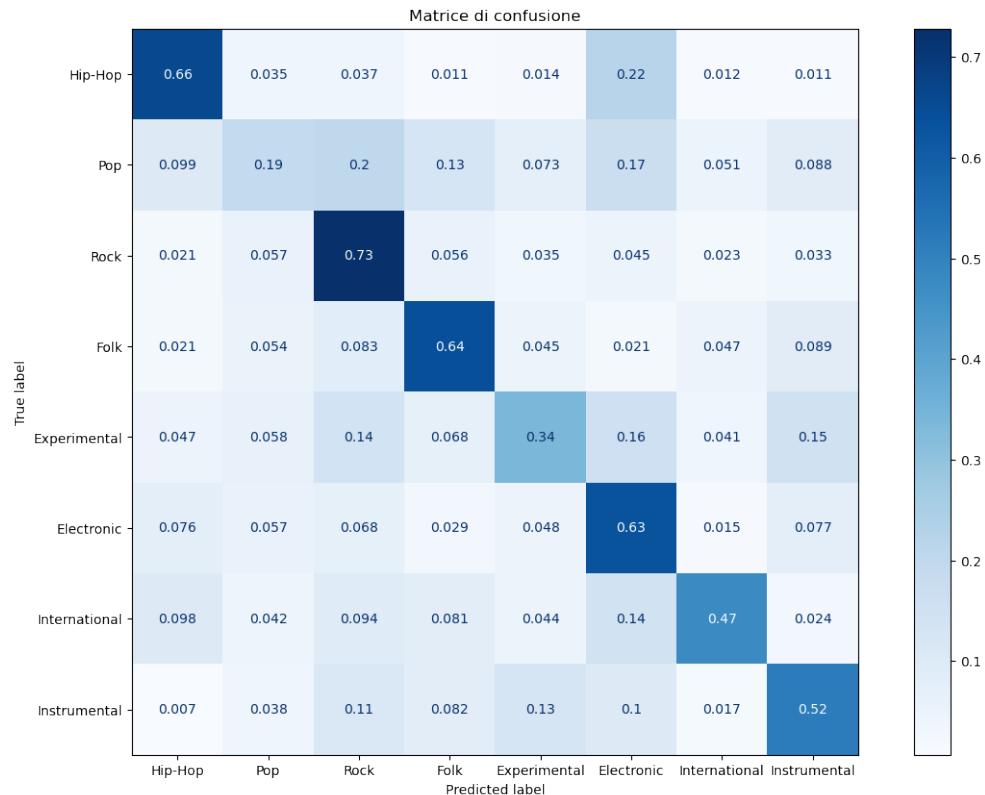


Figura 6.4: Matrice di confusione della validazione con Random Forest

Accuratezza: 0.5829

Random Forest					
Classe	Precisione	Recupero	F1-Score	Support	
Electronic	0.64	0.63	0.63	2104	
Experimental	0.38	0.34	0.36	739	
Folk	0.57	0.64	0.60	915	
Hip-Hop	0.66	0.66	0.66	1136	
Instrumental	0.41	0.52	0.46	718	
International	0.49	0.47	0.48	457	
Pop	0.25	0.19	0.22	765	
Rock	0.73	0.73	0.73	2337	
Media pesata	0.58	0.58	0.58	9171	

6.2 Visualizzazione grafica del dataset in uno spazio a due dimensioni (embedding)

t-SNE è una tecnica probabilistica di riduzione della dimensionalità non lineare per l'embedding di dataset ad alta dimensionalità in uno spazio a due o tre dimensioni, nel quale possono essere visualizzati tramite un grafico di dispersione. L'algoritmo modella i punti in modo che oggetti vicini nello spazio originale risultino vicini nello spazio a dimensionalità ridotta, e oggetti lontani risultino lontani, cercando di preservare la struttura locale. [20]

Utilizzando t-SNE è possibile passare da uno spazio vettoriale di 130 dimensioni ad uno spazio di 2 dimensioni. In questo modo è possibile visualizzare i campioni del dataset, ovvero le tracce musicali, come punti all'interno di un piano.

Se il dataset ha delle feature significative per la classificazione di generi musicali, le tracce musicali appartenenti allo stesso genere tenderanno ad essere raggruppate.

```
tsne = TSNE(early_exaggeration = 30.0, n_components=2,  
            verbose=1, perplexity=40, n_iter=1000)  
tsne_results = tsne.fit_transform(dataset)
```

Si sono estratti casualmente dal dataset 1000 campioni per ogni genere musicale, in modo da visualizzare più chiaramente i campioni delle classi meno rappresentate.

Si è poi voluto rappresentare l'intero dataset.

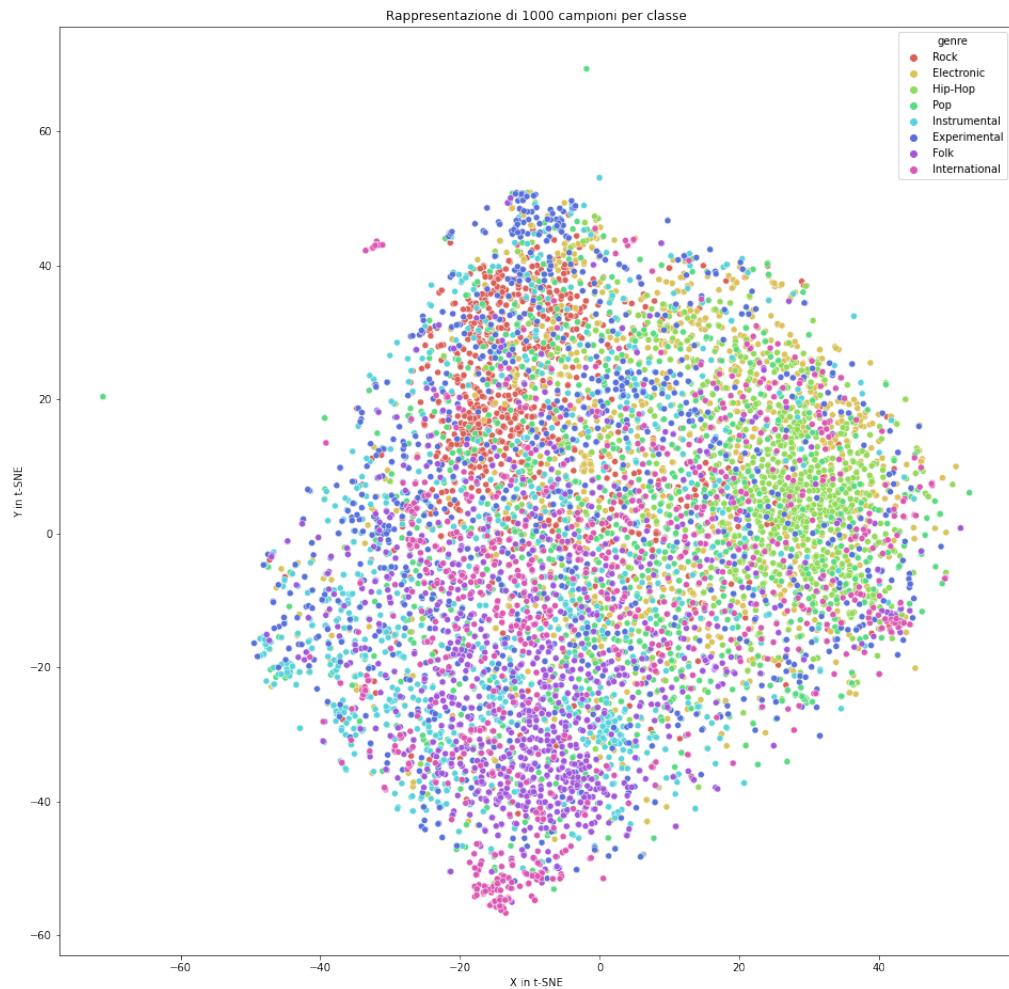


Figura 6.5: Visualizzazione grafica tramite t-SNE di 1000 campioni casuali per genere, estratti dal dataset. Si iniziano a notare alcuni cluster, zone in cui i campioni appartenenti alla stessa classe sono più densi. Il grafico è comunque disordinato a causa del basso numero di campioni preso in considerazione

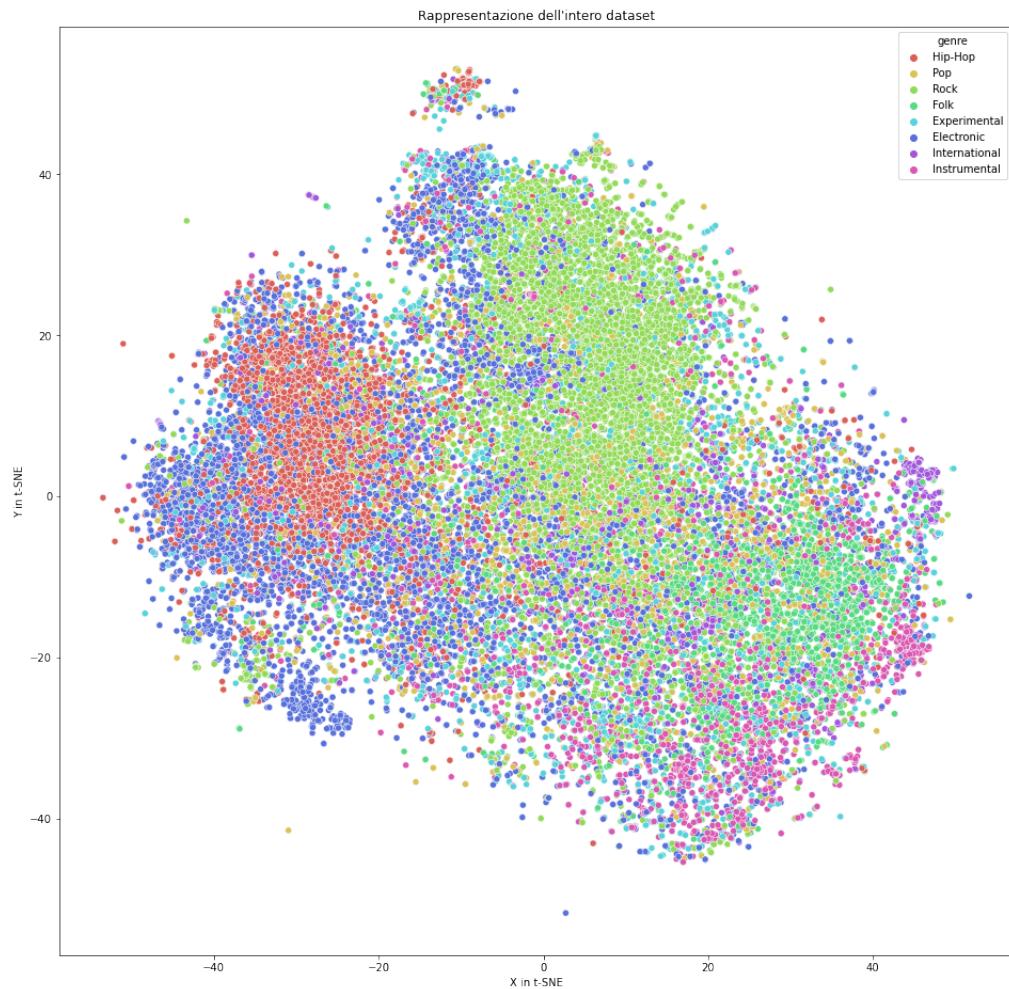


Figura 6.6: Visualizzazione grafica tramite t-SNE dell'intero dataset. Si nota come le classi meglio riconosciute dai classificatori, ovvero Rock, Hip-Hop e Folk (rispettivamente di colore verde chiaro, rosso e verde scuro) abbiano i campioni raggruppati in cluster ben definiti

Conclusioni e sviluppi futuri

L’obiettivo di ottenere un classificatore di generi musicali piuttosto efficace è stato raggiunto. Si è così dimostrato che, osservando proprietà fisiche di un audio, si possono determinare caratteristiche ad alto livello di astrazione, come il genere musicale. Per fare ciò il machine learning è sicuramente uno degli strumenti migliori.

Nell’ambito della classificazione di generi musicali ci possono essere diversi sviluppi futuri in grado di migliorarne i risultati. Un miglioramento significativo può essere l’utilizzo di un sistema di chord recognition. Un sistema che è in grado di distinguere gli accordi utilizzati porterebbe un grande vantaggio, in quanto i diversi tipi di scale e accordi sono fattori fondamentali nella distinzione dei generi musicali. Un altro miglioramento può essere quello di tenere in considerazione alcune delle informazioni spettrali correlate al tempo. Nel progetto di questa tesi, in cui si utilizzano funzioni come la media rispetto al tempo per ridurre il numero di dati, vanno persi alcuni pattern che potrebbero essere molto utili al riconoscimento del genere musicale di appartenenza. Ad esempio, dal cromagramma (o, meglio ancora, da un sistema di chord recognition) si potrebbe ricavare la progressione degli accordi presenti, ovvero l’ordine in cui essi si presentano. Un ultimo modo per migliorare l’efficacia del classificatore è quello di aumentare il numero di tracce musicali utilizzate per il training.

Bibliografia

- [1] Music information retrieval. https://en.wikipedia.org/wiki/Music_information_retrieval.
- [2] M. Schedl, E. Gómez, and J. Urbano. Music information retrieval: Recent developments and applications. *Foundations and Trends in Information Retrieval*, 8:128–140, 2014. https://repositori.upf.edu/bitstream/handle/10230/27565/Urbano_ftir_mus.pdf.
- [3] Apprendimento automatico. <https://it.wikipedia.org/wiki/ApprendimentoAutomatico>.
- [4] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31:249–268, 2007. <http://www.informatica.si/index.php/informatica/article/viewFile/148/140>.
- [5] Algoritmo support vector machine. <https://lorenzogovoni.com/support-vector-machine/>.
- [6] Come funziona un algoritmo di regressione logistica. <https://lorenzogovoni.com/regressione-logistica/>.
- [7] Modello logit. https://it.wikipedia.org/wiki/Modello_logit.
- [8] K-nearest neighbors. https://it.wikipedia.org/wiki/K-nearest_neighbors.

- [9] Come l'algoritmo random forest migliora le previsioni degli alberi decisionali. <https://lorenzogovoni.com/random-forest/>.
- [10] Python. <https://it.wikipedia.org/wiki/Python>.
- [11] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson. Fma: A dataset for music analysis. <https://github.com/mdefff/fma>.
- [12] Teorema del campionamento di nyquist-shannon. https://it.wikipedia.org/wiki/Teorema_del_campionamento_di_Nyquist-Shannon.
- [13] R. Di Federico. Short time fourier transform. *Analisi ed Elaborazione del Suono*, pages 4–15, 1999. <http://www.dei.unipd.it/~musica/Dispense/Cap4.pdf>.
- [14] Mel frequency cepstral coefficient (mfcc) tutorial. <http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/>.
- [15] Mfcc theory and implementation. <https://www.kaggle.com/ilyamich/mfcc-implementation-and-tutorial>.
- [16] Speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what's in-between. <https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html>.
- [17] Equal temperament. https://en.wikipedia.org/wiki/Equal_temperament.
- [18] E. Schubert and J. Wolfe. Does timbral brightness scale with frequency and spectral centroid? *ACTA ACUSTICA UNITED WITH ACUSTICA*, 92:820–825, 2006. <https://newt.phys.unsw.edu.au/~jw/reprints/SchubertWolfe06.pdf>.

- [19] The 5 feature selection algorithms every data scientist should know.
<https://towardsdatascience.com/the-5-feature-selection-algorithms-every-data-scientist-need-to-know-3a6b566efd2>.
- [20] t-distributed stochastic neighbor embedding.
https://it.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding.