UNIVERSITÀ DI PISA

Data Mining 2 - Project Report

Davide Perra

Giulia Calvo

MSc in Data Science and Business Informatics

FMA: A Dataset for Music Analysis

Anno accademico: 2020/2021

ABSTRACT

I dataset proposti per questa seconda parte di corso raccolgono dati sulle caratteristiche di brani appartenenti ad un archivio gratuito che può essere utilizzato liberamente per valutare diverse task utili nel recupero di informazioni musicali (MIR). Uno degli argomenti di ricerca del MIR classico è la classificazione del genere, che classifica gli elementi musicali in uno dei generi predefiniti come classica, jazz, rock, ecc. Quello che faremo sarà occuparci dei dati usando metodi di Imbalancing Learning e lavorare con gli Outlier. Useremo diversi tipi di algoritmi per classificare la classe 'genre_top'. Infine, ci occuperemo di analizzare dei Time Series data.

1. PREPARAZIONE DEL DATASET

1.1 Data Understanding

Per lo svolgimento di questo progetto, ci sono stati forniti 4 file, scaricabili tramite questo link: https://github.com/mdeff/fma.
Questi files sono così formati:

- tracks (106574 righe, 52 colonne);
- genres (163 righe, 4 colonne)
- features (106574 righe, 518 colonne)
- echonest (13129 righe, 249 colonne).

Sono pure scaricabili 106574 canzoni da 16341 artisti e 14854 album, con 917 GB di file e 343 giorni di musica creativa per un totale di 161 generi.
Ogni traccia viene identificata da un track_id, in modo che, durante la comparazione dei vari dataset, si riesca a rintracciare le varie caratteristiche di quella determinata canzone negli altri dataset.

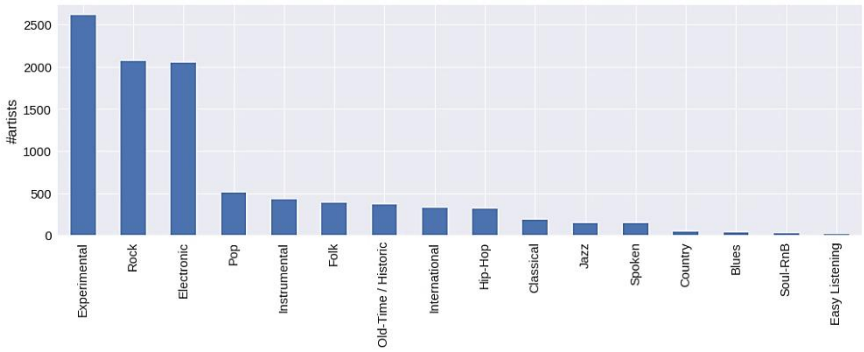
1)Tracks

Il file 'tracks' è un file multindex, quindi possiede diversi livelli di astrazione dei dati.
Sono presenti 4 principali suddivisioni:

Track	bit_rate, comments, composer, date_created ,date_recorded, duration,favorites, genre_top, genres, genres_all, information, interest, language_code, license, listens, lyricist, number, publisher, tags, title
Album	comments, date_created, date_released, engineer, favorites, id, information, listens, producer, tags, title, tracks, type
Artist	active_year_begin, active_year_end, associated_labels, bio, comments, date_created, favorites, id, latitude, location, longitude, members, name, related_projects, tags, website, wikipedia_page
Set	Split, subset

In Set sono presenti informazioni utili ai fini del progetto.
In particolare:

- split: in base allo scopo finale: *Train (80%), Validation (10%), Test (10%);*
- subset: in base alla grandezza del subset: *small: 8000, medium: 25000, large: 106574.*



2) Genres:

Il dataset è composto da 161 generi musicali all'interno della raccolta, di cui 16 sono quelli principali, i restanti sono subgeneri.
All'interno del file troviamo 4 colonne (#tracks, parent, title, top_level).

3)Features:
É un dataset che contiene features estratte usando la libreria librosa; quest'ultima descrive le caratteristiche di ogni canzone in base a "Spectral features":

- chroma_cens: Calcola la variante di crominanza 'Chroma Energy Normalized' (CENS)
- chroma_cqt: Cromagramma a Q costante
- chroma_stft: Calcola un cromagramma da una forma d'onda o uno spettrogramma di potenza
- mfcc: Coefficienti cefalici di frequenza mel (MFCC)
- rmse: Calcola il valore RMS (root-mean-square) per ogni fotogramma, dai campioni audio y o da uno spettrogramma S.
- spectral_bandwidth: Calcola la larghezza di banda spettrale del p'ordine.
- spectral_centroid: Calcola il centroide spettrale.
- spectral_contrast: Calcola il contrasto spettrale
- spectral_rolloff: Calcola la frequenza di roll-off.
- tonnetz: Calcola le caratteristiche del centroide tonale (tonnetz)
- zcr (zero_crossing_rate): Calcola il tasso di passaggio per lo zero di una serie temporale audio.

Ognuna di queste features viene caratterizzata da 7 tipologie di statistiche: kurtosis, min, mean, max, median, skew, std.
Tenendo conto delle informazioni che siamo riusciti a reperire, mentre l'orecchio umano è allenato a riconoscere features di alto livello come ritmo, melodia, genere in maniera molto naturale, questo tipo di features vengono estratte da librosa ad un livello di astrazione medio/basso, analizzando quindi dei suoni che abbiano un senso per le macchine.
Attraverso queste informazioni di medio/basso livello, il computer riesce a categorizzare il tipo di audio e a percepire quello che per gli umani è più facile da comprendere.

4) Echonest
È una collezione di dati musicali ma anche una piattaforma musicale posseduta da Spotify, sin dal 2014, che si occupa di utilizzare degli algoritmi per poter individuare delle canzoni sempre nuove agli utenti di una piattaforma musicale basandosi sui gusti personali di ogni utente.
Nel nostro caso, ci viene fornito un dataset contenete ulteriori features che utilizzeremo più vanti durante il progetto.

1.2. SCELTA DELLA TASK

Tenendo conto che il dataset riguarda una raccolta di brani copyright-free e sapendo che questo tipo di brani vengono solitamente utilizzati per dei contenuti nelle piattaforme digitali.
Abbiamo pensato di focalizzare la nostra attenzione sul riconoscimento delle canzoni che hanno come "genre_top" *Old Time/Historic*, così da evitare di proporlo a coloro che utilizzano la raccolta.
La nostra scelta è giustificata dal fatto che raramente all'interno dei social i contenuti sono associati a brani di vecchio stampo, quindi nell'ottica di usare questo dataset per ottimizzare l'esperienza dell'utente all'interno del sito.

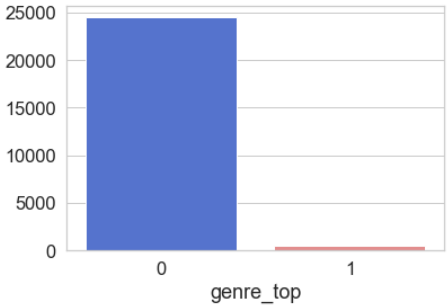
1.3. DATA PREPARATION

Per lo svolgimento della task da noi ideata, abbiamo creato un dataset ad hoc per svolgerla al meglio.
Abbiamo estratto dal dataset "Tracks" gli attributi "genre_top", "split" e "subset", mentre da "features" abbiamo selezionato "spectral_centroid" e "spectral_rolloff".
Abbiamo rinominato le colonne e le statistiche riguardanti "spectral rolloff" saranno precedute dal nominativo "roll", e da "centr" per quelle riguardanti "spectral centroid".
Dopodiché li abbiamo uniti tramite il track_id, ottenendo un dataset con 14 features:

'split', 'subset', 'genre_top', 'roll_kurtosis', 'roll_max', 'roll_mean', 'roll_median', 'roll_min', 'roll_skew', 'roll_std', 'centr_kurtosis', 'centr_max', 'centr_mean', 'centr_median','centr_min', 'centr_skew', 'centr_std'.

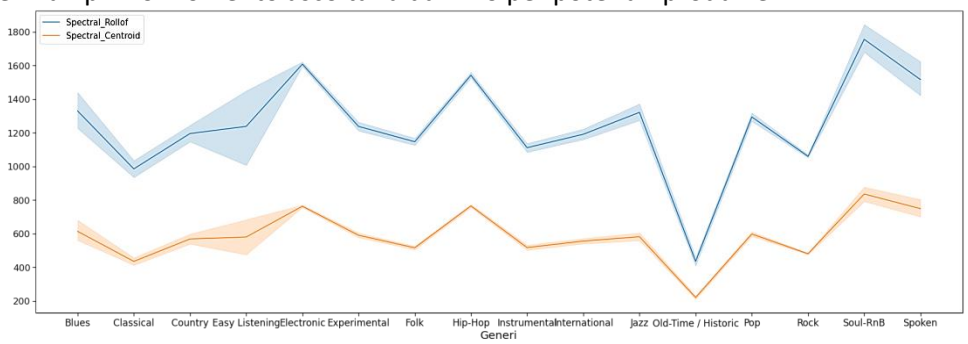
Successivamente, abbiamo deciso di concentrarci solamente sulle tracce facenti parte del subset medium e, dopo aver diviso in training, validation e test, abbiamo ottenuto:

- 19922 training samples;
- 2505 validation samples;
- 2573testing samples.



Abbiamo quindi selezionato come Target Variable *Y= 'genre top'* e in seguito utilizzato un label encoding per distinguere il genere target (Old Time/ Historic, etichettato come 1) da tutti gli altri generi, etichettati come 0, creando così un binary classification problem.

Per classificare correttamente il genere ci siamo affidati alle features descritte precedentemente: in particolare abbiamo usato "spectral_centroid", che è comunemente associato alla misura della luminosità di un suono, e "spectral_rolloff" che è la frequenza al di sotto della quale è contenuto il 99% dell'energia dello spettro.
In base alle nostre ricerche, sappiamo per certo che il genere target che stiamo cercando di classificare è fondato su canzoni riprodotte durante le feste di paese con strumenti principalmente a corda. Inoltre, è un tipo di genere difficile da riprodurre usando dei semplici spartiti in quanto chi si accinge a questo tipo di musica deve in un primo momento ascoltarla dal vivo per poterla riprodurre.



Abbiamo quindi deciso di utilizzarli in quanto "spectral_centroid' riesce a catturare molto bene le frequenze medio-alte di tutti i generi musicali mentre "spectral_rolloff" mette in evidenza quali sono i generi con una distribuzione energetica del suono maggiore.

In base a questo grafico notiamo che utilizzando la standard deviation delle due features scelte, riusciamo a distinguere il genere "Old Time/Historic" rispetto gli altri.

1.4 OUTLIER DETECTION

L'oggetto di questa sezione è di identificare il top 1% degli outliers.
I metodi che sono stati adottati a tale scopo sono stati il Local Outlier Factor, Angle Based Outlier Detection e Isolation Forest.
Per rappresentare gli outliers individuati all'interno del dataset, abbiamo usato due variabili, "roll_skew" e "centr_skew".
Abbiamo preferito non ricorrere al Principal Component Analysis in quanto, anche se riscontrato essere molto positivo in altri punti del nostro progetto, per quanto riguardala rappresentazione degli outliers con diverse metodologie, graficamente non riusciva a rappresentare al meglio le diversità tra le varie tecniche.
Le due variabili selezionate hanno fornito il miglior risultato in termini di visualizzazione degli outliers in due dimensioni e sono state scelte dopo una rigorosa analisi di tutte le possibili copie di attributi.

Per settare i parametri di ogni metodo, abbiamo deciso di ricorrere a una Grid Search utilizzando una cross validation pari a 5, ma non abbiamo visto miglioramenti esponenziali tra quelli trovati con quest'ultima e i valori di default, quindi, per semplicità, abbiamo deciso di utilizzare questi ultimi.

1.4.1. Isolation Forest

L' isolation Forest è un tipo di outlier detector che genera diversi alberi e i punti che mediamente si trovano all'inizio saranno quei punti considerati outlier per questo tipo di detector.
Essendo questo metodo efficiente dal punto di vista computazionale e capace di lavorare con dati di grosse dimensioni abbiamo deciso di utilizzarlo all'interno della nostra analisi.
Tramite l'isolation Forest abbiamo ottenuto un numero di errori pari 1993.

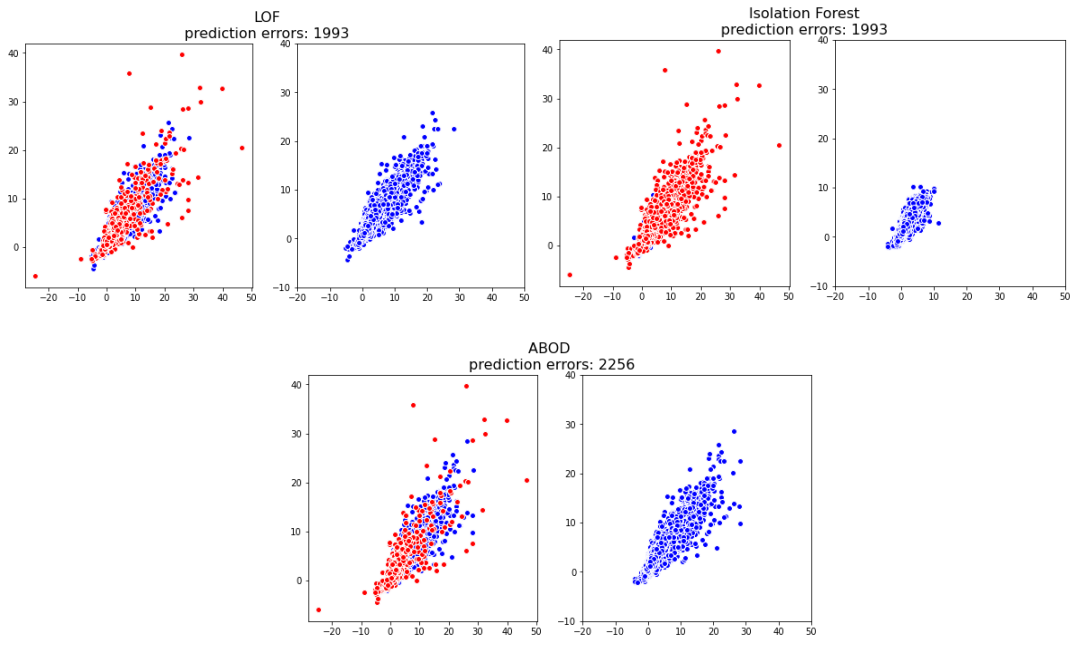
1.4.2. L'Angle Based outliers detection (ABOD)

L'Angle Based outliers detection fa parte degli angle-based approach e questo genere di outlier detector riesce ad essere molto più stabile con distanze in grandi spazi dimensionali perché parte dal presupposto che gli outliers si trovano ai bordi delle distribuzioni dei dati.
Considera un oggetto outlier se la maggior parte degli altri oggetti accanto a questo si trovano in direzioni simili. Anche se dal punto di vista computazionale è costoso, siamo riusciti ad individuare molte più anomalie rispetto al precedente detector, per un totale di 2256 outlier.

1.4.3. Local Outlier Factor (LOF)

Spostandoci su un detector basato sulla densità abbiamo deciso di utilizzare un Local Outlier Factor: quest'ultimo calcola la local density deviation di un dato punto rispetto ai suoi vicini, considerando outlier i samples che hanno una bassa densità rispetto i loro neighbors.
Abbiamo scelto questo tipo di outlier detector in quanto in grado di vedere gli outliers locali, nonostante uno dei contro sia che questo tipo di approccio da come output un punteggio e non gli outlier veri e propri.
I punti che sono stati classificati come outliers con questo detector sono 1993.

Qui sotto mostriamo prima il dataset originale in cui individuiamo in rosso i punti individuati come outliers e affianco il dataset pulito in cui abbiamo rimosso gli outliers identificati precedentemente.



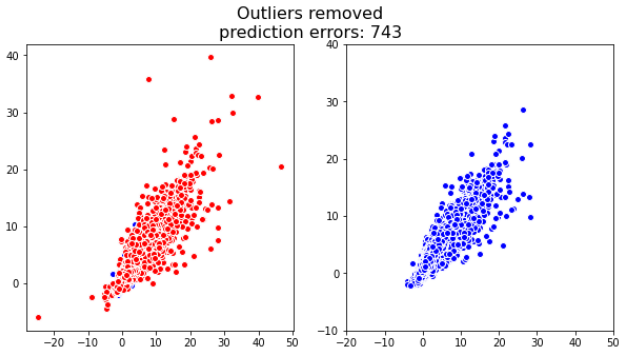
Confrontando i risultati ottenuti dai 3 algoritmi notiamo che tra gli outliers individuati:

- LOF e Isolation Forest hanno in comune 460 outliers;
- LOF e ABOD hanno in comune 845 outliers;
- Isolation Forest e ABOD hanno in comune 743 outliers;
- LOF, Isolation Forest e ABOD hanno in comune 316 outliers.

Nonostante la natura diversa dei tre algoritmi, possiamo notare dai grafici che l'ABOD e il LOF hanno evidenziato come outliers punti molto simili, mentre l'isolation forest ha mostrato una concentrazione di outliers nelle misurazioni prese nell'asse delle ordinate a partire da un valore di 10.

Differenti tecniche sono basate su differenti assunzioni. Quindi non possiamo dire che un metodo è migliore rispetto ad un altro in quanto ciascuno individua differenti risultati in base a quel determinato approccio.
Una buona prassi, quindi, è fare una comparazione tra più metodologie.
Detto questo, abbiamo deciso di rimuovere i 743 outliers trovati tramite l'utilizzo del ABOD e del isolation Forest. Non abbiamo considerato pure quelli individuati dal LOF in quanto questo risulta essere meno efficiente quando vengono trattate più dimensioni, mentre l'Isolation Forest e l'ABOD sono approcci adatti in un high dimensional dataset come il nostro.
Abbiamo deciso di rimuovere gli outliers invece di sostituirli con un valore (come, ad esempio, la media) in quanto, considerando il fatto che stiamo utilizzando molti dati, rimuoverli non danneggerà il nostro dataset.

Nei paragrafi 2 e 3 tratteremo vari classificatori e verrà utilizzato il dataset pulito dei 743 individuati.



2. CLASSIFICATION METHODS

2.1 DECISION TREE

Nella seguente sezione descriveremo la metodologia usata per costruire il decision tree in grado di predire se un genere musicale e Old Time-Historic o no.

2.1.1 Data preparation

Prima di costruire il nostro modello dobbiamo verificare la composizione del dataset il quale è già stato suddiviso in training (80%), validation(10%) e test(10%).

Ci siamo occupati di vedere quante tracks sono Old Time/ Historic all'interno nostro dataset:

Training set		Validation set		Test set	
Old Time – Historic	371	Old Time – Historic	51	Old Time – Historic	51
Altri generi	18808	Altri generi	2454	Altri generi	2522

Possiamo notare che è presente uno sbilanciamento nei dati che influenzerà i nostri classificatori nel predire la classe corretta.

Questo aspetto verrà curato nella sezione 2.3.

2.1.2 Scelta Hyperparameter

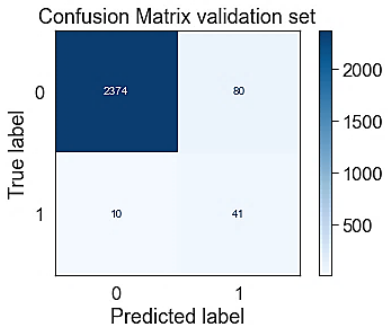
Nel settare i parametri ottimali abbiamo utilizzato differenti k-fold cross validation sul training set. Abbiamo successivamente selezionato il miglior modello e lo abbiamo riallenato su tutto il training set.

Qui sotto vengono riportati i 3 migliori risultati ottenuti in termini di accuracy con un 3-fold e 5-fold cross validation e il miglior modello per ogni cross validation; In giallo i risultati migliori:

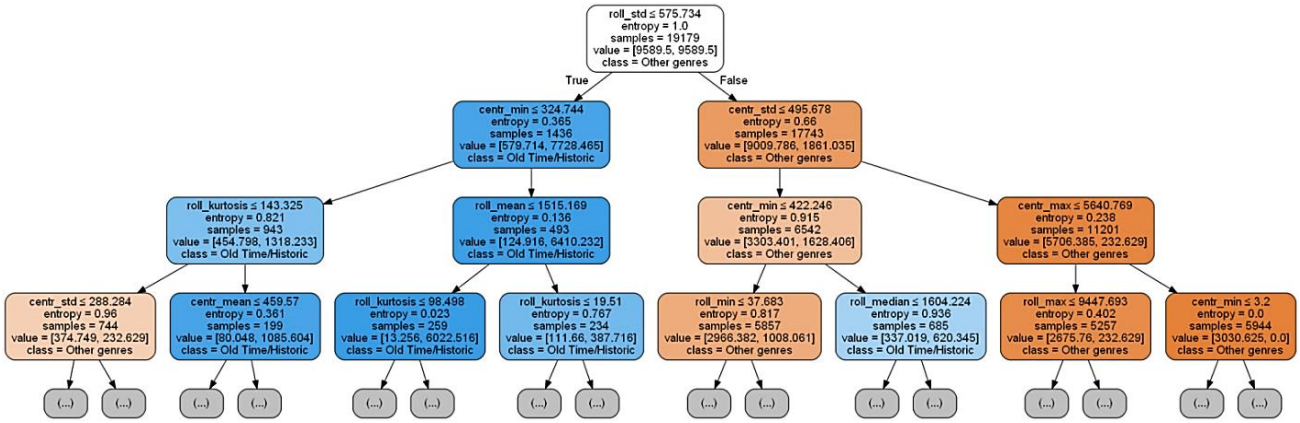
3-fold cross validation	
parameters	mean validation score
criterion:gini min_sample_split:3 min_sample_leaf:3 max_deph:9	0.978
criterion:gini min_sample_split:9 min_sample_leaf:3 max_deph:9	0.977
criterion:gini min_sample_split:6 min_sample_leaf:6 max_deph:9	0.976

5-fold cross validation	
parameters	mean validation score
criterion:entropy min_sample_split:2 min_sample_leaf:3 max_deph:9	0.979
criterion:entropy min_sample_split:9 min_sample_leaf:2 max_deph:9	0.978
criterion:entropy min_sample_split:4 min_sample_leaf:4 max_deph:9	0.978

Successivamente abbiamo testato la performance del modello sul validation set; abbiamo scelto di utilizzare il risultato del 5-fold cross validation in quanto ci ha dato una migliore performance ottenendo un accuracy del 96,40%



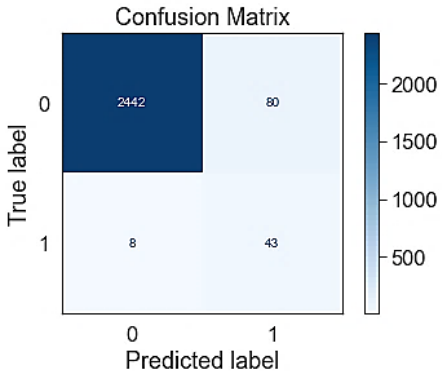
	Precision (%)	*Recall (%)*	*f1-score (%)*
Other genres	100	97	98
Old Time	34	80	48
Average scores	67	89	73
Accuracy (%)	96		

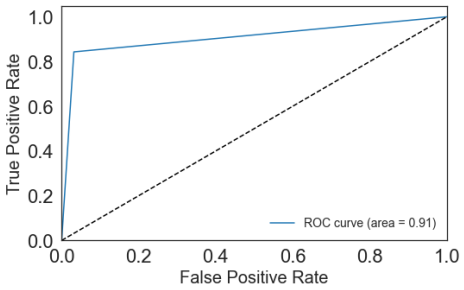


2.1.3 Blind Test e Risultati

L'accuracy ottenuta nel test set (0,9657) si avvicina notevolmente a quella stimata nel validation set. Dalla confusion matrix possiamo notare che il nostro classificatore ha predetto correttamente 43 canzoni con Old Time/Historic, mentre 8 no.

La ROC curve mostra un AUC score del 91 %, quindi possiamo dire che il Decision Tree ha avuto una performance abbastanza buona frutto soprattutto della cross validation utilizzata.





	Precision (%)	*Recall (%)*	*f1-score (%)*
Other genres	100	97	98
Old Time	35	84	49
Average scores	67	91	74
Accuracy (%)	97		

2.2 K-NEAREST NEIGHBOUR

Il secondo classificatore che andremo a trattare è il K-Nearest Neighbour.

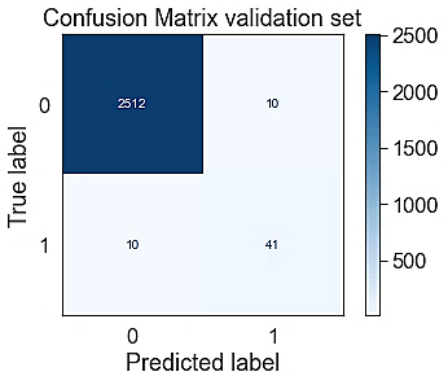
Abbiamo utilizzato 14 dimensioni. Sapendo che il KNN è un classificatore che soffre della Curse of dimensionality, abbiamo provato a ridurre le dimensioni da 14 a 2 con il metodo della PCA, ma i risultati sul training originale (senza PCA) sono risultati migliori. Successivamente, li abbiamo normalizzati con un Robust Scaler.

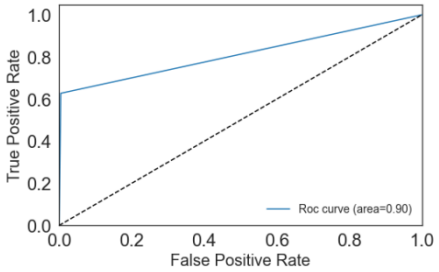
Per selezionare il miglior parametro K, abbiamo usato un 5-fold cross validation e successivamente abbiamo analizzato i risultati ottenuti.

Il set di parametri usati nella cv sono i seguenti:

parameters configuration	
n_neighbors	range(2,20)
distance	Euclidean, Manhattan
weight	uniform, distance

La miglior configurazione ottenuta è stata quella con la Mahattan distance, k=11 e un distance weight.
Con queste impostazioni, abbiamo ottenuto un mean validation score del 99% (in termini di accuracy). Abbiamo poi allenato il nostro modello usando i parametri elencati sopra e analizzato i risultati ottenuti sul test set.
Le tabelle sotto riassumono i risultati.





	Precision (%)	*Recall (%)*	*f1-score (%)*
Other genres	100	100	100
Old Time	80	80	80
Average scores	90	90	90
Accuracy (%)	99		

Questo modello sembrerebbe distinguere le due classi meglio rispetto al Decision Tree. Abbiamo ottenuto abbastanza positivi sia nella validation, che nel test, in cui quest'ultimi si sono rivelati notevolmente migliori rispetto i primi anche se l'utilizzo della Cv non ci ha dato risultati buoni come quelli del Decision Tree.

2.3. IMBALANCED LEARNING TECHNIQUES

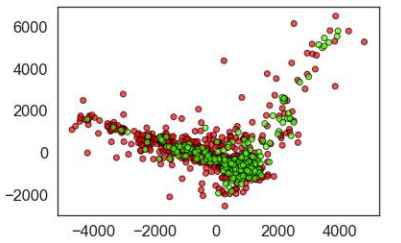
Come abbiamo già potuto vedere nella sezione 2.1, il training set è altamente sbilanciato rispetto al nostro target attribute object della classificazione.
Abbiamo deciso di risolvere questo problema usando differenti imbalanced learning techniques.
Le strategie adottate sono due:

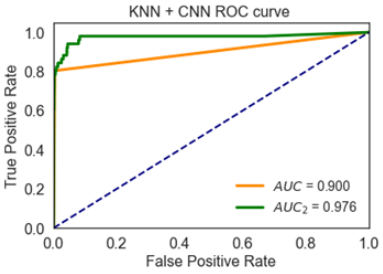
- utilizzare metodo di Undersampling: Random Undersampling e Condensed Nearest Neighbor.
- utilizzare metodo di Oversampling: Random Oversampling e Synthetic Minority Oversampling Technique.

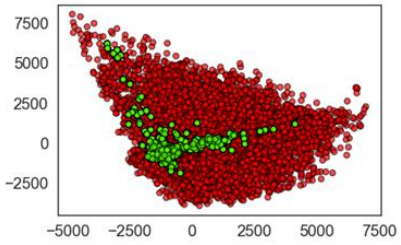
Queste tecniche aumentano/diminuiscono il numero totale di dati in modo di ottenere un bilanciamento nel dataset e sono state applicate sul training set in modo che, l'introduzione dei dati artificiali creati, non andasse ad introdurre un qualche bias rendendo poco significativi gli score del test set.

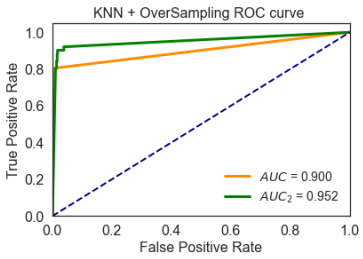
La tabella sotto mostra come il miglior approccio per questa task risulti essere il KNN senza nessuna classe bilanciata, ottenendo un F1-score pari a 0.90.
Se volessimo considerare però la tecnica di imbalance che ha riscontrato un migliore trade off tra fp e fn, risulterebbe essere il KNN applicando il Condense Nearest Neighbor, con una ROC curve di 0.98.
Ma, possiamo comunque affermare che il Random Oversampling ha avuto una performance abbastanza buona in entrambi i classificatori.

	Accuracy	Precision		AVG	Recall		AVG	F1-Score		AVG	AUC
		0	1		0	1		0	1		
Decision Tree	0.97	1.00	0.35	0.67	0.97	0.84	0.91	0.98	0.49	0.74	0.91
Decision Tree + Undersampling	0.90	1.00	0.14	0.57	0.90	0.84	0.87	0.94	0.25	0.59	0.87
Decision Tree + CNN	0.92	1.00	0.18	0.59	0.92	0.90	0.91	0.95	0.30	0.63	0.93
Decision Tree + OverSampling	0.97	1.00	0.36	0.68	0.97	0.84	0.91	0.98	0.50	0.74	0.92
Decision Tree + SMOTE	0.94	1.00	0.23	0.61	0.94	0.84	0.89	0.97	0.36	0.66	0.92
KNN	0.99	1.00	0.80	0.90	1.00	0.80	0.90	1.00	0.80	0.90	0.90
KNN + Undersampling	0.90	1.00	0.15	0.58	0.90	0.90	0.90	0.95	0.26	0.60	0.96
KNN + CNN	0.98	1.00	0.58	0.79	0.99	0.82	0.91	0.99	0.68	0.84	0.98
KNN + OverSampling	0.98	1.00	0.35	0.68	0.97	0.90	0.93	0.98	0.51	0.75	0.95
KNN + SMOTE	0.95	1.00	0.28	0.64	0.95	0.92	0.94	0.98	0.43	0.70	0.98









Dopo aver sperimentato diverse tecniche di outlier detector e imbalance learning, abbiamo notato come queste, nonostante sembrano molto efficaci, non ci diano differenze esponenziali rispetto all'utilizzo del dataset principale.

C'è anche da dire che, per quanto riguarda gli outliers, abbiamo solamente rimosso 1%; ciò implica che solo una piccola parte tra tutte le anomalie è stata rimossa.

Abbiamo successivamente provato a rimuovere tutti gli outliers presenti sul dataset, ma ciò ha portato a ridurre il nostro target value a sole 5 istanze in tutto il training set.

Siamo quindi arrivati alla conclusione che, anche se la regola generale ci porta ad eliminare tutti gli outliers all'interno del dataset, in certi casi rimuovere solamente una piccola percentuale rispetto a tutte le anomalie potrebbe essere un buon trade off tra pulizia del dataset e mantenerlo intatto.

3. CLASSIFICATION METHODS ALTERNATIVI

In questa parte del progetto andremmo a risolvere la nostra classification task utilizzando altri classificatori trattati durante il programma di Data Mining 2: Naive Bayes Classifier, Logistic Regression, Rule-based Classifiers, Support Vector Machines, Neural Networks, Ensemble Methods.

Valuteremo ogni metodo di classificazione con le stesse tecniche impiegate col Decision Tree e KNN, e successivamente, andremo a capire quale classificatore sembri più adatto per il nostro obiettivo.

Infine, ideeremo una nuova task per risolvere un regressor problem.

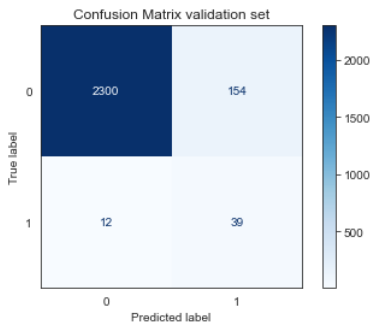
3.1 NAIVE BAYES CLASSIFIER

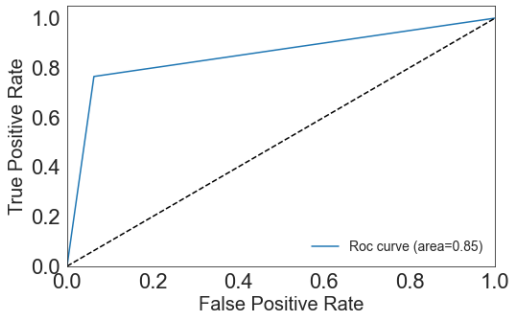
Questo tipo di classificatore si distingue dagli altri in quanto basato sulle probabilità, infatti riusciamo a classificare un nuovo record tenendo conto della probabilità che è un certo evento avvenga.

Tenendo conto di ciò che si è visto a lezione non abbiamo settato nessun parametro diverso da quello di default. Abbiamo utilizzato la libreria sklearn.naive_bayes: quest'ultima ci permette di classificare sia utilizzando la distribuzione gaussiana delle probabilità sia utilizzando attributi categorici.
I risultati ottenuti usando entrambi i metodi sono completamente differenti.

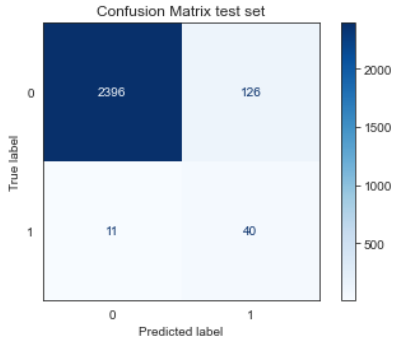
Utilizzando GaussianNB() ed allenando i dati usando questa libreria siamo riusciti ad ottenere un buon risultato su tutti i set.

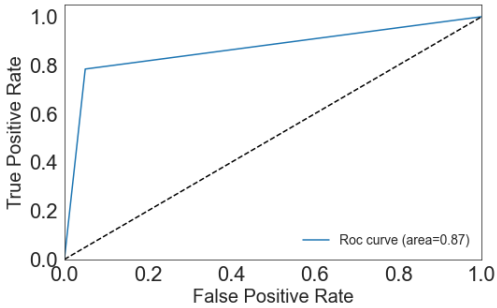
Utilizzando il validation set abbiamo ottenuto quanto segue: siamo riusciti a classificare correttamente 2339 records , classificato 12 Records come Old Time nonostante quest'ultimi fossero tracks di altro genere mentre abbiamo classificato 154 tracks come genere differente da old time anche se in realtà è proprio old time. Il risultato rispetto al training non cambia di molto, infatti troviamo una Roc Curve con un'area pari a 0,85





Come ci aspettavamo in base a ciò che si era visto con il validation, anche in questo caso i risultati si discostano di poco.





	Precision (%)	*Recall (%)*	*f1-score (%)*
Other genres	100	95	97
Old Time	24	78	37
Average scores	62	87	67
Accuracy (%)	*95*		

Creando un dataset con soli attributi categorici ed utilizzando CategoricalNB purtroppo non siamo riusciti a classificare ma come visto a lezione era qualcosa che poteva capitare in quanto questo tipo di classificatore

facente parte della libreria sklearn.naive_bayes non è in grado di lavorare con attributi categorici. Infatti, abbiamo ottenuto una ROC curve con area pari ad uno procedendo con una classificazione usando attributi categorici.

I risultati ottenuti con questo tipo di classificatore rispecchiano totalmente ciò che si è visto per la teoria: visto che questo tipo di classificatore è basato su un teorema che richiede una forte indipendenza tra le features e tenuto conto che le features utilizzate per la classificazione hanno comunque dei valori alti in entrambi i casi, per quanto siamo riusciti a classificare tutti gli altri generi abbiamo riscontrato dei problemi nel classificare correttamente il genere che ci interessa.

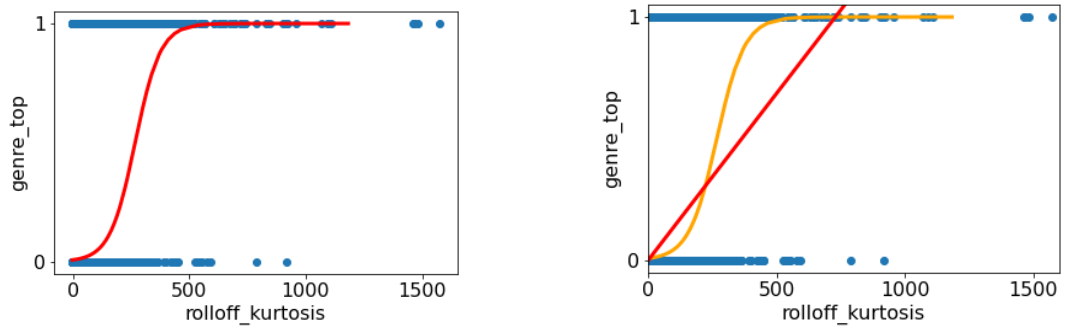
3.2 LOGISTIC REGRESSION

Essendo la regressione lineare un tipo di classificatore che predice anche valori non nel predominio, possiamo utilizzare la logistic regression per classificare la nostra target variable.

Utilizzando la libreria sklearn.linear_model e importando 'LogisticRegression' abbiamo cercato di ottenere dei risultati analizzando feature per feature, tra tutte le 14 che abbiamo deciso di utilizzare.

Abbiamo quindi plottato i risultati ma l'unica feature in grado di aiutarci nella classificazione si è rivelata la feature "rolloff_kurtosis".

Questo è il risultato con la Logistic Regression:



Anche se non si vede nel grafico sappiamo che l'intercetta del regressor è -4.5851 mentre il coefficiente è pari a 0,0128. Supponendo di aumentare di 100 unità il valore di "rolloff_kurtosis" sappiamo in base a questo classificatore che c'è lo 0,0353% di probabilità in più che la track che stiamo analizzando è di tipo Old-Time.

Purtroppo, i risultati sul test set (ma anche sugli altri set) con questo tipo di classificatore sono stati molto deludenti (cosa che ci aspettavamo tenendo conto della teoria):

2515 ***True label =0*** ***Predicted label = 0***	***7*** ***True label =0*** ***Predicted label = 1***
39 ***True label =1*** ***Predicted label = 0***	***12*** ***True label =1*** ***Predicted label = 1***

	Precision (%)	*Recall (%)*	*f1-score (%)*
Other genres	98	100	99
Old Time	63	24	34
Average scores	81	62	67
Accuracy (%)	*98*		

3.3 RULE BASED CLASSIFIER

Visto che la nostra task si basa su valori continui non siamo riusciti ad utilizzare nessun'altra libreria al di fuori di ripper. Quest'ultima si occupa di creare delle regole per poter classificare. Nel nostro caso la libreria

ha creato delle regole su valori continui che ci permettessero di classificare al meglio il 'genre_top' Ricordiamo sempre che la variabile target 1 è 'Old-Time/ Historic'

Validation Set:

	Precision (%)	*Recall (%)*	*f1-score (%)*
Other genres	99	99	99
Old Time	65	63	64
Average scores	82	81	81
Accuracy (%)	*99*		

2437 True label =0 Predicted label = 0	***17 True label =0 Predicted label = 1***
19 True label =1 Predicted label = 0	***32 True label =1 Predicted label = 1***

Test set:

	Precision (%)	*Recall (%)*	*f1-score (%)*
Other genres	99	99	99
Old Time	68	71	69
Average scores	83	85	84
Accuracy (%)	*99*		

2505 True label =0 Predicted label = 0	***17 True label =0 Predicted label = 1***
15 True label =1 Predicted label = 0	***36 True label =1 Predicted label = 1***

Nonostante ci aspettassimo dei risultati peggiori, siamo comunque soddisfatti.
I risultati ottenuti con questo classificatore risultano stranamente discreti tenendo conto dell'utilizzo di attributi continui.

3.4 SUPPORT VECTOR MACHINE (SVM)

3.4.1 NonLinear

Per capire quali hyper parameter migliori usare, abbiamo utilizzato una Grid Search, con i seguenti parametri:

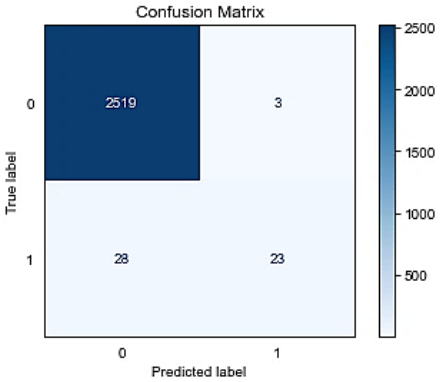
- C:0.001, 50.0, 100.0, 500.0, 1000.0
- kernel: poly, rbf, sigmoid

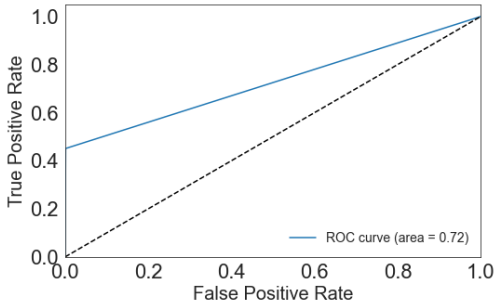
I migliori parametri sono stati C=1000.0, con Radial basis function kernel, ottenendo così un Mean validation score di 0.992. Abbiamo comunque ottenuto risultati insoddisfacenti; infatti quelli del validation set mostrano che il nostro classificatore non riesce a distinguere le due classi, attribuendo ogni istanza alla classe 0 (altri generi); infatti, non riconosce nessuna canzone con genere Old Time. Ciò può essere confermato dalla confusion matrix in cui TN corrispondono a 0, contro i FP che equivalgono a 51, e dalla roc_curve con un auc pari a 0.5.

3.4.2 Linear

Come per il metodo precedente, anche qui abbiamo utilizzato una Grid Search per cercare il parametro ottimale di C. In questo caso si è rivelato essere 40, ottenendo un mean validation score pari a 0.983.
Durante la valdation phase, siamo riusciti ad ottenere risultati migliori rispetto al SVM non linear.

Per questo motivo, abbiamo deciso di utilizzare il parametro trovato inizialmente per il test, ottenendo i seguenti risultati:





	Precision (%)	*Recall (%)*	*f1-score (%)*
Other genres	99	100	99
Old Time	88	45	60
Average scores	94	72	80
Accuracy (%)	99		

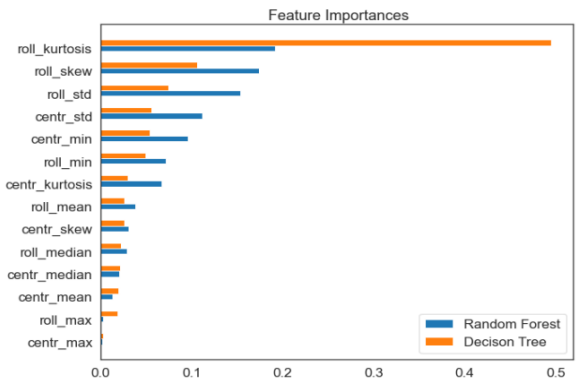
Sebbene i risultati siano migliori di quelli ottenuti durante l'allenamento del modello, il nostro classificatore non riesce bene a distinguere la classe positiva, mentre riesce a farlo perfettamente per la classe 0; infatti mentre solamente 3 vengono misclassificate dalla classe 0, nella nostra classe di interesse 28 su 51 (più della metà) non vengono classificate correttamente. Ciò, infatti, si traduce con una roc curve avente un auc uguale a 0.72. Questo risultato ci porta a pensare che i nostri dati siano separabili in maniera lineare.

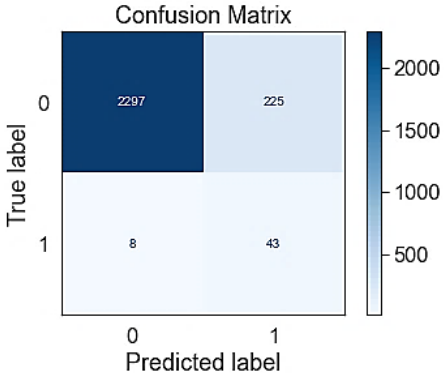
3.5 Random Forest

In questa sezione andremo a discutere dei risultati ottenuti sul dataset con un metodo di ensemble learning, il Random Forest.

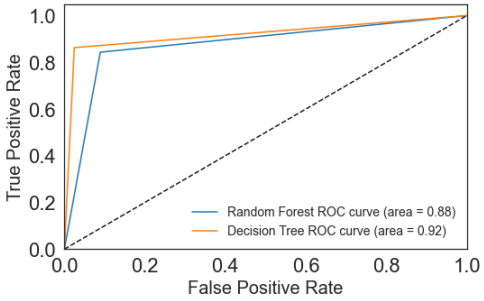
Abbiamo confrontato i risultati ottenuti dal Decision Tree con quelli del Random Forest utilizzando lo stesso sottoinsieme di attributi.
Nel caso del Random Forest, gli iperparametri considerati sono: N° Estimators, Criterion, Max features, Max depth, Min samples split, Min saples leaf.
Tramite una RandomSearch con una cross validation pari a 5, i migliori hyperparameters sono risultati:

'n_estimators': 300, 'min_samples_split': 6, 'min_samples_leaf': 10, 'max_features': 'sqrt', 'max_depth': 2, 'criterion': 'entropy', 'class_weight':' 'balanced_subsample'





Score	*Decision Tree*	*Random Forest*
Accuracy	0.97	0.91
Precision	0.67	0.58
Recall	0.91	0.88
F1	0.74	0.61
ROC	0.92	0.88



Possiamo notare come entrambi i classificatori diano la stessa importanza alle features presenti nel dataset; in particolare la kurtosis, skew e standard deviation della spectral rolloff sembrerebbero le più rilevanti per entrambe le metodologie.
Nonostante ciò, il Decision Tree sembrerebbe un classificatore migliore rispetto al Random Forest; questo, infatti, pur predicendo lo stesso numero di false positive, ha un numero troppo elevato di false negative, ovvero 225 contro gli 80 che ha individuato il Decision Tree.

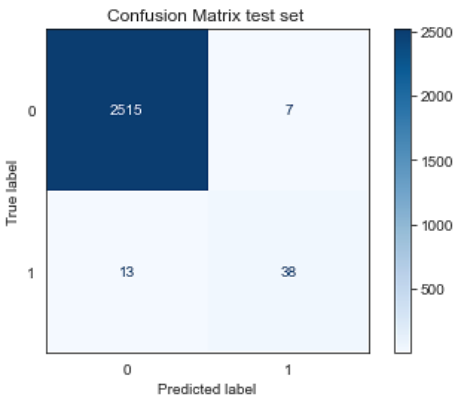
3.6 NEURAL NETWORK

Utilizzando la libreria sklearn.neural_network ed importando l'MLP Classifier (Multi-layer Percepton), che è un algoritmo appartenente alla famiglia del supervised learning, abbiamo cercato di ottenere dei risultati migliori rispetto a quelli ottenuti con keras.
Prima di partire con questo classificatore abbiamo normalizzato i dati attraverso un Robust Scaler.

Abbiamo cominciato la classificazione impostando valori random (solo per orientarci su quali parametri concentrarci per la Grid Search) e basandoci su ciò che si è visto a lezione: sapevamo appunto che tra tutte le funzioni di attivazione la 'relu' è quella più usata; abbiamo poi deciso di mettere 3 hidden layer composti rispettivamente da 6/12/24 'neuroni'; abbiamo in seguito impostato i parametri di alpha e momentum a 0,1 e 0,25 (visto che è sempre consigliato cominciare con dei valori bassi per poi cambiarli in base ai risultati ottenuti); abbiamo poi impostato il learning rate (il tasso di apprendimento dei pesi) = 'adaptive' visto che quest'ultimo mantiene il tasso costante a quello iniziale fin quando la training loss continua a scendere; infine abbiamo impostato il parametro 'early_stopping' = true in quanto volevamo evitare di andare in overfitting.

I risultati ottenuti sul *test set* con questa prima ricerca random sono:

Parametri	
hidden_layer_sizes	6, 12, 24
alpha	0,1
Learning rate	'adaptive'
Activation Function	'relu'
Early stopping	True
Momentum	0,25
Solver	'Adam'

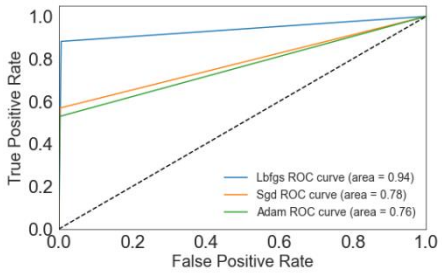


	Precision (%)	*Recall (%)*	*f1-score (%)*
Other genres	99	99	100
Old Time	84	75	79
Average scores	91	87	89
Accuracy (%)	***99***		

Abbiamo utilizzato una relu adaptive function e diminuito il numero di Hidden layers portandolo ad un unico hidden layer che però può avere un range di neuroni che va da 36 a 68. Abbiamo poi impostato il valore di alpha (che è il penalization term) pari a 1 (scelta guidata dagli outlier presenti nel dataset) così che se una particolare combinazione di features contribuisce in maniera sproporzionata alla classificazione allora sarà penalizzata molto nell'epochs successiva.

Il nostro lavoro si è concentrato nel cambiare di volta in volta il solver utilizzato per la classificazione.

SOLVER	*LBFGS*	*SGD*	*ADAM*
hidden layer	*43*	*40*	*45*
Accuracy	0.99	0.99	0.99
Precision	0.86	0.91	0.92
Recall	0.85	0.78	0.76
F1	0.89	0.83	0.82



A seguito di questi risultati abbiamo deciso di utilizzare come solver LBFGS per tutte le analisi seguenti, in quanto quest'ultimo è il migliore nell'individuare la nostra target variable.

Abbiamo poi provato a cambiare l'Activation Function: tenendo conto dei risultati ottenuti abbiamo deciso di continuare la nostra analisi con l'activation function 'Relu' in quanto in ogni caso si è sempre dimostrata in grado di predire meglio la nostra target variable.

Arrivati a questo punto abbiamo quindi cominciato a cercare i parametri migliori utilizzando diverse Grid Search k-fold cross validation.
Visto che in generale la funzione 'relu' è quella che ci ha dato i risultati migliori, continueremo ad utilizzarla come Activation Function. Utilizzeremo 'lbfgs' come solver per lo stesso motivo visto per l'Activation Function. Inoltre, utilizzeremo sempre 3 Hidden layer, andremo a cambiare di volta in volta il numero di neuroni all'interno senza però andare oltre ai 40 per layer.
I vari parametri usati nelle svariate Grid Search che abbiamo fatto sono:

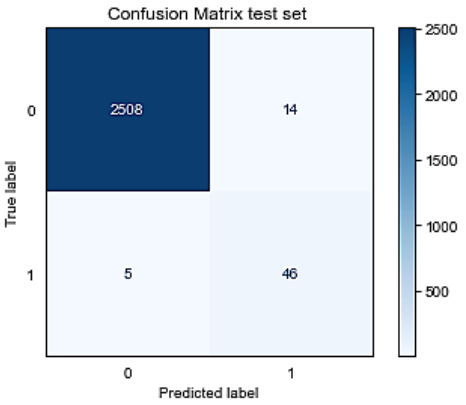
Grid Search + k-fold cross validation (Cv=3)	
hidden_layer_sizes	(10,30,10) / (20,30,40) / (6,12,24) / (24,12,6) / (12,24,6) / (6,24,12)
Activation Function	'relu'
Solver	'lbfgs'
Learning rate	'adaptive'
Momentum	0,1/ 0,2/ 0,5/ 0,7/ 0,8 / 0,9 / 1
Early_stopping	True/False
Alpha	0,1/ 0,2 / 0,5 / 0,7 / 0,8/ 0,9/ 1

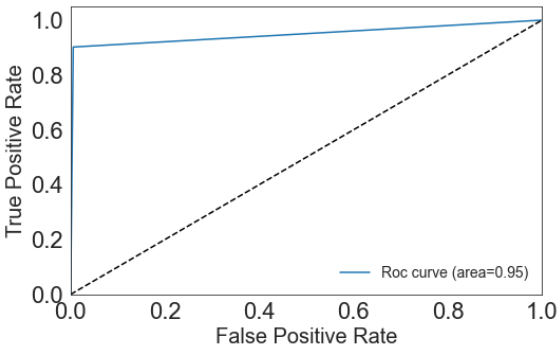
In un primo momento per poter scegliere il modello migliore abbiamo utilizzato i risultati ottenuti con il Validation set, tenendo il risultato che in generale riuscisse a predire bene la nostra target variable.

PARAMETERS	
hidden_layer_sizes	12,24,6
alpha	0,8
Learning rate	'adaptive'
Activation Function	'relu'
Early stopping	True
Momentum	0,8
Solver	'lbfgs'

VALIDATION			
	Precision (%)	*Recall (%)*	*f1-score (%)*
Other genres	99	99	100
Old Time	71	72	71
Average scores	85	86	85
Accuracy (%)	99		

TEST			
	Precision (%)	*Recall (%)*	*f1-score (%)*
Other genres	100	99	100
Old Time	77	90	83
Average scores	88	95	91
Accuracy (%)	99		





3.7. IMBALANCING TECHNIQUES

Come abbiamo già visto nella sezione 2, il nostro dataset è altamente sbilanciato e ciò influisce sulla classificazione.

Quindi, sono state utilizzate le imbalance tecniques per affrontare in questa problematica.

In particolare, abbiamo adoperato il Condense Nearest Neighbours e il Random Oversampling, metodi che ci hanno dato i migliori risultati col Decision Tree e KNN.

Tra tutti i classificatori utilizzati in questo progetto, la rete neurale sembrerebbe la più efficace nel predire correttamente, anche senza l'utilizzo delle tecniche di under e over sampling.

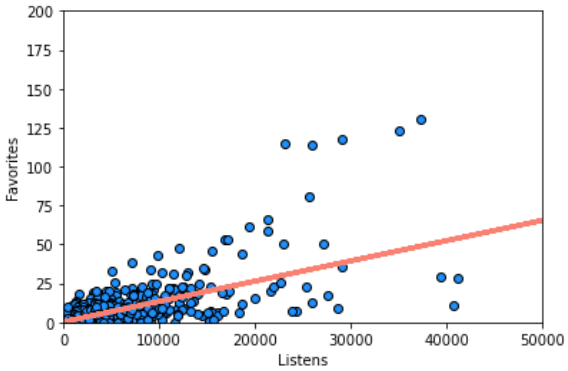
	Accuracy	Precision		AVG	Recall		AVG	F1-Score		AVG
		0	1		0	1		0	1	
Decision Tree	0.97	1.00	0.35	0.67	0.97	0.84	0.91	0.98	0.49	0.74
Decision Tree + CNN	0.92	1.00	0.18	0.59	0.92	0.90	0.91	0.95	0.30	0.63
Decision Tree +OverSampling	0.97	1.00	0.36	0.68	0.97	0.84	0.91	0.98	0.50	0.74
KNN	0.99	1.00	0.80	0.90	1.00	0.80	0.90	1.00	0.80	0.90
KNN + CNN	0.98	1.00	0.58	0.79	0.99	0.82	0.91	0.99	0.68	0.84
KNN + OverSampling	0.98	1.00	0.35	0.68	0.97	0.90	0.93	0.98	0.51	0.75
Naive Bayes	0.95	1.00	0.24	0.62	0.95	0.78	0.87	0.97	0.37	0.67
Naive Bayes + CNN	0.96	0.99	0.31	0.65	0.97	0.75	0.86	0.98	0.44	0.71
Naive Bayes + OverSampling	0.91	1.00	0.16	0.58	0.91	0.84	0.88	0.95	0.27	0.61
Logistic Regression	0.98	0.99	0.63	0.81	1.00	0.37	0.68	0.99	0.47	0.73
Logistic Regression + CNN	0.98	0.99	0.64	0.81	1.00	0.41	0.70	0.99	0.50	0.75
Logistic Regression + OverSampling	0.94	0.99	0.21	0.60	0.94	0.73	0.83	0.97	0.32	0.64
Rule Based	0.99	0.99	0.63	0.81	0.99	0.71	0.85	0.99	0.67	0.83
Rule Based + CNN	0.98	0.99	0.44	0.72	0.99	0.57	0.78	0.99	0.50	0.74
Rule Based + OverSampling	0.97	0.99	0.40	0.69	0.98	0.53	0.76	0.99	0.45	0.72
SVM	0.99	0.99	0.67	0.83	0.99	0.75	0.87	0.99	0.70	0.85
SVM + CNN	0.99	0.99	0.61	0.80	0.99	0.73	0.86	0.99	0.66	0.83
SVM + OverSampling	0.88	1.00	0.14	0.57	0.88	0.94	0.91	0.94	0.24	0.59
Random Forest	0.91	1.00	0.16	0.58	0.91	0.84	0.88	0.95	0.27	0.61
Random Forest + CNN	0.97	1.00	0.35	0.67	0.97	0.78	0.88	0.98	0.49	0.74
Random Forest + OverSampling	0.90	1.00	0.14	0.57	0.90	0.84	0.87	0.94	0.25	0.60
Neural Network	0.99	1.00	0.76	0.88	0.99	0.88	0.94	1.00	0.82	0.91
Neural Network + CNN	0.98	1.00	0.55	0.77	0.98	0.92	0.95	0.99	0.69	0.84
Neural Network + OverSampling	0.97	1.00	0.40	0.70	0.97	0.98	0.98	0.98	0.56	0.77

Le conclusioni derivanti dal nostro lavoro sono molteplici: possiamo di certo affermare che lavorare su un dataset sbilanciato ha di gran lunga influito sui risultati ma consci del fatto che comunque con alcuni

classificatori l'avremmo spuntata, cosa che siamo riusciti a fare con le reti neurali. In generale possiamo dire di non essere mai andati in overfitting in quanto le decisioni prese per il test set derivano da risultati sul train e sul validation set accettabili. Possiamo di certo affermare che certi classificatori si sono comportati in maniera coerente con ciò che abbiamo visto a lezione.

3.8 REGRESSOR PROBLEM

Quando si parla di retta di regressione, a meno che non si usi una logit function, non possono essere trattati i problemi di classificazione binaria.
Per questo motivo abbiamo deciso di ideare una nuova task specifica per questi tipi di problemi.
Abbiamo attribuito come variabile indipendente X → favorites e variabile dipendente Y → listens; dato il numero di ascolti, vogliamo sapere quale sia la probabilità che quella canzone con quei determinati ascolti vada tra i favoriti.
Abbiamo ottenuto i seguenti risultati:



	SIMPLE REGRESSION
COEFFICIENTS	0.00130032
INTERCEPT	0.4261
R2	0.707
MSE	267.181
MAE	2.812

Successivamente, abbiamo creato un multiple linear regression problem selezionando, oltre a listens, altre explanatory variables: comments, interest e roll_kurtosis.
Il nostro obiettivo è: dato il numero di commenti, il numero di interesse e il valore kurtosis del spectral rolloff (valore più rappresentativo nella logistic regression), quale è la probabilità che una canzone sia tra i favoriti?
Per trovare il miglior fit dei data points abbiamo utilizzato oltre al multiple linear regression, anche la regolarizzazione Tikhnov (ridge regression) e la Lasso Regression, ottenendo i seguenti risultati:

	MULTIPLE LINEAR REGRESSION		
	MULTIPLE	*LASSO*	*RIDGE*
COEFFICIENTS	1.52387688	0.	1.52160555
	5.22625396	0.00052166	5.22625587
	6.76602804	0.00069494	6.76629888
	1.76743535	0.00081827	1.76740849
INTERCEPT	0.0974	0.1140	0.0974
R2	0.779	0.806	0.779
MSE	201.062	177.232	201.027
MAE	2.685	2.667	2.685

Il Multiple regressor e il Ridge Regressor hanno fornito gli stessi risultati, sia in termini di coefficienti e intercetta, che di evaluation.
La Lasso regression ha ottenuto un migliore risultato con un R2 uguale a 0.806. Questo vuol dire che 80,6% dei valori fitta il modello. Ciò tenuto conto anche dal fatto che ha ottenuto un Mean Squared Error e n Mean Absolute Error minori.

4. TIME SERIES

L'obiettivo di questo progetto è stato fin dall'inizio eliminare le canzoni di vecchio stampo, ovvero quelle appartenenti al genere "Old-Time/Historic".
Dopo aver utilizzato vari classificatori per distinguere questo genere dagli altri, ci siamo poi domandati quanto queste effettivamente erano gradite da chi utilizza questa raccolta musicale. Inaspettatamente è risultato che Old Time Historic risulti essere tra i generi più ascoltati, più favoriti e quello a cui gli utenti sono maggiormente interessati (5° tra tutti i top genres per favorites, listens e 6° per interest).
Abbiamo poi notato che nella top3 rientri Classical che, come Old Time, risulta essere un genere più datato rispetto ad altri.
Quindi abbiamo deciso di dedicare questa parte di progetto nell'analizzare questo paradosso e comprendere quali sono i tratti caratteristici per cui canzoni di vecchio stampo risultano ancora oggi abbastanza ascoltate.
Per fare ciò, abbiamo creato un dataset ad hoc che riesca a catturare gli aspetti essenziali che caratterizzano i due generi e che ci permetta di comprendere perché questi due generi vadano ancora di moda.

4.1 CREAZIONE DATASET

Nella creazione del dataset per l'analisi delle time series tra i due generi scelti, innanzitutto abbiamo scaricato la libreria mp3 medium dal link [GitHub - mdeff/fma: FMA: A Dataset For Music Analysis]
Avremmo voluto occuparci del dataset small, ma in questo, essendoci solamente 8000 tracce, non include Old time geners.

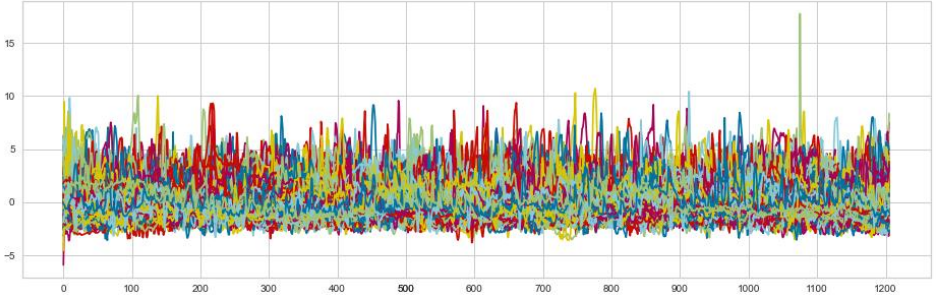
Successivamente, tramite la libreria librosa, abbiamo deciso di estrarre le waves per ogni canzone tramite la feature "spectral centoid" la quale, fin dall'inizio del nostro progetto, si è rivelata abbastanza buona per discriminare il genere Old-time / Historic dagli altri.
Quindi abbiamo estratto 28 secondi di ogni track utilizzando un sampling rate di 22050 e un hop_length di 512, ottenendo 1206 colonne di time series.
(Per fare ciò, abbiamo utilizzato il file jupyter fornito da Gaetano riadattandolo per la nostra task)

Abbiamo poi rimosso dal nostro dataset tutti gli altri generi musicali, lasciando solamente quelli per cui abbiamo deciso di dedicare la nostra analisi.
Abbiamo quindi ottenuto un dataset formato da 1028 tracce, di cui 502 Old-Time e 526 Classical, ognuna delle quali è composta da 1206 time stamps.
In particolare, in questa sezione ci occuperemo di tre principali topic riguardanti le time series estratte: Clustering, Motif e Discords discovery e Classification.

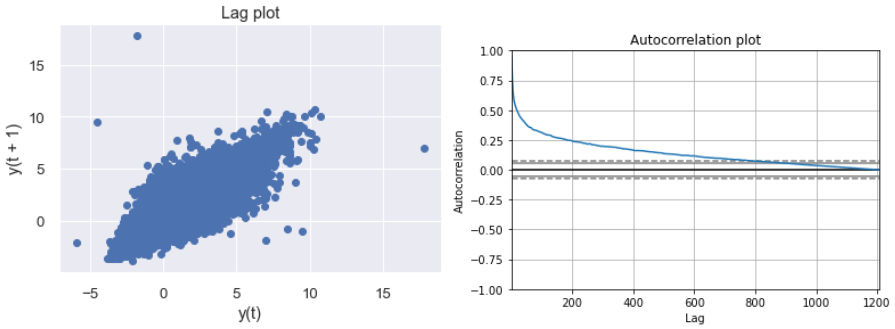
4.2. CLUSTERING

4.2.1 Data Preparation

Prima di procedere con l'analisi, abbiamo deciso di normalizzare i dati con l'offset translation e l'amplitude scaling, in modo da ottenere risultati migliori nel clustering.



Dalla analisi della Auto-correlazione è possibile notare che il dataset non ha valori generati in maniera random. Mentre l'Auto-correlazione plot mostra che ci sono valori positivi di auto-correlazione.

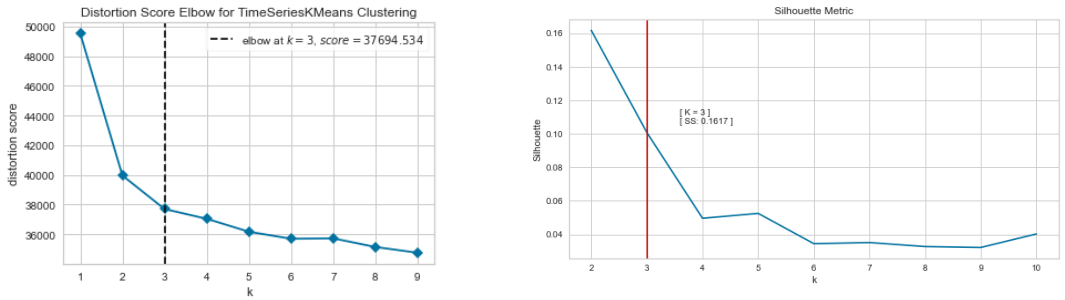


4.2.2. Clustering KMeans

L'algoritmo K-Means è stato eseguito utilizzando sia l'Euclidean distance e la Dynamic Time Warping come proximity metrics. Per raggiungere la convergenza, abbiamo inizializzato i nostri centroidi runnando il K-Means 10 volte con un numero massimo di iterazioni per un singolo run impostato a 200. Il miglior output è stato poi selezionato dal modello.

Identificazione best k

Il miglior numero di k è stato selezionato usando l'Elbow Method, che ha calcolato un distorsion score (SSE) per k ∈ (2,10).
Il valore ottimale di SSE è stato ottenuto al k=3, che corrisponde a uno SSE score = 37694.534 e un Silhouette score=0.1617.



In base a questo, abbiamo applicato un K-Means algorithm settando un numero di clusters pari a 3.

Approximated Clustering

Avendo un dataset formato da 1206 time stamps, è stato impossibile calcolare la Dynamic Time Warping per l'elevato costo computazionale.
Per questo motivo abbiamo deciso di ridurre la dimensionalità attraverso le tecniche di approssimazione; in particolare utilizzeremo la *Piecewise Aggregation Approximation* e la *Simbolic Aggergate Approximation*.
Entrambe le metodologie richiedono come parametro l'utilizzo di un certo numero di segmenti, su cui poi verranno ridotte le Time Series.

Tenendo conto della potenza dei nostri computer e del livello computazionale, abbiamo fatto diversi tentativi per capire quale fosse il miglior numero di segmenti su cui suddividere le time series.
Abbiamo provato i seguenti segmenti: 60, 120 e 240.
Abbiamo riscontrato che con l'aumentare dei segmenti, i vari cluster riuscivano sempre più a suddividere i due generi musicali.
Il risultato migliore è stato infatti ottenuto utilizzando un segmento pari a 240.

Mentre per la Simbolic Aggregation Approximation, oltre al numero di segmenti, richiede come parametro anche il numero di lettere da usare. Dopo svariate prove, abbiamo deciso di utilizzare un numero di lettere pari a 8, le quali ci hanno dato un migliore risultato.

Descrizione del miglior clustering

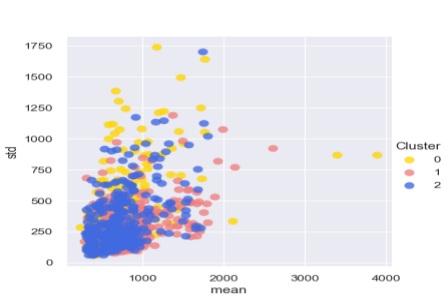
Il migliore K-Means algorithm è stato quello ottenuto tramite la Simbolic Aggergate ApproXimation utilizzando la Dynamic Time Warping distance.
I data points sono distribuiti nei tre clusters individuati nel seguente modo:

- **CLUSTER 0**: formato da 217 Classical tracks e 39 Old-Time / Historic.
  Risulta essere il meno favorito e il meno ascoltato fra gli altri 2 clusters, con un numero inferiore di durata del brano (150) e un bit rate (170000) e con un massimo di 3 commenti.
  Valori statistici sulla spectral centroid: valore medio pari a 2000 e std pari a 500.

- **CLUSTER 1**: formato da 71 Classical tracks e 351 Old-Time / Historic.
  È il più favorito e il più ascoltato, con una maggiore durata (600) e numero di bit rate (superiori a 300000) e ha ricevuto massimo 5 commenti.
  Valori statistici sulla spectral centroid: media uguale a 1500 e std 4000.

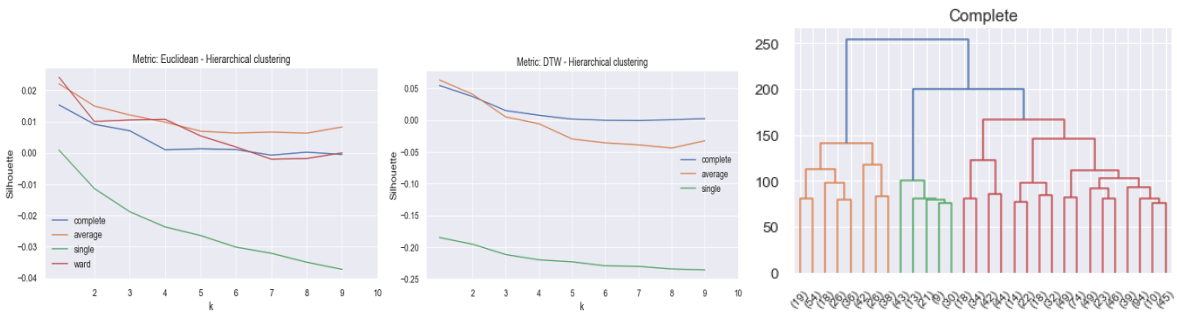
- **CLUSTER 2:** formato da Classical tracks 238 e 112 Old-Time / Historic.
  Si discosta per poco dal cluster 1 per numero di ascolti e favoriti, ma è quello che ha ricevuto massimo 9 commenti per canzone.
  Ha un media una durata di 270 e bit rate in media che raggiungono fino a 250000.
  Valori statistici sulla spectral centroid: media uguale a 1000 e std 3000.

Successivamente, abbiamo deciso di rappresentare graficamente i risultati ottenuti dei tre clusters tramite uno scatter plot.

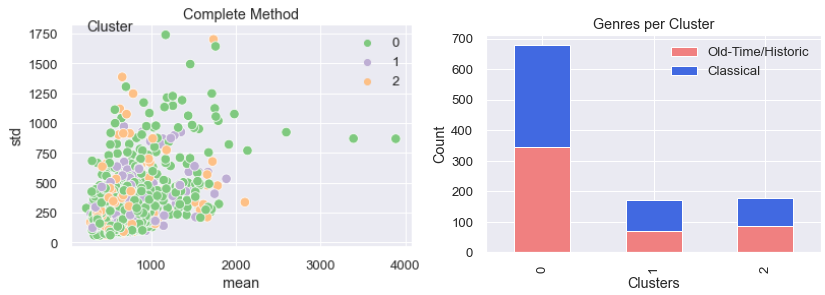


4.2.3. Hierarchical Clustering

Non contenti del risultato, ci siamo cimentati nel provare a trovare risultati migliori attraverso l'algoritmo dello Hierarchical clustering.
Le stesse variabiabili utilizzate per il K-Means sono state impiegate per questo clustering. L'algoritmo è stato testato con l'Euclidean e DTW (precomputed) proximity metrics. Nella nostra analisi abbiamo applicato i seguenti algoritmi: *Single Link*, *Complete Link*, *Average Link* e *Ward's Method*.
Per ognuno di loro, abbiamo provato a rappresentare i clustering con un numero di clusters in range (2,10) e calcolato la silhouette score per ognuno di loro.
I risultati sono stati plottati nella line-graph che mostra come la silhouette score (calcolata usando i diversi algoritmi dello hierarchical) varia per i diversi valori di k.



Tenendo conto dei risultati della silhouette, abbiamo deciso di provare a trovare tre clusters sperimentando con entrambe le due Proximity metrics.
L'unico risultato migliore è stato ottenuto dal Complete Link con la Dynamic Time Warping distance.
Abbiamo osservato come il cluster 0 contenga canzoni della maggior parte di entrambi i generi e risulti essere il più favorito e ascoltato degli altri 2 clusters.



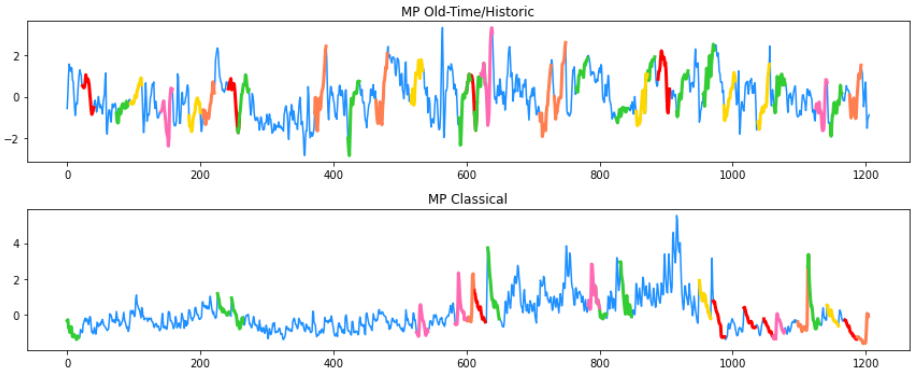
Inizialmente ci aspettavamo risultati migliori, sia a livello di visualizzazione dei clusters e soprattutto che i clusters dividessero meglio i due generi considerati, ma possiamo ritenerci soddisfatti del risultato ottenuto.
Abbiamo ipotizzato che ciò potrebbe essere dipeso da due motivi:
- non aver utilizzato l'intera Time Series ma una sua approssimazione (per i motivi già spiegati sopra);
- la features spectral centroid potrebbe essere non adatta per questa tecnica unsupervised.

4.3. Motifs, Anomalies e Shapelet Discovery

Obiettivo fondamentale di questa sezione è stato fin da subito capire quali fossero le caratteristiche che avessero in comune questi due generi, caratteristiche che li portando ad essere tra i più ascoltati.
Questo aspetto può essere scoperto tramite la Motif, Shapelet e Anomaly Discovery.

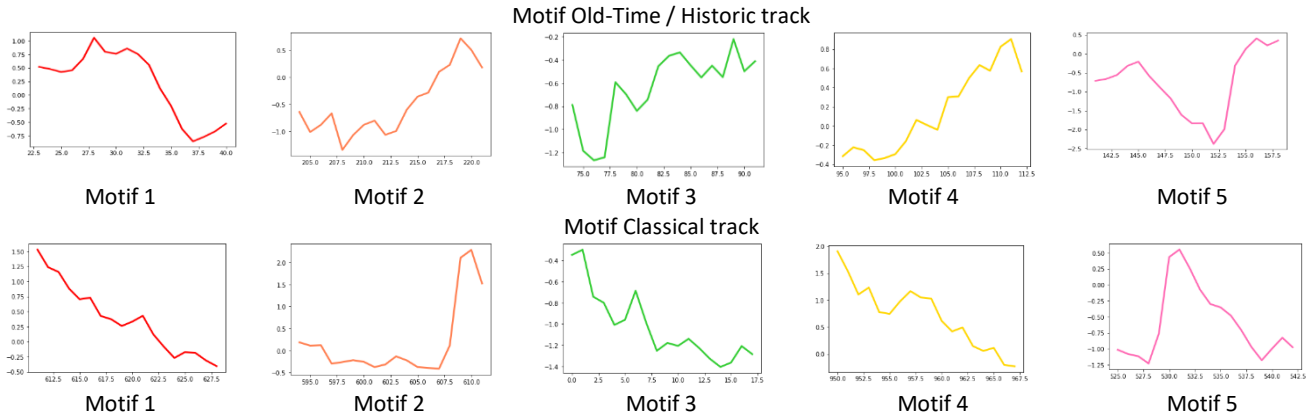
4.3.1. Motif Discovery

La Motif discovery è una ricerca che si occupa di approssimare una TS per poter così trovare dei pezzi più piccoli della TS analizzata che si somiglino. Abbiamo deciso di fare questa analisi selezionando una canzone random per i due generi. Tale decisione è stata presa dal fatto che abbiamo constatato che diverse tracce tra loro dello stesso genere, presentassero la stessa Matrix Profile, quindi selezionare una canone di un genere rispetto ad un'altra non avrebbe influito nella motif discovery.
Dopo avere selezionate le tracce su cui effettuare l'analisi, abbiamo estratto le loro matrix profile tramite il stamp optimization algortithm.
Questo richiede come parametro un certo numero di windows, su cui poi andrà ad applicarsi alle 2 time series considerate.
Dopo svariati tentativi, abbiamo deciso di utilizzare un window di 18, in modo tale da non cambiare di troppo la forma delle due time series considerate.
.
Abbiamo scelto di individuare le 5 top motif:



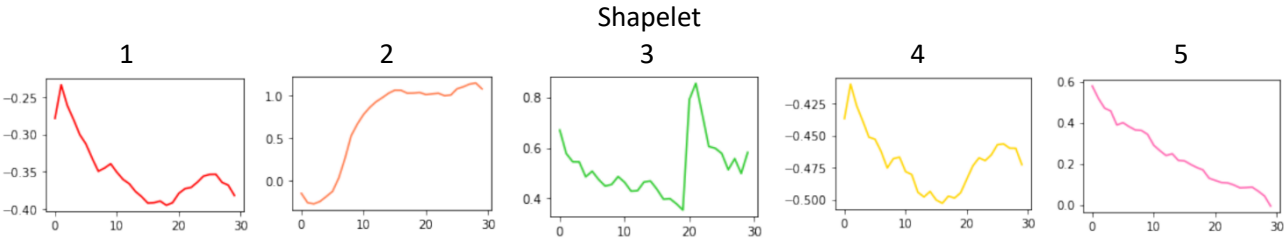
MPI	*Old-Time*	*Classical*
1	0,8391	0,6670
2	0,8825	0,7070
3	1,0153	0,8074
4	1,0750	0,8752
5	1,0904	1,0291

Possiamo notare che il genere Old-Time è formato prevalentemente da stessi suoni che si ripetono più volte in tutta la durata della canzone, mentre per la music classica questo avviene molto meno.
Sia dal grafico che dal MPI, le varie motif tra solo sono molto vicine, pure se i generi sono diversi.
Interessante è stato considerare la forma.
In particolare, le motif numero 2 si assomigliano parecchio tra loro, mentre la motif 1 old time ha una shape più simile alla motif classical numero 5.



4.3.2. Shapelet Discovery

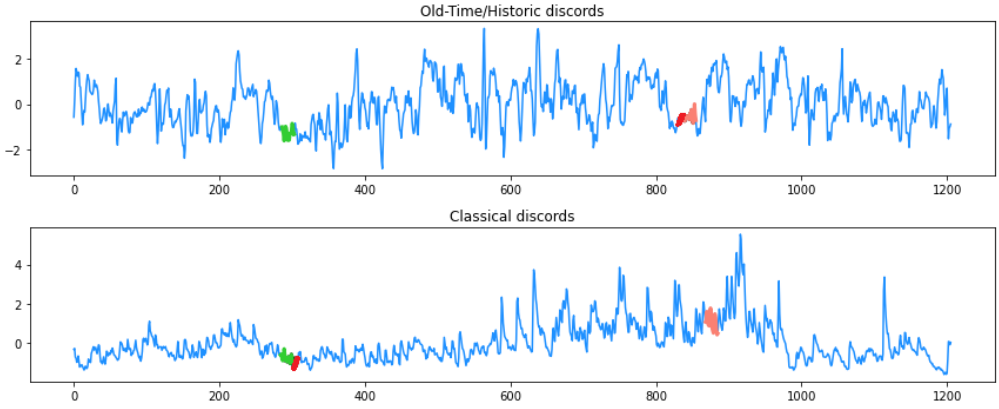
Una shapelet è un pattern che è massimamente rappresentativo di una certa classe rispetto a un dataset.
La differenza principale tra Motif e Shapelet è data dal fatto che la prima si occupa di trovare dei pezzi di TS ricorrenti all'interno di una singola traccia, mentre nella Shapelet il concetto è lo stesso ma applicato a tutto il dataset riguardante le TS in modo da operare delle classificazioni.
Abbiamo estratto 5 top shapelet utilizzando Grabocka implementation e ne abbiamo plottato i risultati:



Possiamo notare alcune somiglianze nella shape tra motif e shapelet estratte.
In particolare, la shapelet 4 con la motif 4 del Classical risulta simile: dobbiamo però dire che mentre la shapelet utilizza una size di 30, la motif risulta molto più piccola; per questo motivo sembrerebbe che la motif 4 sia rappresentata nella prima parte della shapelet 4 individuata.

4.3.3. Discords Discovery

Utilizzando la Matrix Profile calcolta per i motif, abbaimo continuato il nostro lavoro alla ricerca di anomalie all'interno di entrambi i generi.
Come possiamo vedere dai grafici, le anomalie individuate sono localizzate nelle stesse zone per entrambi i generi con la sola eccezione dell'anomalia evidenziata in rosso; quest'ultima si trova vicino a quella verde nel genera classical (posizione 291) mentre per l'Old-Time la troviamo poco prima dell'anomalia arancione(posizione 835).
Le anomalie riscontrate hanno però una shape simile tra di loro senza picchi particolari distinguibili all'interno della time series.



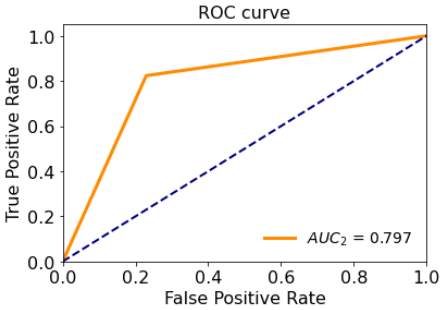
4.4. Shapelet Classification

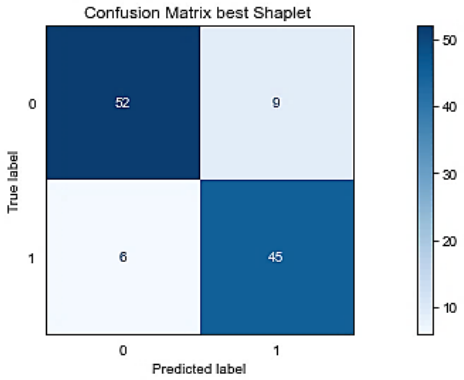
Sapendo che la Time Series Classification ha come obiettivo una funzione f che riesca, dallo spazio delle possibili Time Series, a trovare uno spazio di classificazione possibili e che ha bisogno di parametri imposti dall'utente, ci siamo occupati di far caricare diversi parametri all'interno del nostro file: abbiamo giocato sul numero di shapelet (in quanto il nostro dataset ci permetteva di trovarne più di una) e anche sul parametro l (la lunghezza). Dopo aver provato diverse configurazioni e aver confrontato i risultati abbiamo optato per la Shapelet Discovery utilizzando la libreria Grabocka e settando una lunghezza pari a 30 e mostrando all'interno del modello le top 5 Shapelets.

Abbiamo quindi provato i seguenti parametri andando a combinarli di volta in volta per trovare la classificazione con i risultati migliori:

Parameters	
Optimizer	***'sgd' 'Adam' 'RMSprop' 'Adamax'***
Weight regularizer	0,01 / 0,02 /0,05/ 0,1 / 0,2 /0,5 / 0,7 / 0,8 / 1
Max iteration	150 / 200

Il risultato migliore a seguito di diverse prove è stato utilizzando un RMSprop optimizer, con un weight regularization pari a 0.05 e un numero di massime iterazioni pari a 200.



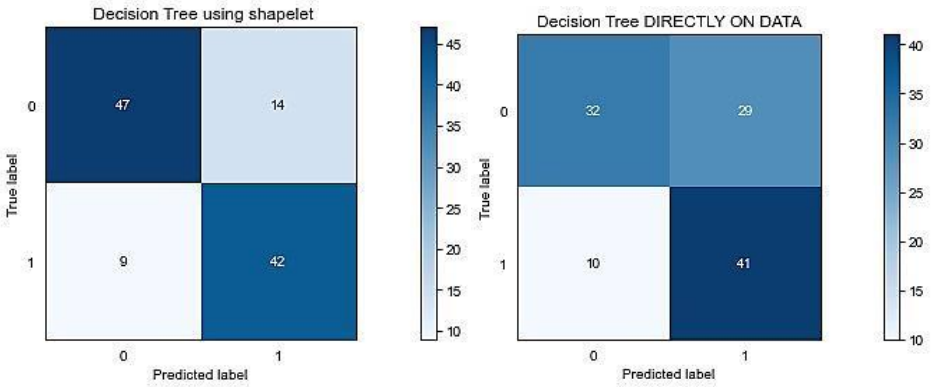


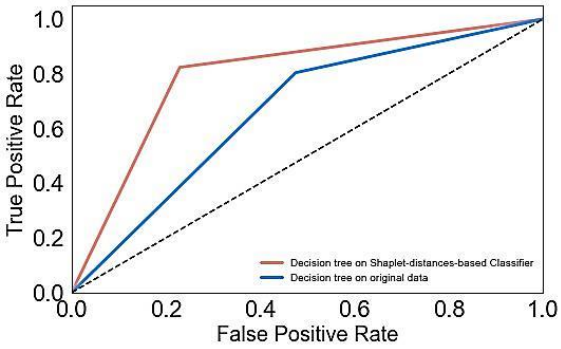
	Precision (%)	*Recall (%)*	*f1-score (%)*
Classical	90	89	89
Old Time	87	88	87
Average scores	88	88	88
Accuracy (%)	88		

Successivamente abbiamo utilizzato questa classificazione per trasformare i nostri dati: adesso il nostro train set è composto da una matrix con 807 righe e 5 colonne (in linea con ciò che abbiamo visto prima); abbiamo apportato la stessa trasformazione anche al test set.
Purtroppo, in questo nuovo dataset non siamo riusciti ad utilizzare la Dynamic Time Warpying distance; quindi utilizzeremo come metrica la distanza euclidea.
Tenendo conto di questo nuovo dataset e di quello originale creato da noi, abbiamo quindi continuato il nostro lavoro occupandoci del Decision Tree e del KNN.
In entrambi i classificatori abbiamo usato delle grid search per settare i parametri.

4.4.1. DECISION TREE

In questo caso ci siamo occupati di una classificazione semplice, evidenziando la differenza tra il classificatore usando la TS originale e quello trasformato tramite la shapelet.
Nel primo caso (Decision Tree sulla TS originale- grafico a sinistra) abbiamo ottenuto una recall media del 66% e una precision media del 67%.
Nel secondo caso (Decision Tree sulla TS dopo la shapelet) abbiamo ottenuto una recall media del 80% e una precision media del 79%.
In generale quindi l'utilizzo della shapelet ha aiutato predire risultati migliori con questo classificatore.

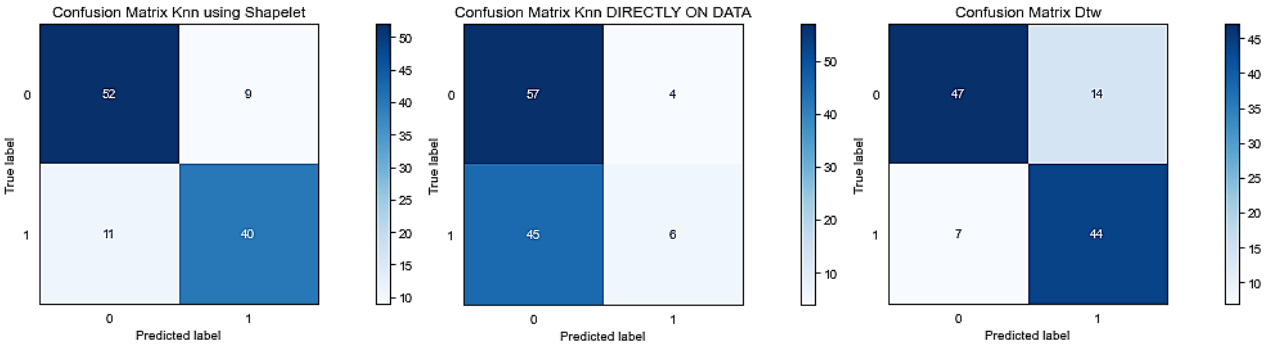




Score	*Decision Tree*	*Decison tree Shapelet*
Accuracy	0.65	0.80
Precision	0.67	0.80
Recall	0.66	0.79
F1	0.65	0.79
ROC	0.66	0.88

4.4.2. K-nearest neighbors

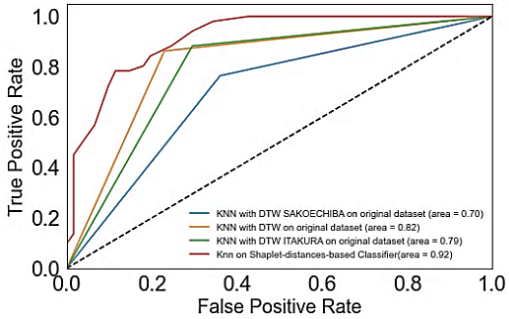
Qui invece oltre ad operare la classificazione con la shapelet abbiamo deciso di utilizzare il dataset originale in modo da confrontarlo con quello con la DTW. Per poter comprendere a pieno i vantaggi di questi due metodi abbiamo deciso di confrontare le rispettive confusion matrix.
I risultati derivanti dal lavoro svolto dalla DTW e dalla Shapelet sono evidenti dalle confusion matrix riportate sotto:



Nella confusion matrix di sinistra utilizziamo il KNN sul dataset trasformato con la shapelet. Le altre due rappresentano i risultati dei classificatori in cui utilizziamo il KNN nel dataset originale con differenti metric parameters: euclidean(centrale) e dtw(destra).

Per velocizzare il calcolo computazionale del DTW distance, abbiamo successivamente utilizzato due constraints: la Sakoe-Chiba Band e l'Itakura Parallelogram.
Di seguito le rispettive Roc curve (confrontate anche con la shapelet).

	KNN ON ORIGINAL DATASET			KNN
Score	*DTW*	*DTW Sakoe*	*DTW Itakura*	SHAPELET
Accuracy	0.81	0.70	0.79	0.82
Precision	0.82	0.70	0.80	0.82
Recall	0.81	0.70	0.79	0.82
F1	0.81	0.70	0.79	0.82



In questo caso, visto che nella nostra task per questa seconda parte ci siamo focalizzati maggiormente sul riconoscimento di 2 generi (e non un genere contro tutti).
Il classificatore col risultato migliore (come si evince anche dalle Roc curve) è quello ottenuto usando la TS dopo la shapelet.

CONCLUSIONI

Tenendo conto di ciò che abbiamo individuato utilizzando le motif e le shapelets e tenendo in considerazione l'obiettivo di questa parte di analisi possiamo affermare che fra i due tipi di canzoni c'è qualche somiglianza che potrebbe giustificare il fatto che entrambi vanno di moda.
Utilizzando la shapelet siamo riusciti a terminare due diverse task: la prima consiste nel mappare all'interno delle Time Series dei due generi analizzati delle parti della Time Series in grado di aiutarci nella classificazione, nella seconda task ci siamo focalizzati nelle differenze e uguaglianze tra Shapelets e Motifs.
In generale possiamo dire che grazie all'utilizzo di Shapelet e DTW siamo riusciti ad ottenere dei risultati molto buoni; questo secondo noi è dovuto anche dal fatto che le due classi erano perfettamente bilanciate tra loro.
L'unica nota negativa, come già emerso nel report, è il costo computazionale dell'utilizzo delle DTW: le difficoltà maggiori che abbiamo affrontato riguardano le tempistiche per un dataset che è comunque ristretto.

5. SEQUENTIAL PATTERNS MINING

Lo scopo di questa task è l'individuazione di frequent patterns contigui di lunghezza minima pari a 4.
Per l'obiettivo si è provveduto a discretizzare la time con l'algoritmo SAX.

5.1. Preprocessing

In questa fase abbiamo utilizzato lo stesso dataset impiegato per la Time Series Analysis (sezione 3.).
I dati sono stati normalizzati con Z-normalization e sono stati discretizzati usando il Sax algoritmo, che permette di trasformare una sequenza di numeri razionali in una sequenza di lettere.
La libreria utiizzata può essere trovata nel seguente link: https://github.com/seninp/saxpy.git.
Per questa analisi è stato scelto un numero di segmenti pari a 10 e un alfabeto di 7 valori (a, b, c, d, e, f, g); in questo modo il dataset è stato diviso in 7 bins, ognuno dei quali corrisponde a una lettera.

5.2. PrefixSpan Algorithm

Per scoprire i motifs all'interno del dataset, abbiamo utilizzato l'algoritmo PrefixSpan.
E' stato scelto di applicare una lenght pari a 4.
L'algoritmo è stato lanciato più volte con diversi valori di minsup e successivamente si è provveduto ad individuare le 5 sequenze più frequenti:

Minsup	Tot n°of Sequences
15	3087
30	1299
45	763
60	517
75	368

TOP 5 MOST FREQUET SEQUENCES	
Patterns	Support
'd','d','d','d'	394
'd','d','d','c'	352
'd','d','c','d'	341
'd','c','d','d'	325
'c','d','d','d'	323

Cambiando il valore del minimum support, nessuna sequenza di 5 o più è stata trovata. I patterns più frequenti consistono in una sottosequenza di lunghezza 4 con items che hanno un alto valore, identificati dalle lettere c(sup=965), d(sup=957).

6. Advanced Clustering e Transactional Clustering

Per questa parte del progetto abbiamo optato per un dataset composto dalle features utilizzati nelle classificazioni della parte 2, cioè Spectral Rolloff e Spectral Centroid, con l'eccezione che al posto di utilizzare tutte le canzoni, utilizzeremo soltanto i generi Classical e Old Time/Historic.

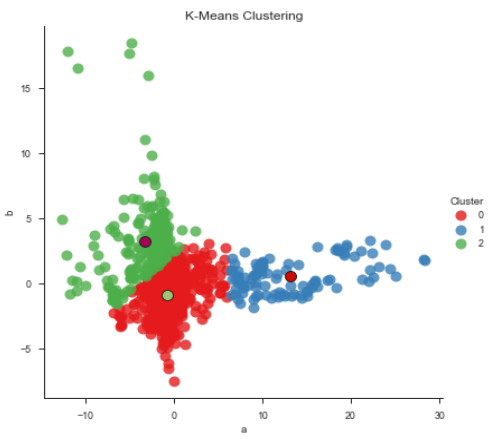
Dato che le due features hanno 7 valori ciascuno abbiamo deciso di operare in un primo momento un Robust Scaler e poi una PCA pari a 2 in modo da riuscire a plottare i risultati dei vari cluster che abbiamo utilizzato.

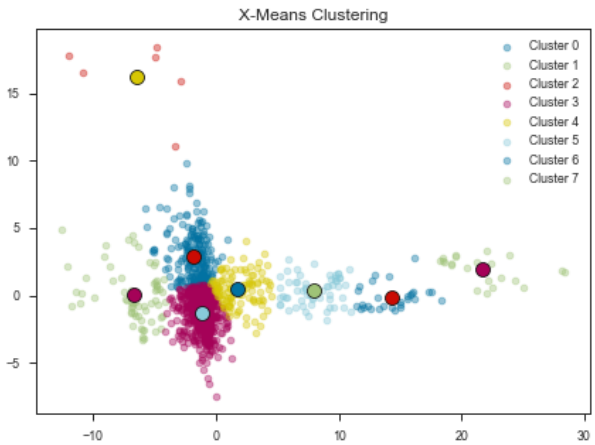
Abbiamo deciso di utilizzare in un primo momento il k-means per evidenziare come il parametro esterno K possa influenzare totalmente la clusterizzazione. Per l'Advanced clustering abbiamo utilizzato un X-means mentre per il Transactional Clustering abbiamo utilizzato il K-mode trasformando le variabili continue in categoriche tramite pandas, ottendo tre bins per ciascuna variabile categorica.

6.1. K-MEANS & X-MEANS

Per definire il valore di k abbiamo utilizzato l'elbow method, ottenendo un valore ottimale di k=3 e un silhouette score pari a 0.4740.

Mentre l'algoritmo X-Means si è fermato a 8 clusters





K-MEANS				
Clusters	**Classical**	**Old Time**	**TOT**	**Centroids**
0	423	393	816	-0.76 , 0.92
1	61	39	100	-0.76 , 0.92
2	135	78	213	13.18 , 0.57

X-MEANS				
Clusters	**Classical**	**Old Time**	**TOT**	**Centroids**
0	122	94	216	-1.85 , 2.87
1	57	7	64	-6.72 , 0.04
2	6	0	6	-6.47 , 16.24
3	305	314	619	-1.16 , -1.34
4	62	50	112	1.78 , 0.43
5	32	28	60	7.96 , 0.37
6	26	4	30	14.27 , -0.11
7	9	13	22	21.75 , 1.95

Utilizzando entrambi gli algoritmi, sembra che nessuno dei due sia riuscito ad individuare un cluster più per un genere, ma in contrario ci sia la prevalenza di un cluster che contenga più la metà di entrambi i generi; nel caso del K-means, il cluster 0 raggruppa circa il 72% di tutte le canzoni, mentre nel X-Means il cluster 3 include 305 classical tracks e 314 Old Time tracks.

6.2. K-Modes

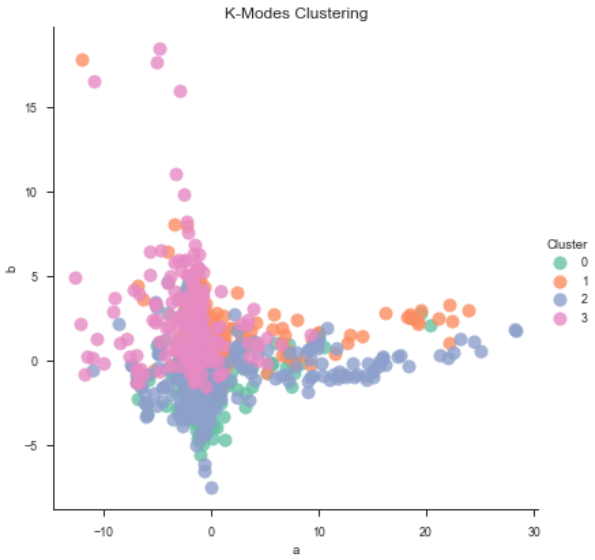
Una differenza sostanziale tra K-Means e K-Modes è data dal fatto che quest'ultimo utilizza la moda come centroide. Avendo solamente come variabile categorica "genre_top", abbiamo dovuto trasformare le variabili continue in categoriche, raggruppandole in 5 bins della stessa dimensione.

Successivamente, tramite l'elbow method abbiamo trovato un numero ottimale di k=4, e quindi abbiamo proceduto con l'algoritmo.

Sorprendentemente, con un cluster di tipo transactional siamo riusciti nel nostro intento: trovare quel cluster che riuscisse a distinguere meglio il genere Old Time/Historic. Infatti, il cluster 0 contiene 408 tracce di questo genere e solamente 17 delle canzoni classiche. Quindi possiamo assumere che questo rappresenti le caratteristiche del genere Old-Tme Historic. In particolare abbiamo riscontrato che il valore più discriminante tra il cluster 0 e gli altri sia il valore "min" di spectral_rolloff e di spectra_centroid che risulta leggermente più elevato, andando rispettivamente in un range di 398,2-796,4 per il primo e un range 254.8-509.6 per il secondo. Gli altri clusters arrivano a valori inferiori.

Mentre, guardando agli altri clusters, il cluster 2 si differenzia dagli altri per un valore di rollof_std maggiore (range 989-1872). Molto simili invece sembrano risultare i clusters 0 e 1 in quanto possiedono gli stessi valori statistici roll_std (range 106-989), centr skew (-2,5 – 4) e centr std (range 55-434).

Gli altri valori statistici sembrano essere uguali. Infatti, ciò avvalora la nostra tesi (e come già riscontrato precedentemente in altre analisi) che questi due generi siano abbastanza simili tra loro.



K-MODES			
Clusters	**Classical**	**Old Time**	**TOT**
0	17	408	425
1	132	24	156
2	199	34	233
3	217	44	261

7. EXPLAINABILITY

In questa ultima parte del progetto ci siamo occupati di utilizzare gli Algoritmi affrontati a lezione utili a rendere più comprensibili le scelte fatte dai classificatori.
Sull'onda del lavoro svolto nelle task precedenti abbiamo preparato il dataset: abbiamo deciso di utilizzare un Decision Tree sui due generi analizzati nelle time series, cioè Old Time/ Historic e Classical, ottenendo un dataset più o meno equilibrato: Classical 619 canzoni e Old-Time / Historic 510 canzoni.
Abbiamo poi diviso in training, validation e test set rispettivamente composto da 903, 113 e 113 canzoni.
Visto che abbiamo sempre usato le features per classificare abbiamo continuato per questa strada: come in precedenza abbiamo usato spectral rolloff e spectral centroid;

Dopo aver usato il decision tree, abbiamo usato questo classificatore all'interno della libreria Lime e abbiamo ottenuto i seguenti risultati.

7.1. LIME

In questo caso abbiamo deciso di confrontare i risultati di istanze predette correttamente che avessero un risultato inequivocabile.

Tramite l'Explenation method LIME siamo arrivati alle seguenti conclusioni:

- nel caso della classificazione del genere Classical le features centr_median, roll_min, roll_kurtosis, centr_min, roll_skew, roll_median, centr_skew, centr_mean contribuiscono per le relative percentuali a predire che il pezzo sia Old Time/Historic mentre roll_std e roll_max spingono verso la classificazione come Classical.
  Alla fine però i valori alti di roll_std e roll_max (nonostante sembri il contrario) portano a classificare il pezzo come Classical;
- nel caso della classificazione del genere Old Time/Historic le features roll_kurtosis, centr_median, roll_min, centr_min, centr_std, roll_skew, centr_skew contribuiscono per le relative percentuali a predire che il pezzo sia Old Time/Historic mentre roll_std, roll_max e centr_kurtosis max spingono verso la classificazione come Classical.
  Alla fine però i valori alti di roll_kurtosis, centr_median, roll_min, centr_min, centr_std, roll_skew, centr_skew portano a classificare il pezzo come Old time/Historic;

In entrambi i casi roll_std e roll_max sono le features che aiutano a spostare la classificazione verso Classical mentre centr_median, roll_min, roll_kurtosis, centr_min, roll_skew, centr_skew aiutano a spostare la classificazione verso Old Time/ Historic

CLASSICAL

Feature	Value	Importanza nella classificazione
centr_median	615.38	13,26%
roll_min	10.77	12,32%
roll_kurtosis	16.24	11,95%
roll_std	463.88	-8,22%
centr_min	166.75	1,97%
roll_skew	2.46	1,75%
roll_median	1162.79	1,50%
roll_max	8333.35	-1,00%
centr_skew	3.57	0,53%
centr_mean	620.82	0,17%

OLD TIME /HISTORIC

Feature	Value	Importanza nella classificazione
roll_kurtosis	735.81	12,89%
centr_median	664.45	11,91%
roll_min	689.06	10,65%
roll_std	158.45	-9,71%
roll_max	9087.01	-3,96%
centr_min	472.66	2,28%
centr_kurtosis	431.88	-1,35%
centr_std	91.80	0,97%
roll_skew	15.47	0,51%
centr_skew	10.50	0,29%