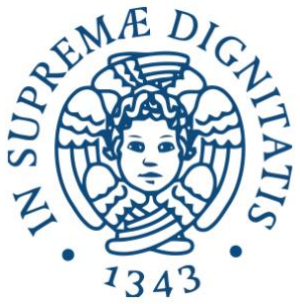

Università di Pisa

Dipartimento di Informatica

Corso di Laurea Magistrale in

Data Science and Business Informatics

Progetto Data Mining 2

Relazione sul progetto:

Music Analysis

A cura di:

Giuseppe Milazzo, 620190

Mario Bianchi, 616658

Alessandro Vavalà, 620168

Anno accademico 2020/2021

Indice

1 Introduction, Imbalanced Learning and Anomaly Detection

1.1 Data Understanding

Il Music Analysis Dataset è composto da 4 subset: "*tracks*" con 106574 righe e 52 colonne, "*genres*" con 163 righe e 4 colonne, "*features*" con 106574 righe e 518 colonne ed "*echonest*" con 13129 righe e 249 colonne. Da questi 4 subset è stato ricavato un dataset con 13129 osservazioni e 20 variabili. Di queste 20 variabili, 8 sono le "*audio*_features" di "*echonest*", ossia "*acousticness*", "*danceability*", "*energy*", "*instrumentalness*", "*liveness*", "*speechiness*", "*tempo*" e "*valence*"; 11 sono nuove variabili che sono state ricavate calcolando la media dei rispettivi momenti dell'indicatore "*mean*" per ogni variabile del subset "*features*". La variabile "*Electronic Labeler*" è stata ricavata dalla variabile "*genre*" ed è stata modificata per indicare solo il genere "*Eletronic*", genere scelto perché quello con maggior presenza nel *dataset*.

	acousticness	danceability	energy	instrumentalness	liveness	speechiness	tempo	valence	mfcc_mean
mean	0,526822797	0,489265585	0,538782377	0,640781442	0,184839458	0,095769751	122,7364935	0,441814672	-1,062557074
std	0,381784208	0,189996462	0,273901481	0,361279017	0,154819473	0,131795743	34,11180787	0,275017103	5,768691165
min	9,035E-07	0,051435142	2,01659E-05	0	0,025916382	0,022323668	26,131	0,00001	-26,67200813
max	0,99579645	0,968644662	0,999963711	0,997120945	0,980330001	0,966177407	249,616	0,99999	11,74127721

	chroma_cens_mean	chroma_cqt_mean	chroma_stft_mean	tonnetz_mean	spectral_contrast_mean	spectral_centroid_mean	spectral_bandwidth_mean	spectral_rolloff_mean	rmse_mean	zcr_mean
mean	0,24926843	0,498176946	0,45217421	0,003252153	20,26937079	1210,174491	1451,800058	2462,125241	4,411806915	0,052108355
std	0,016483316	0,065168139	0,081907788	0,014680045	1,428003092	414,4110822	390,4978429	950,6774044	2,212223431	0,021841504
min	0,154400609	0,243457822	0,151049563	-0,085529422	12,33243043	214,3581085	343,5090027	324,3878174	0,039436772	0,005178985
max	0,285262855	0,741233389	0,790975849	0,070725668	27,88653633	5259,629395	3309,828857	9201,961914	23,66340447	0,454479694

Tabella 1.1

Tutte le variabili sono di tipo numerico tranne la variabile "*Electronic Labeler*" che ha valori {1, 0}.

Nella seguente tabella è possibile osservare le variabili e il loro significato.

Variabile	significato
acoustiness	misura quanto una traccia sia acustica
danceability	misura quanto è adatta una traccia per ballare
energy	misura la percettività di intensità di intensità e attività della traccia
instrumentalness	misura quanto una traccia sia strumentale
liveness	valore che descrive la probabilità che la traccia sia stata registrata dal vivo
speachness	misura il livello di parole presenti nella traccia
tempo	misura la velocità del *beat* sottostante
valence	misura la positività musicale di trasmessa da una traccia
mfcc_mean	misura la media dello spettro di potenza a breve termine di un suono
chroma_cens_mean	media di più finestre temporali di breve durata del contenuto armonico dell'audio estratto dallo spettro di magnitudine utilizzando il Chroma Energy Normalized (CENS)
chroma_cqt_mean	media di più finestre temporali di breve durata del contenuto armonico dell'audio estratto dallo spettro di magnitudine utilizzando la trasformazioe CQT
chroma_stft_mean	media di più finestre temporali di breve durata del contenuto armonico dell'audio estratto dallo spettro di magnitudine utilizzando la trasformazioe STFT
tonnetz_mean	media di misura dello spazio tonale
spectral_contrast_mean	misura della media della differenza di livello tra picchi e valli dello spettro
spectral_centroid_mean	media in cui l'energia di uno spetrro è centrata nella traccia
spectral_bandwidth_mean	misura la media dell'estensione della funzione di trasferimento di potenza intorno alla frequenza centrale
spectral_rolloff_mean	misura della media della frequenza al di sotto della quale vi è una determinata percentuale dell'energia spettrale totale
rmse_mean	media della misura normalizzata dell'energia del segnale in una finestra temporale
zcr_mean	media della velocità alla quale un segnale passa da positivo a zero a negativo, o da negativo a zero a positivo
Electronic labeler	variabile categoriche che indica se una traccia appartine al genere *Electronic* o meno

Tabella 1.2

È stato deciso di utilizzare solamente l'indice *mean* in quanto per le variabili indicate in Tabella 1.2 poiché gli altri indicatori erano ridondanti per la nostra analisi.

Non ci sono valori mancanti nel dataset.

Per quanto riguarda la variabile "*Electronic labeler*", circa il 26% delle tracce totali sono del genere **Electronic**.

1.2 Rilevamento dei valori anomali

È stato deciso di utilizzare 3 differenti metodi per il rilevamento dei valori anomali nel dataset:

1. IQR
2. ABOD
3. KNN

È stato deciso quindi di rimuovere dal dataset solo le istanze che erano identificate come valori anomali da tutti e tre i metodi adottati.

Il primo metodo dell'IQR ha riconosciuto un totale di 4562 outliers su un totale di 13129 records. Nella Figura 1.2 troviamo il boxplot delle variabili con il livello di magnitudine maggiore: osserviamo come il metodo dell'IQR non sia il piu adatto nell' identificare i valori anomali nel dataset.

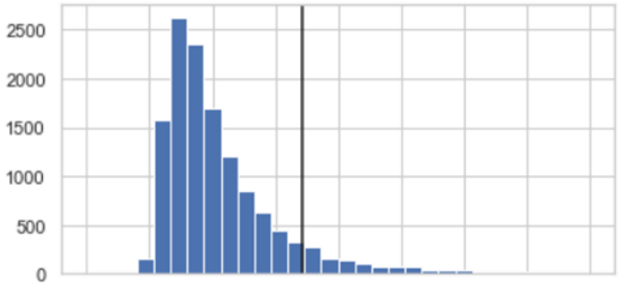

Figura 1.1

Il secondo metodo adottato, l'ABOD, dopo varie modifiche del threshold, ha identificato un totale di 1397 valori anomali. Il parametro 'contaminazione' è stato settato a 0.1, mentre con il metodo del KNN sono stati identificati un totale di 1109 valori anomali, utilizzando la distanza 'minkowski' ed un numero di vicini pari a 5. L'*outlier score* medio dei punti considerati outlier con il metodo del KNN è pari a 136.7, mentre quello dei punti non considerati outlier è pari a 45, con un valore massimo di 116.5.

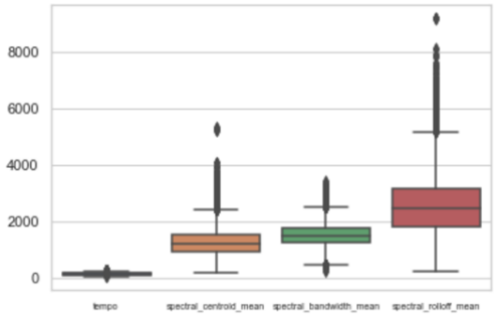

Figura 1.2

I valori anomali considerati tali da tutti i metodi utilizzati sono pari a 660. Tra questi 268 sono del genere 'Electronic', mentre 392 non lo sono: è stato quindi deciso di eliminare questi valori dal dataset.

1.3 Classification

Si è deciso di svolgere due differenti task di classificazione: il primo consiste nella classificazione delle tracce di genere **Electronic**, il secondo nella classificazione delle tracce registrate prima del 2011. In entrambi i casi i metodi di classificazione utilizzati sono Decision Tree e KNN.

Per lo svolgimento del secondo task è stato necessario creare una nuova variabile binaria, **pre2011**, che presenta il valore 1 nei casi in cui la canzone sia stata 'creata' negli anni dal 2008 al 2010 incluso, e presenta il valore 0 per gli anni successivi. Per generare questa variabile si è partiti dalla variabile *date_created* del subset *tracks*.

La distribuzione di *pre2011* è sbilanciata verso l'1: 8818 tracce sono state registrate tra il 2008 e il 2010, contro le 3651 registrate tra il 2011 ed il 2015.

1.3.1 Classificazione delle tracce del genere electronic

1.3.1.1 Decision Tree

Per ottimizzare le prestazioni del Decision Tree è stato necessario svolgere il tuning dei parametri principali, così da individuare la combinazione migliore.

Dopo aver sperimentato con diversi valori, e alla luce di quanto evidenziato dalle curve di *Precision-Recall*, *Cumulative Gains* e *Lift* (Figura 1.4), si è scelto di utilizzare come *test set* il 30% del dataset.

Si è optato per il metodo *Cross-Validation* con k pari a 5. Nella Tabella 1.3 sono rappresentati i migliori modelli ottenuti da 50 iterazioni. Il risultato della classificazione effettuata con i migliori parametri ha evidenziato che la variabile

	Mean Test Score	Standard Deviation Test Score	Min Sample Leaf	Min Sample Split	Criterion	Max Depth
0	0.797817	0.012842	80	200	entropy	18
1	0.796935	0.015099	70	10	gini	5
2	0.795251	0.010673	90	140	entropy	36

Tabella 1.3

danceability è la più rilevante ai fini della classificazione del genere *electronic*. Seguono in ordine di importanza *instrumentalness* e *acousticness* (la rilevanza delle variabili è rappresentata nella Figura 1.3).

Non vi sono significative differenze nel livello di **accuracy** tra Training Set (0.81) e Test Set (0.79), il che testimonia che non sono presenti fenomeni di overfitting o underfitting. Il modello è in grado di classificare il valore 1 della variabile target con una **precision** di 0.65, un **recall** di 0.34 e un F1-score di 0.45.

La *confusion matrix* è riportata nella Tabella 1.4.

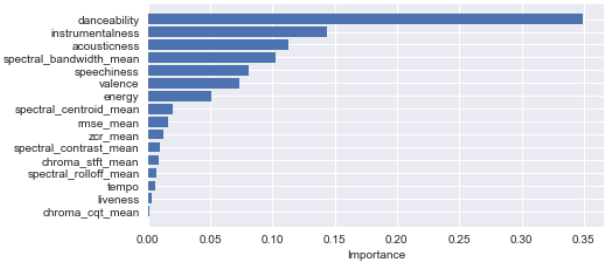

Figura 1.3

Come è normale aspettarsi quando la variabile target è sbilanciata verso il valore 0, il modello è molto più preciso nel predire quando un'osservazione non è pari a 1 piuttosto che nel predire quando lo è. In altre parole, il modello è più preciso nel classificare canzoni non appartenenti al genere della musica elettronica.

TN = 2648	FP = 168
FN = 608	TP = 317

Tabella 1.4

Sebbene il livello di accuracy sia piuttosto elevato, il livello di recall non è soddisfacente. Nel caso in questione, infatti, sarebbe preferibile un modello con un livello di recall più alto, anche a discapito della precision: se l'obiettivo è quello di classificare la musica elettronica è preferibile che il modello non 'escluda' tracce realmente elettroniche, anche a costo di 'includere' qualche canzone non elettronica in più.

La ROC curve è rappresentata nel paragrafo successivo in Figura 3.3, a confronto con il KNN.

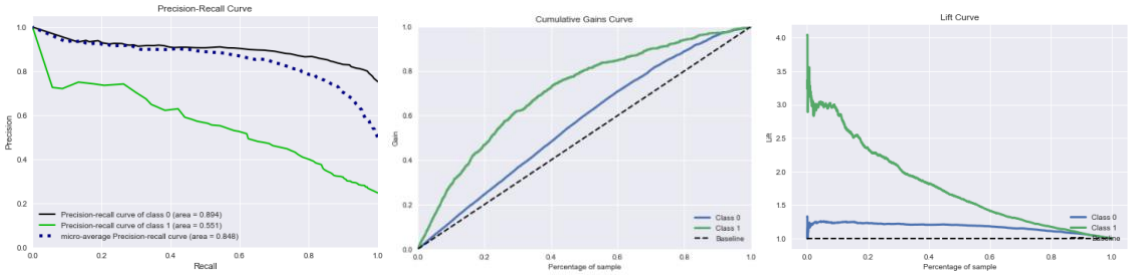

Figura 1.4

1.3.1.2 KNN

Per il KNN il dataset è stato standardizzato, è stata selezionata una *test size* del 33% ed è stato utilizzato il metodo *Cross-Validation* con k pari a 5. Sono stati provati iterativamente diversi valori per i parametri *n_neighbors*, *leaf_size*, *algorithm* (*'ball_tree', kd_tree', 'brute'*), *weights ('distance','uniform'*) e *metric ('euclidean', 'cosine', 'minkowski'*). Il modello migliore è risultato quello descritto dai seguenti parametri:

```
Mean validation score: 0.798 (std: 0.010)
Parameters: {'weights': 'distance', 'n_neighbors': 22,
       'metric': 'minkowski', 'leaf_size': 101, 'algorithm': 'ball_tree'}
```

TN = 2955	FP = 143
FN = 632	TP = 385

Tabella 1.5

Il livello di test accuracy riscontrato è di 0.81. La **precision** per la classificazione di *Electronic Labeler = 1* è pari a 0.73, il **recall** a 0.38 e **l'F1-score** a 0.50. Nella Tabella 1.5 sono riportati i valori della **confusion matrix**.

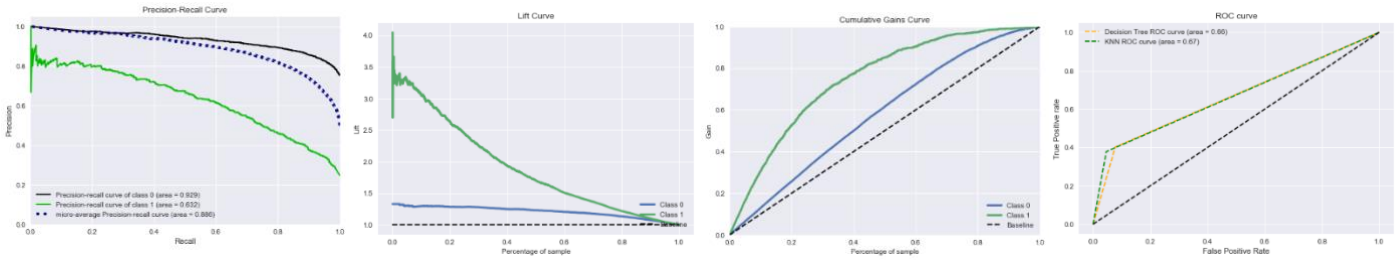

Figura 1.5

Figura 1.6

Nella Figura 1.5 sono riportate **Precision-Recall Curve**, **Cumulative Gains curve** e **Lift Curve**. Infine, nella Figura 1.6 sono rappresentate le **ROC curve** per Decision Tree e per KNN.

1.3.2 Classificazione delle tracce pre-2011

1.3.2.1 Decision Tree

Come descritto ad inizio capitolo, l'obiettivo è creare un modello in grado di individuare quali canzoni sono state prodotte negli anni 2008, 2009 e 2010. Dopo aver aggiunto la variabile *pre2011* si è proceduto con il tuning dei parametri nelle modalità indicate nei paragrafi precedenti. È stata utilizzata una *test size* del 40% e la *Cross-Validation* con k pari a 10. A seguito delle iterazioni sono risultati migliori i tre modelli descritti nella Tabella 1.6.

	Mean Test Score	Standard Deviation Test Score	Min Sample Leaf	Min Sample Split	Criterion	Max Depth
0	0.758283	0.070532	50	80	gini	9
2	0.757961	0.065759	200	70	gini	6
3	0.757961	0.065759	200	10	gini	38

Tabella 1.6

Anche in questa classificazione le variabili principali sono *acousticness* e *instrumentalness*.

In questo caso, la situazione è opposta a quella delle precedenti classificazioni: la classe da classificare, ossia *pre2011=1*, è la classe di maggioranza. Infatti, seppur il valore dell'**Accuracy** di 0.76 è simile a quanto riscontrato in precedenza, il livello delle altre metriche è decisamente più elevato: la **Precision** nella classificazione della classe 1 è pari a 0.78, il **Recall** a 0.93 e l'**F1-score** a 0.85. La **confusion matrix** è riportata nella Tabella 1.7.

TN = 239	FP = 953
FN = 508	TP = 3288

Tabella 1.7

La ROC curve sarà presentata nel paragrafo successivo.

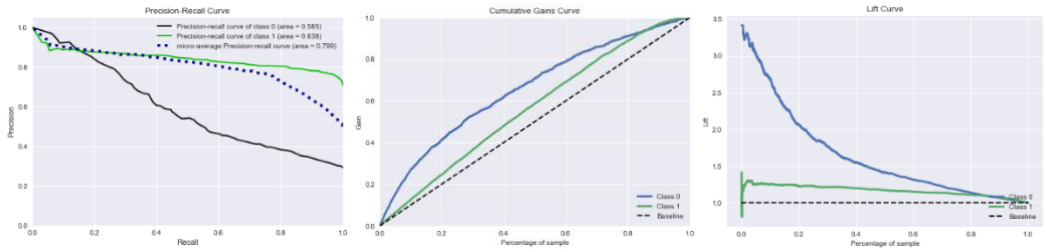

Figura 1.6

1.3.2.2 KNN

La scelta dei parametri, svolta come descritto nel Paragrafo 1.3.1.1, è ricaduta su una *test size* del 33% e una *Cross-Validation* con k pari a 5. Il modello più performante è quello che si ottiene con i seguenti parametri:

```
Mean validation score: 0.758 (std: 0.062)
Parameters: {'weights': 'uniform', 'n_neighbors': 23, 'metric': 'cosine',
       'leaf_size': 100, 'algorithm': 'brute'}
```

Il valore dell'**Accuracy** è di 0.77, mentre **Precision**, **Recall** e **F1-score** sono pari rispettivamente a 0.78, 0.95 e 0.86. La **confusion matrix** è rappresentata nella Tabella 1.8.

TN = 434	FP = 771
FN = 143	TP = 2767

Tabella 1.8

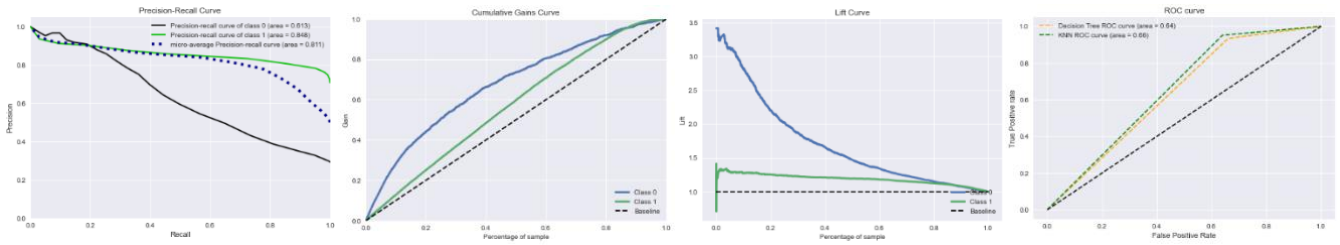

Figura 1.7

1.4 Imbalanced Learning

L'obiettivo della classificazione consiste nel riconoscere le tracce *Electronic*. Sono state utilizzate diverse tecniche di Imbalanced Learning, ed i risultati migliori sono stati raggiunti utilizzando la tecnica SMOTE, cui risultati verranno

esposti nei paragrafi dal 1.4.2 al 1.4.7, mentre nel paragrafo 1.4.8 verranno esposti i risultati delle varie tecniche utilizzate, comparandole tra loro.

1.4.1 Sbilanciamento del dataset

Prima di tutto, trasformiamo il dataset in modo tale che la classe target presenti uno sbilanciamento più elevato: una volta eliminata gran parte delle canzoni di genere "Electronic", la distribuzione della classe target è:

- **Other genre** (classe 0): 97%
- **Electronic** (classe 1): 3%

1.4.2 Oversampling - SMOTE

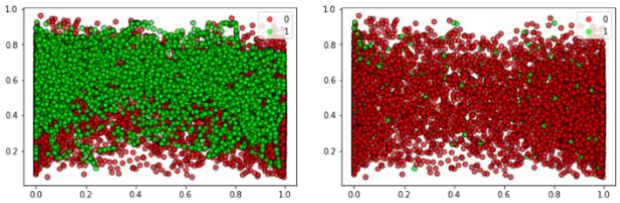

Figura 1.8

La Figura 1.8 confronta la distribuzione della classe target nel training set (una volta aumentate le osservazioni della classe 1 con la tecnica SMOTE) con la distribuzione della classe target nel training set sbilanciato. Il training set così trasformato conta 6570 osservazioni per ogni classe.

1.4.2.1 Scelta dei parametri per il Decision Tree Classifier

Nella Tabella 4.1 sono esposti i risultati delle tre migliori combinazioni di parametri individuate effettuando il tuninig servendosi della metodologia *Cross-Validation* con K = 5.

Per il modello verrà utilizzata la combinazione di *rank 1* effettuando una modifica sul parametro *min_impurity_decrease*, assegnando a quest'ultimo un valore di 0.007 per evitare l'overfitting.

Model rank	Mean Test Score	Standard Deviation Test Score	Min Sample Leaf	Min Sample Split	Criterion	Max Depth
1	0.901073	0.008864	10	10	entropy	13
2	0.882679	0.009878	10	60	entropy	12
3	0.882306	0.009470	10	40	entropy	10

Tabella 1.9

1.4.2.2 Validation sul training e test set

Analizzando la **confusion matrix** (Tabella 1.10) ed osservando gli indicatori esposti nella Tabella 1.10, possiamo vedere che tra il training set ed il test set vi è un livello simile di *accuracy*, il che aiuta ad evitare fenomeni di *overfitting* e *underfitting*. Inoltre, considerando che il training set è diviso 50% classe 0 e 50% classe 1, il modello classifica in modo corretto gran parte delle canzoni raggiungendo un livello di **Accuracy** sul training set di 0.79.

Metrics Test Set				
	Accuracy: 0.7767			
Class	precision	recall	f1-score	support
Other genre = 0	0.98	0.78	0.85	2816
Electronic = 1	0.08	0.57	0.14	90
macro avg	0.53	0.68	0.50	2906
weighted avg	0.95	0.78	0.85	2906

Metrics Training Set				
	Accuracy: 0.7928			
Class	precision	recall	f1-score	support
Other genre = 0	0.80	0.78	0.79	6570
Electronic = 1	0.78	0.81	0.80	6570
macro avg	0.79	0.79	0.79	13140
weighted avg	0.79	0.79	0.79	13140

Tabella 1.11

TN = 2205	FP = 610
FN = 39	TP = 51

Tabella 1.10

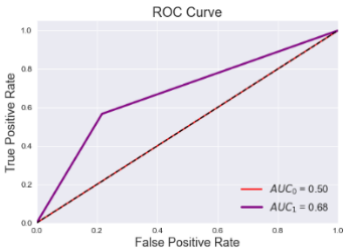

Figura 1.9

Mentre, considerando le prestazioni sul test set, il modello è in grado, in parte, di riconoscere le canzoni *Electronic* come tali, commettendo meno errori in termini di *false negative*, aumentando così il recall.

Tuttavia, notiamo un basso livello di **precision** (0.08) per la classe 1: il modello classifica come *Electronic* molte canzoni di un altro genere musicale, generando così un alto numero di falsi positivi.

Nella Figura 1.9 viene rappresentata la **ROC Curve** per il modello con i parametri sopra esposti (AUC_1), confrontata con la ROC curve del modello creato sul training set non bilanciato (AUC_0). Gli effetti del bilanciamento sono visibili, e grazie allo stesso si ottiene un livello di AUC di 0.68.

1.4.2.3 *Performance generali del modello di classificazione*

Come abbiamo potuto vedere nel paragrafo precedente, il modello presenta dei bassi valori di *precision* e *f1-score*. Si è quindi provveduto a studiare il modello in modo più flessibile, visualizzando le *ROC Curves* e la **Precision-Recall curve** in modo tale da capire come cambiano le performance del modello nel classificare la classe 1 (*Electronic*) per diversi livelli di *threshold*.

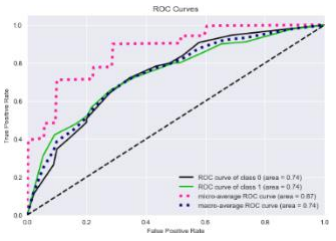

Figura 1.10

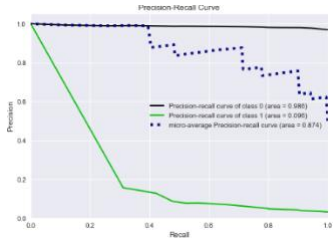

Figura 1.11

Nella Figura 1.10 sono rappresentate le curve di entrambe le classi: osservando la ROC Curve per la classe 1, notiamo come per alti livelli di threshold il classificatore commette pochi errori, generando un basso *FPR*. Invece nella Figura 1.11 possiamo vedere come la *Precision – Recall Curve* sia bassa, il che dimostra che non è possibile ottenere un risultato sufficientemente alto per entrambe le metriche. Per capire meglio come cambiano i valori di *precision* e *recall* per determinati livelli di threshold, osserviamo la Figura 1.14: la recall diminuisce velocemente non appena il threshold aumenta, viceversa aumenta gradualmente la *precision* del classificatore all'aumentare del threshold.

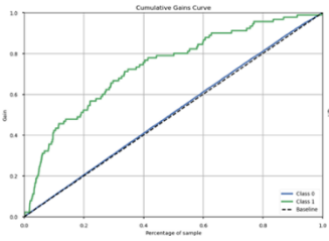

Figura 1.12

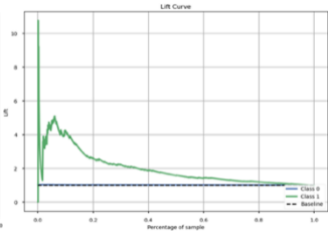

Figura 1.13

Possiamo vedere dalla **Cumulative Gains Curve** (Figura 1.12) che per la classe 1 la selezione del primo 40% delle canzoni osservate ha un'alta probabilità di essere classificato come *Electronic*, e che contiene quasi l'80% delle canzoni che sono effettivamente di genere *Electronic*. Osservando anche la **Lift Curve** (Figura 1.13), l'utilizzo del modello produce potenzialmente dei risultati quasi 2 volte migliori rispetto alla selezione casuale.

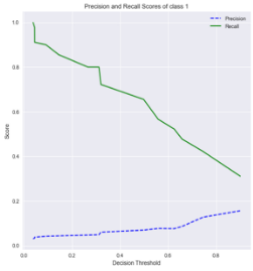

Figura 1.14

1.4.2.4 *Scelta del threshold e Decision Tree con threshold modificato*

Visto l'andamento di *precision* e *recall* nella Figura 1.14, modifichiamo il threshold decisionale sul test set, così da aumentare la precisione del modello nel riconoscere la classe 1, e cercando allo stesso tempo di non diminuire di molto la sensibilità del classificatore. Per il nostro obiettivo individuiamo un threshold con un valore di 0.75, punto in cui la *precision* è in salita e il *recall* ha un valore sopra lo 0.4.

Nella Tabella 1.12 vengono riassunte le metriche di valutazione del modello al quale è stato modificato il threshold decisionale. Dal confronto con il modello con il threshold di default a 0.5 (paragrafo 1.4.2.2) emerge che per la classe 1 la modifica ha portato a dei risultati migliori in termini di *precision* e *f1-score*, dati dal fatto che il classificatore commette meno errori in termini di *false positive*. In conclusione, si osserva che aumentare il threshold per avere un modello più preciso porta ad una lieve diminuzione del TPR e dell'AUC (figura 1.16).

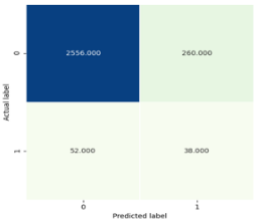

Figura 1.15

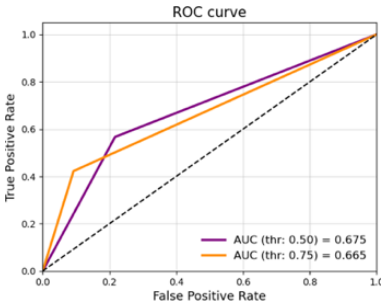

Figura 1.16

Accuracy: 0.8926			
Class	**precision**	**recall**	**f1-score**
Other genre = **0**	0.98	0.91	0.94
Electronic = **1**	0.13	0.42	0.20

Tabella 1.12

1.4.3 Conclusione su Imbalanced Learning

Qui riportiamo la tabella 1.13 dove sono riepilogati i modelli provati con i rispettivi risultati ed alle figure 1.17 e 1.18 sono riportate e confrontate le *Precision-Recall Curves* e le *ROC Curves* dei diversi modelli provati.

Osservando le ROC Curves, notiamo che tutti i modelli provati si comportano in modo simile per alti valori di threshold, generando un basso livello FP, ma superato un certo livello di TPR commettono molti errori.

Dal confronto delle Precision-Recall curves emerge che il classificatore implementato sul dataset ribilanciato con SMOTE è l'unico che permette di ottenere di ottenere un livello sufficientemente alto di Recall senza ridurre notevolmente la Precision. Buoni risultati sono stati ottenuti anche modificando il paramentro "*class_weight*".

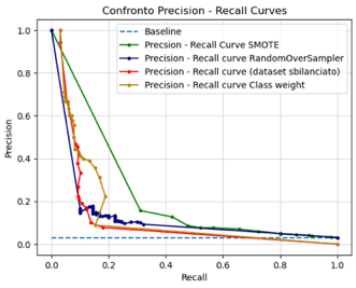

Figura 1.17

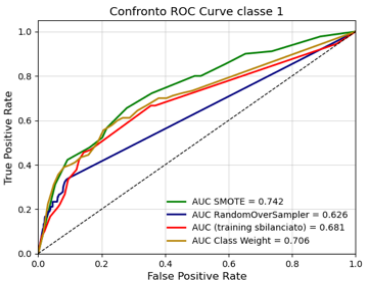

Figura 1.18

		Metriche test set				
Tecnica	**Modello**	**Accuracy**	**Precision (class 1)**	**Recall (class 1)**	**F-1 score (class 1)**	**Confusion matrix**
Dataset sbilanciato	Decision Tree Classifier (min_samples_split = 30, min_samples_leaf = 40, max_depth = 5, criterion = gini)	0.9690	0.00	0.00	0.00	TN = 2816 FN = 90 FP = 0 TP = 0
RandomOverSampler	Decision Tree Classifier (min_samples_split = 40, min_samples_leaf = 10, max_depth = 29, criterion = entropy)	0.9191	0.09	0.17	0.11	TN = 2646 FN = 75 FP = 160 TP = 15
RandomOverSampler + cambio threshold a 0.20	Decision Tree Classifier (min_samples_split = 40, min_samples_leaf = 10, max_depth = 29, criterion = entropy)	0.9194	0.13	0.21	0.16	TN = 2650 FN = 81 FP = 153 TP = 22
SMOTE	Decision Tree Classifier (min_samples_split = 10, min_samples_leaf = 10, max_depth = 13, criterion = entropy, min_impurity_decrease = 0.007)	0.7767	0.08	0.57	0.14	TN = 2206 FN = 39 FP = 610 TP = 51
SMOTE + cambio threshold a 0.75	Decision Tree Classifier (min_samples_split = 10, min_samples_leaf = 10, max_depth = 13, criterion = entropy, min_impurity_decrease = 0.007)	0.8926	0.13	0.42	0.20	TN = 2556 FN = 52 FP = 260 TP = 38
Modifica "Class Weight"	Decision Tree Classifier (class_weight = {0: 1, 1: 11}, min_samples_split = 110, min_samples_leaf = 120, max_depth = 10, criterion = 'entropy')	0.8451	0.10	0.49	0.16	TN = 2412 FN = 46 FP = 404 TP = 44

Tabella 1.13 Classificatori

2 Advanced Classification Methods

2.1 Naive Bayes

Il classificatore Naive Bayes è stato applicato ad un dataset composto esclusivamente da attributi numerici continui (ad eccezione della variabile target). Per questa ragione, è stato implementato nella declinazione **Gaussian Naive Bayes**, la quale suppone che le probabilità delle variabili siano distribuite secondo una funzione gaussiana.

TN = 3180	FP = 762
FN = 548	TP = 747

Tabella 2.1

Dopo aver sperimentato con diversi *test size*, si è optato per il 42%, e l'**Accuracy** ottenuta è stata del 75%. I valori di **Precision**, **Recall** e **F1-score** per la classificazione della variabile target pari a 1 si sono attestati rispettivamente allo 0.5, 0.58 e 0.53.

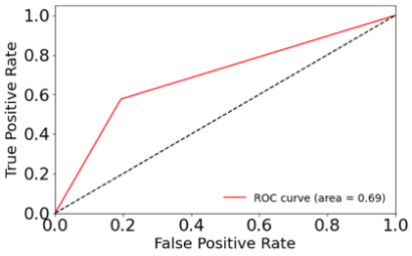

Figura 2.2.1

La composizione della **confusion matrix** è rappresentata nella Tabella 2.1. La ROC curve è rappresentata nella Figura 2.1.

2.2 Logistic Regression

Attraverso la Logistic Regression è stata riprodotta la classificazione effettuata nel paragrafo 1.3.1, ossia la classificazione delle canzoni di genere "Electronic".

Il data set è stato diviso in 70% training set e 30% test set ed il tuning è stato effettuato utilizzando il *Repeated Stratified KFold* con k = 5 sui seguenti parametri: **Solver**, scelto tra *lbfgs* e *liblinear*; *C*, scelto su una scala logaritmica da -4 a 20; **Penalty**, scelto tra *l1* ed *l2*.

Model rank	Mean Test Score	Standard Deviation Test Score	C	Solver	Penalty
1	0.802269	0.009374	0.233572	lbfgs	l2
2	0.802258	0.009278	0.233572	liblinear	l1
3	0.802246	0.009486	0.615848	liblinear	l2

Tabella 2.2

Con le combinazioni in tabella 2.2 si ottengono gli stessi risultati per le metriche di valutazione della classificazione, e si è deciso quindi di scegliere la combinazione con *rank 1*.

2.2.1 Validation con Training Test e Test Set

Nella Tabella 2.3 vengono riassunte le metriche di valutazione del classificatore. Anzitutto, notiamo che non vi sono particolari differenze di **Accuracy** tra training set e test set, quindi il modello è in grado di generalizzare. Dalle metriche esposte, è possibile affermare che il modello classifica con precisione le canzoni appartenenti alla classe 1 *Electronic*. Tuttavia, l'alta precisione fa sì che il modello classifichi poche canzoni come *Electronic*, producendo un alto numero di *false negative* (**confusion matrix**, Figura 6.1) e abbassando il recall.

Metrics Test Set				
	Accuracy: 0.8126			
Class	precision	recall	f1-score	support
Other genre = 0	0.83	0.94	0.88	2816
Electronic = 1	0.70	0.42	0.53	925
macro avg	0.77	0.68	0.70	3741
weighted avg	0.88	0.81	0.79	3741

Metrics Training Set				
	Accuracy: 0.8061			
Class	precision	recall	f1-score	support
Other genre = 0	0.83	0.94	0.94	6570
Electronic = 1	0.68	0.41	0.51	2158
macro avg	0.75	0.67	0.69	8728
weighted avg	0.79	0.81	0.79	8728

Tabella 2.3

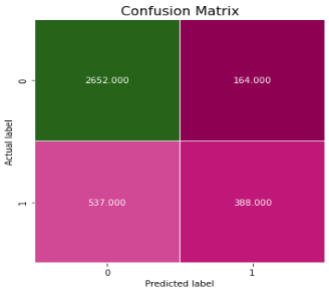

Figura 2.2

2.2.2 Performance generali del modello di classificazione

Nella Figura 2.3 vengono rappresentate le ROC curve di entrambe le classi. Osservando la ROC Curve della classe 1, notiamo come il modello classifichi correttamente molte canzoni *Electronic* commettendo pochi errori fino ad un livello di TPR pari a circa 0.65, ed infatti le canzoni appartenenti ad altri generi classificate come elettroniche sono 164.

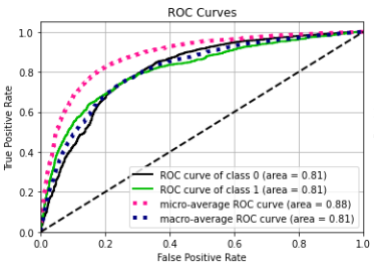

Figura 2.3

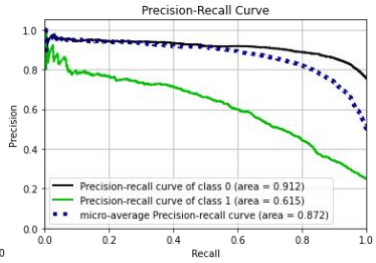

Figura 2.4

Dalla Precision – Recall Curve (Figura 2.4), dato che non vi è un significativo sbilanciamento tra le classi, notiamo come la curva diminuisca gradualmente mostrando che è possibile ottenere dei livelli sufficientemente alti per entrambe le metriche.

Con *Cumulative Gains Curve* e la *Lift Curve* emerge come l'utilizzo del classificatore, per entrambe le classi, migliori effettivamente i risultati rispetto alla selezione casuale. Focalizzandoci solamente su *Cumulative Gains* e *Lift curve* della classe 1 e selezionando il primo 20% delle canzoni con una più alta probabilità di appartenere alla classe *Electronic* si ottiene che più del 50% di queste lo siano effettivamente. Infatti, la *Lift Curve* indica che, sempre selezionando il primo 20% delle canzoni con una più alta probabilità di appartenere alla classe "Electronic", si ottengono risultati più di 2,5 volte migliori rispetto alla selezione casuale.

2.2.3 Modifica threshold

Mantenendo un threshold decisionale a 0.5 si ottiene un classificatore molto preciso, in quanto commette pochi errori nel classificare le canzoni appartenenti alla classe 0 come canzoni appartenenti alla classe 1. Infatti, osservando la ROC Curve nella Figura 2.5 (AUC thr: 0.50) si nota un livello di FPR prossimo allo 0.1.

Dato il nostro obiettivo di classificazione, un più alto livello di *recall* è preferibile cosicché il modello non commetta errori nel classificare tracce effettivamente elettroniche come appartenenti ad altri generi, anche a costo di aumentare i FP.

Settando un threshold a 0.35 si ottiene un classificatore che presenta per la classe 1 **precision** = 0.57, **recall** = 0.59 e **F1-score** = 0.58. Come vediamo nella **confusion matrix** (Tabella 2.4), e confrontando con il classificatore con threshold a 0.5, i TP e FP sono aumentati ottenendo dei risultati migliori in termini di *recall*, *F1 score* e *ROC Curve* di 0.72 (Figura 2.6, AUC thr: 0.35), migliorando la classificazione della classe *Electronic*.

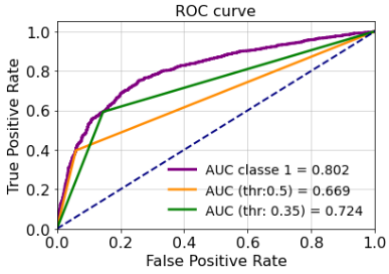

Figura 2.5

TN = 2407	FP = 409
FN = 377	TP = 548

Tabella 2.4

2.3 Ruled Based Classifier

Per la Ruled Based Classification è stato deciso di creare un dataset differente, formato soltanto da variabili categoriche: le variabili continue usate sono state discretizzate con il metodo dei quantili.

Il nuovo dataset presenta un totale di 9383 righe e 6 colonne, di queste tracce circa 5 mila sono tracce in comune con il dataset utilizzato per le altre classificazioni. Non vi sono dati mancanti.

Le variabili sono:

1. “*Season*” ricavata estraendo le diverse stagioni basandosi sul mese da ‘track’, 'date_created' in tracks
2. “*Duration*” ricavata da ‘track’, ‘duration’ in tracks e discretizzata per ‘Short Duration’, ’Medium Duration’ e ‘Long Duration’
3. “*nome_cantante*” ricavata da ‘artist’,’name’ in tracks
4. “*Genre*” ricavata da 'track','genre_top' in tracks
5. “*Language*” ricavata da 'track','language_code' in tracks e successivamente discretizzata per ‘English’ e ‘Not English’
6. “*Popularity*” ricavata sommando le variabili continue ‘comments’,’favorites’,’interest’ e ’listens’ da ‘track’ in tracks e successivamente discretizzando per 'Low popularity', 'Medium popularity' e 'High popularity'; è stata utilizzata come variabile target nella Ruled Based Classification.

L’ algoritmo utilizzato è stato il CN2, il metodo utilizzato non ha incluso la colonna “nome_cantante” all’ interno della classificazione.

Sono state effettuate diverse prove modificando il parametro ‘**min_covered_examples’**, che trova solo le regole che coprono almeno un certo numero di esempi, e ‘**max_rule_length’**, con il quale vengono identificate solo le regole con almeno un certo numero di condizioni. Di seguito alcune delle regole ottenute con diverse combinazioni di questi 2 parametri:

- min_covered_examples = 400; max_rule_length = 2:

IF	THEN
Genre==Electronic AND Season==Spring	Popularity=Medium popularity
Genre==Rock AND Season==Winter	Popularity=Medium popularity
Season==Winter AND Genre!=Experimental	Popularity=High popularity
Season!=Summer AND Genre==Folk	Popularity=Medium popularity
Genre==Experimental AND Duration==Medium duration	Popularity=Low popularity
Genre==Rock AND Duration==Short duration	Popularity=Medium popularity
Season==Autumn AND Genre==Rock	Popularity=Medium popularity
Genre==Experimental AND Season!=Winter	Popularity=Low popularity
Season==Spring AND Genre!=Rock	Popularity=High popularity
Season==Spring AND Language==English	Popularity=Medium popularity
Genre==Electronic AND Duration!=Short duration	Popularity=Medium popularity
Season==Autumn AND Genre!=Spoken	Popularity=Medium popularity
Genre!=Rock AND Season==Summer	Popularity=Medium popularity
Language==English AND Duration!=Short duration	Popularity=Low popularity

- min_covered_examples = 600; max_rule_length = 3:

IF	THEN
Genre==Rock AND Season==Winter AND Language==English	Popularity=Medium popularity
Genre==Electronic AND Season!=Autumn AND Season!=Winter	Popularity=Medium popularity
Season==Winter AND Genre!=Experimental AND Genre!=Electronic	Popularity=High popularity
Genre==Rock AND Duration==Short duration AND Language==English	Popularity=Medium popularity
Genre==Experimental AND Season!=Winter AND Duration!=Long duration	Popularity=Low popularity
Genre==Rock AND Season==Autumn AND Language==English	Popularity=Medium popularity
Season==Spring AND Genre!=Rock AND Genre!=Experimental	Popularity=Medium popularity
Season==Autumn AND Duration!=Medium duration AND Genre!=Soul-RnB	Popularity=Medium popularity
Genre!=Rock AND Genre!=Experimental AND Genre!=Folk	Popularity=Medium popularity

- min_covered_examples = 800; max_rule_length = 4:

IF	THEN
Season==Winter AND Genre!=Experimental AND Genre!=Pop AND Genre!=Rock	Popularity=High popularity
Season==Spring AND Genre!=Experimental AND Genre!=Instrumental AND Genre!=Folk	Popularity=Medium popularity
Genre==Rock AND Season==Autumn AND Duration!=Long duration AND Language!=Not English	Popularity=Medium popularity
Genre!=Rock AND Genre!=Experimental AND Season!=Autumn AND Duration!=Short duration	Popularity=Medium popularity
Duration==Long duration AND Genre!=Soul-RnB AND Language==English AND Genre!=Country	Popularity=Medium popularity
Genre!=Rock AND Genre!=Spoken AND Genre!=Instrumental AND Genre!=Electronic	Popularity=Medium popularity

Di seguito la distribuzione delle classi identificate utilizzando le diverse combinazioni:

1. Con min_covered_examples = 400 e max_rule_length = 2 il 64% delle classi trovate è 'Medium popularity' e solo il 18% ciascuno di 'Low popularity' e 'High popularity'
2. Con min_covered_examples = 600 e max_rule_length = 3 l'78% delle classi trovate è 'Medium popularity', e solo l'11% ciascuno di 'Low popularity' e 'High popularity'
3. Con min_covered_examples = 800 e max_rule_length = 4 l'83% delle classi trovate è 'Medium popularity' e il 17% di 'High popularity', nessuna regola con almeno 4 condizioni da 'Low popularity'.

Non è stata trovata nessuna regola che avesse la condizione Language==Not English, probabilmente a seguito della distribuzione stessa della variabile 'Language' nel dataset che presenta nel piu del 90% delle tracce il genere 'English'.

2.4 Support Vector Machines

In questo paragrafo andremo a classificare le canzoni di genere "Electronic".

Model rank	Mean Test Score	Standard Deviation Test Score	C	Kernel	Gamma	Degree
1	0.810862	0.005653	10	rbf	0.010	/
2	0.810701	0.006597	1000	rbf	0.001	/
3	0.810564	0.006646	100	rbf	0.010	/
4	0.798751	0.001980	1000	poly	0.010	6
5	0.798751	0.001829	10	poly	0.010	3

Tabella 2.5

Anzitutto, così come per gli altri classificatori, ricerchiamo la combinazione migliore dei parametri: **Kernel** scelta tra linear, *rbf*, *poly* e *linear*; **Gamma** [1, 0.1, 0.01, 0.001, 0.0001]; **C** [0.1, 1, 10, 100, 1000] e **Degree** [1, 2, 3, 4, 5, 6] (nel caso di kernel = poly).

I risultati esposti nella Tabella 8.1 sono stati ottenuti implementando il *10 Cross-Validation* con 50 iterazioni. Dalla tabella esposta notiamo che tra le combinazioni maggiormente performanti ottenute non vi è alcuna con kernel = "linear": ciò indica che al fine di un buon risultato occorre ricercare un iperpiano non lineare ricorrendo al "*kernel trick*", così da mappare le caratteristiche in uno spazio multidimensionale.

Nei paragrafi successivi, verranno esposti i risultati del modello creato con la combinazione di *rank 1*.

2.4.1 Validation con training set e test set

Metrics Test Set				
Accuracy: 0.8131				
Class	precision	recall	f1-score	support
Other genre = 0	0.83	0.95	0.88	2816
Electronic = 1	0.70	0.42	0.51	925
macro avg	0.77	0.67	0.70	3741
weighted avg	0.80	0.81	0.79	3741

Metrics Training Set				
Accuracy: 0.8198				
Class	precision	recall	f1-score	support
Other genre = 0	0.83	0.94	0.89	6570
Electronic = 1	0.74	0.41	0.53	2158
macro avg	0.79	0.68	0.71	8728
weighted avg	0.81	0.82	0.80	8728

Tabella 2.6

Nella tabella riepilogativa delle performance (Tabella 2.6) notiamo che i valori di tutte le metriche sono simili tra test set e training set, il che porta ad affermare che non vi è presenza di overfitting e underfitting: ciò è ricollegabile al basso valore di *gamma*, il quale aiuta a non effettuare un *fit* esatto solo sulle osservazioni del training set.

Il modello risulta essere più preciso (precision = 0.70) che sensibile (recall = 0.42), e questo più basso valore di recall è osservabile dalla *confusion matrix* (Tabella 2.7): molte canzoni di genere "Electronic" (classe 1) vengono classificate come canzoni di un altro genere (classe 0). Questo risultato è dato dal basso valore di regolarizzazione C (ricordiamo C = 10 nel nostro modello), il quale crea un iperpiano con un margine più ampio consentendo l'errata classificazione di molte osservazioni.

TN = 2673	FP = 143
FN = 556	TP = 369

Tabella 2.7

Infatti, settando C = 1000 e lasciando invariati gli altri parametri, è possibile ottenere, per la classe 1, i seguenti risultati: **precision** = 0.67, **recall** = 0.47, **F1-score** = 0.55 e **Accuracy** = 0.821. La confusion matrix è riportata nella Tabella 8.4. Come possiamo vedere, ridurre il margine nell'iperpiano ha portato ad un risultato migliore con un recall più elevata e alla classificazione di più canzoni "Electornic" come tali, il tutto commettendo meno errori in termini di *false negative*.

TN = 2605	FP = 211
FN = 490	TP = 435

Tabella 2.8

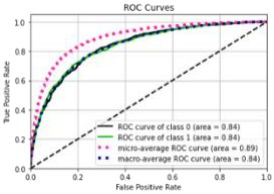

Figura 2.7

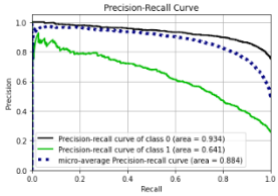

Figura 2.8

Dalla *ROC Curve* notiamo che il modello presentato inizia a commettere molti errori superato un TPR pari a circa 0.70, questo per entrambe le classi. Invece, la *Precision – Recall Curve* suggerisce che per dati livelli di treshold è possibili ottenere alti valori per entrambe le metriche.

Mentre *Cumulative Gains* e *Lift Curve* suggeriscono che che con l'utilizzo del modello si ha un risultato due volte e mezzo migliore rispetto alla selezione casuale.

2.5 Ensemble classifiers

L'apprendimento Ensemble sfrutta le predizioni di molteplici classificatori per ottenere prestazioni migliori di quelle basate su un singolo classificatore. Alla base di tale metodo c'è il principio di "saggezza della folla", secondo il quale la conoscenza della collettività supera la conoscenza individuale, anche nel caso in cui questo singolo sia esperto della materia. Un Ensemble classifier costruisce una serie di *base classifiers* ed effettua la classificazione finale tramite il voto di maggioranza dei diversi classificatori.

I metodi utilizzati al fine di classificare la variabile target "Electronic labeler" sono stati Random Forest, Bagging e Boosting (nella versione AdaBoost).

2.5.1 Random Forest

A differenza di Bagging e Boosting, basati sulla creazione di diversi training set ottenuti dal resampling del dataset originale, il metodo Random Forest seleziona di volta in volta solamente un subset delle variabili. In particolare, ogni Decision Tree è costruito su *bootstrap samples*, in cui viene mantenuta la cardinalità, ma sono selezionati dei vettori random per formare il dataset.

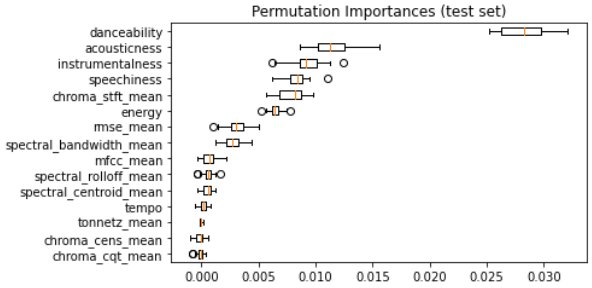

Figura 2.9

Dopo aver realizzato 100 alberi di decisione, secondo l'algoritmo Random Forest le variabili più significative (Figura 2.9) sono ancora quelle estratte dal dataset *echonest*, *danceability* su tutte.

Con test size del 40% e *Cross-Validation* con k = 10, i parametri che garantiscono le prestazioni migliori:

- ***min_samples_split***: 2
- ***min_samples_leaf*****:** 5
- ***max_depth***: 3

L'accuracy ottenuta si attesta all'82%. La **Precision** nella classificazione del valore 1 della variabile target è pari a 0.74, il **recall** a 0.4 e l'**F1-score** allo 0.52. La **confusion matrix** è riportata nella Tabella 2.9.

TN = 3585	FP = 170
FN = 753	TP = 490

Tabella 2.9

Nella Figura 2.10 è rappresentata la ROC curve per il metodo Random Forest, la quale presenta un valore in linea con le classificazioni precedenti.

Si è optato per l'utilizzo di 100 *base_classifiers* poiché, dopo aver fatto diversi tentativi (Figura 2.11), si è notato che i valori dell'accuratezza non si allontanavano significativamente dall'82% per valori prossimi al 100 di *n_estimators*.

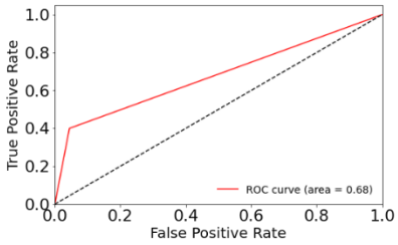

Figura 2.10

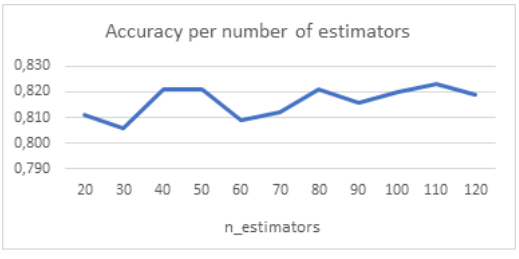

Figura 2.11

2.5.2 Bagging

Il Bootstrap Aggregating, o Bagging, è un metodo di classificazione basato sul *sampling* ripetuto del dataset secondo una distribuzione uniforme delle probabilità. In questo caso, a differenza del metodo Random Forest, ogni *bootstrap sample* mantiene le dimensioni originali del dataset. Inoltre, è possibile che alcune osservazioni appaiano più volte e che quindi altre siano escluse dalla classificazione.

Dopo aver standardizzato il dataset, sono stati utilizzati 4 *base classifiers*: Decision Tree, Random Forest, Support Vector Classifier e KNN. Per ogni classificatore il parametro *n_estimators* scelto è stato 20.

Per ogni Random Forest sono stati generati 100 alberi (moltiplicati per le 20 iterazioni del Bagging). Per SVC e KNN sono stati utilizzati i parametri migliori, ossia quelli emersi nelle analisi precedenti.
Nella Tabella 2.10 sono riportati i risultati ottenuti. I valori di precision, recall e F1-score sono riferiti alla classe 1.

Classificatore	Accuracy	Precision	Recall	F1-score
Decision Tree	0.806	0.66	0.44	0.53
Random Forest	0.819	0.74	0.42	0.53
SVC	0.799	0.61	0.52	0.56
KNN	0.813	0.74	0.37	0.50

Tabella 2.10

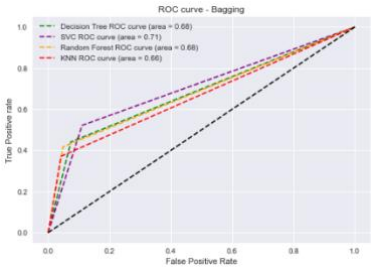

Figura 2.11

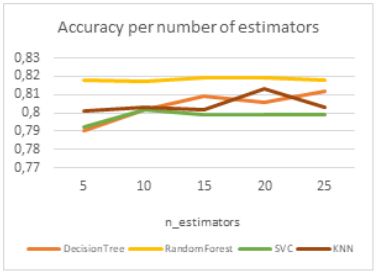

Figura 2.12

Il risultato evidenzia come non sempre il modello con il valore di accuracy più elevato sia quello da preferire. Nel caso in questione, infatti, l'obiettivo è quello di classificare le tracce di genere *electronic*, che ricordiamo essere la minoranza del dataset. Sarebbe quindi da preferire il modello con un recall più elevata, anche a costo di "accettare" qualche falso positivo in più. Per questo motivo SVC, ossia il modello con l'accuracy più bassa, è il classificatore da preferire.
Nella Figura 2.11 è riportato il confronto tra le ROC curve dei quattro modelli.
Si è scelto di utilizzare lo stesso numero di *base classifiers* per ognuno dei quattro algoritmi per poter confrontare i risultati. Il parametro 20 è stato scelto come compromesso tra efficienza e livello di accuracy. Nella Figura 2.12 sono riportati i livelli di accuracy per diversi valori di *n_estimators*.

2.5.3 Boosting

Il Boosting è basato sullo stesso principio del Bagging, con l'aggiunta che ad ogni iterazione viene aumentato il peso dei casi più difficili da classificare. In questo modo all'inizio del processo ogni record ha la stessa probabilità di essere selezionato, mentre con il passare delle iterazioni le osservazioni che vengono misclassificate più volte hanno più probabilità di essere selezionate.
Nell'Adaptive Boosting, o AdaBoost, il peso di un classificatore dipende dal suo *errore rate*, cossicché siano penalizzati i modelli con una scarsa accuracy, ossia quelli generati durante le prime iterazioni.
AdaBoost utilizza gli *stumps*, ossia alberi di decisione molto semplici formati da un nodo e due foglie, i quali costituiscono la moltitudine di *week learners* da cui ottenere la classificazione finale.

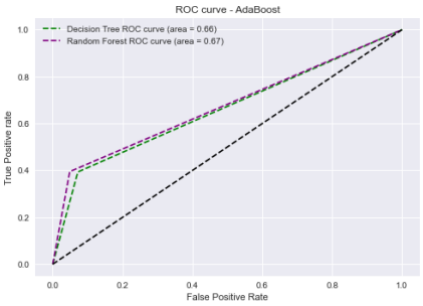

Figura 2.13

L'algoritmo AdaBoost è stato implementato sia con Decision Tree che con Random Forest. Il dataset è stato standardizzato e per ogni classificatore è stato impostato *n_estimators = 20*. Come nel paragrafo precedente, ogni iterazione di Random Forest ha generato 100 alberi.
Nella Tabella 9.3 sono riportati i risultati ottenuti.

Classificatore	Accuracy	Precision	Recall	F1-score
Decision Tree	0.796	0.64	0.39	0.49
Random Forest	0.813	0.73	0.39	0.51

Tabella 2.11

L'analisi delle metriche sopra riportate e la comparazione delle ROC-curve (Figura 2.12) portano alla stessa conclusione: per quanto i risultati siano simili, Random Forest (prevedibilmente) è il classificatore migliore tra i due utilizzati.

2.6 Neural Networks classification

Il dataset è stato standardizzato, sono stati effettuati diversi test sia modificando manualmente i parametri che utilizzando un *grid search*, sono state provate diverse grandezze per il *test size* ed alla fine è stato scelto di utilizzare come *test size*

il 30% del dataset. Le combinazioni provate dei parametri sono le seguenti: **activation** scelto tra *tanh*, *relu* e *logistic*; **solver**: scelto tra *sgd* e *adam*; **hidden layer sizes**: (128,64,32), (128,64), (64,32), (64) e (32); learning rate scelto tra *constant* e *adaptive*; **alpha**: 0.0001 e 0.05
È stato utilizzato il metodo della *Cross-Validation* con k=3, qui di seguito la combinazione di parametri con i quali sono stati ottenuti i migliori risultati: **activation**: *'relu'*; **solver**:*'adam'*; **hidden layer size**: 1 hidden layer da 64 unità; **learning rate**:*'costant'*; **alpha**:0.05.

2.6.1 Modello senza early stopping

È stata quindi ottenuta un'**accuracy** di 0.86 su training set e di 0.82 sul test set. La classificazione della variabile target pari a 1 sul test ha una **precision** pari a 0.69, un **recall** di 0.51 e un **F1-score** pari a 0.59 la ***loss*** *è* di 0.34 (figura 2.14).
La **confusion matrix** viene rappresentata nella Tabella 2.12.

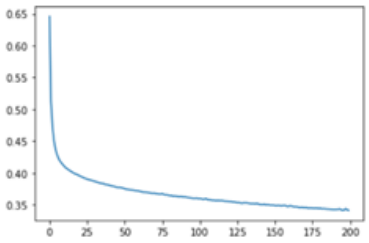

Figura 2.14

TN = 2605	FP = 211
FN = 449	TP = 476

Tabella 2.12

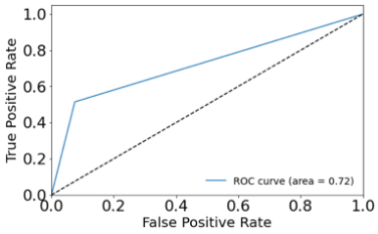

Figura 2.2.25

2.6.2 Modello con early stopping

È stato successivamente performato lo stesso modello, applicando stavolta l'*early stopping.*
L'*early stopping* mette da parte automaticamente il 10% di dati nel training set come validation set e termina il training quando il validation score non aumenta di almeno un certo livello di tolleranza (lasciata come default nel modello) dopo almeno 10 interazioni (anche il valore 10 è di default).
Il modello ottenuto ha un'**accuracy** sul training set pari a 0.83 e sul test pari a 0.82. La classificazione della variabile target pari a 1 sul test ha una **precision** pari a 0.70, un **recall** di 0.48 e un **F1-score** pari a 0.57.

TN = 2625	FP = 191
FN = 477	TP = 448

Tabella 2.13

Il modello presenta un numero di iterazioni inferiore (50 contro 200), ha ottenuto un punteggio della ROC curve inferiore pari a 0.68 ed un livello di *loss* leggermente maggiore pari a 0.37, ma è stato così possibile evitare l'overfitting presente nel modello precedente: notiamo infatti che nella *loss curve* in Figura 2.14 all'altezza della 50esima iterazione si ottengono livelli simili in termini di *loss* (la curva è quasi orizzontale), e che di conseguenza l'applicazione dell' early stopping ha contribuito a ridurre l'overfitting.
La **confusion matrix** viene rappresentata nella Tabella 2.13.

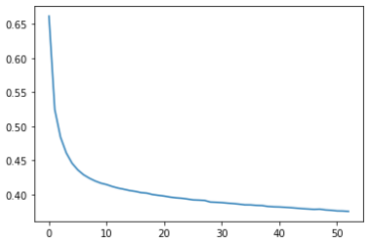

Figura 2.16

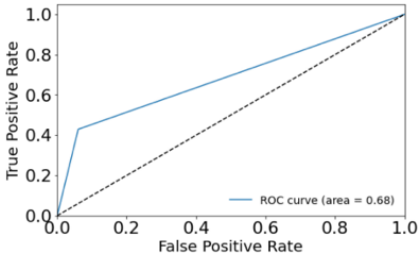

Figura 2.17

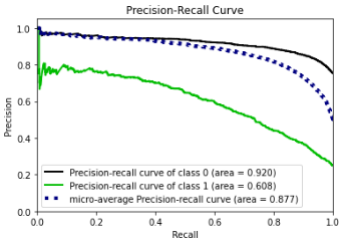

Figura 2.18

2.7 Regressione Lineare

Il dataset è stato standardizzato prima di effettuare la regressione.

2.7.1 Regressione lineare semplice

Si è deciso di studiare la relazione tra la variabile indipendente ***mfcc_mean*,** la quale misura la media degli spretti a breve termine di potenza del suono, e la varibile dipendente ***energy*,** la quale misura l'intensità di una canzone.

La positività del coefficiente indica che vi è una relazione positiva tra le due varibili: quando il valore di *mfcc_mean* aumenta di una unità, aumenta anche la media della variabile dipendente *energy* per un valore pari a 0.63.
Prendendo in considerazione il coefficiente di determinazione R^2, osserviamo che tra le due variabili non vi è una relazione positiva forte, il che la variabile indipendente riesce a spiegare solo il 40% della varianza della variabile dipendente.

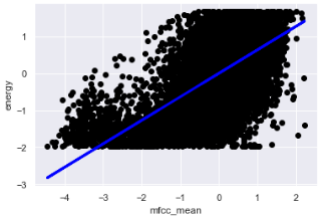

Figura 2.19

Coefficient:	0.6353
MSE:	0.60
MAE:	0.64
R-squared	0.40

Tabella 2.14

2.7.2 Regressione lineare con regolarizzatori

Implementando con la regolarizzazione Ridge la regressione lineare di cui al punto precedente, i valori di MSE, MAE ed R^2 non cambiano. Vi è solo un cambiamento non rilevante nel valore del coefficiente, il quale è a pari a 0.632.

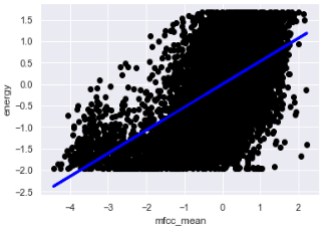

Figura 2.20

Coefficient:	0.5353
MSE:	0.61
MAE:	0.65
R-squared	0.39

Tabella 2.15

Allo stesso tempo, come possiamo vedere dalla Tabella riepilogativa 11.2, effettuando la stessa regressione con la regolarizzazione Lasso si ottiene un errore di poco maggiore rispetto alla regressione lineare senza regolarizzazione ed R^2 di 0.39.

2.7.3 Regressione lineare multipla

Coefficient:	[0.332; 0.429; 0.047; 0.103; 0.215; 0.111]
MSE:	0.33
MAE:	0.44
R-squared	0.67

Tabella 2.16

Si è visto che vi è una relazione positiva tra la variabile dipendente *energy* e la variabile indipendente *mfcc_mean*, ma con un basso R^2.
Implementiamo, quindi, una regressione lineare multipla, così da studiare qual è la relazione tra la variabile dipendente *energy* e più variabili indipendenti. Sono state selezionate le variabili: *mfcc_mean*, *chroma_cqt_mean*, *spectral_contrast_mean*, *spectral_rolloff_mean*, *rmse_mean*, *zcr_mean*, le quali non presentano un'alta correlazione tra loro.

Con la Regressione Lineare Multipla si sono ottenuti dei risultati migliori rispetto alla Regressione Lineare Semplice: tutte le variabili hanno una relazione positiva con la variabile dipendente *energy*, gli errori sono minori e con R^2 di 0.67 vi è una relazione positiva maggiore, il che riesce a spiegare maggiormente la varianza della variabile dipendente.

3 Time Series Analysis

3.1 Data Preparation

Genere	n. tracce	%
International	663	18,34%
Experimental	559	15,46%
Pop	495	13,69%
Hip-Hop	495	13,69%
Folk	464	12,84%
Electronic	417	11,54%
Rock	406	11,23%
Instrumental	116	3,21%

Tabella 3.1

Il dataset estratto per effettuare la Time Series Analysis è formato da 3615 tracce e 313 colonne, le quali rappresentano 20 secondi di ogni traccia, e come variabile è stata scelta "*spectral_centroid*". Il dataset è stato standardizzato. Nella Tabella 3.1 viene riportata la distribuzione dei generi all'interno del dataset creato.

3.2 Similarity e Clustering

3.2.1 Similarity

Sono stati effettuati confronti tra 3 tracce di generi diversi, in particolare la traccia con *id 48444* del genere "Pop", la traccia con *id 12394* del genere "Hip-Hop" e la traccia con *id 23371* del genere "Folk".

Prima di ogni misurazione è stata applicata la trasformazione “Amplitude scaling”.
In tutti i casi è stata calcolata la distanza con l’intera Time Series. È inoltre possibile notare come le time series di canzoni con generi simili, quali “Pop” e “Hip-Hop”, presentino distanze minori rispetto a quelle calcolate tra le stesse e la time series della canzone del genere “Folk” (fa eccezione l’uso della distanza euclidea, che in ogni caso non è la distanza ideale nel rappresentare la distanza tra time series).
Un riassunto delle varie distanze è presente nella Tabella 3.2.

	Euclidea	Manhattan	DTW	Sakow Chiba	Itakura
Pop - Hip-Hop	25	336	10.8	21.94	13.1
Pop - Folk	24	354.7	11.8	21.93	13.7
Hip-Hop - Folk	24.6	348.7	13.2	22.2	15.7

Tabella 3.2

In Figura 3.1 viene mostrato l’**optimal path** trovato rispettivamente con la distanza DTW, applicando il *constraint “Sakoe_Chiba”* ed applicando il *costraint “Itakura”* (il path nei grafici va dal punto in alto a Sx al punto in basso a Dx).

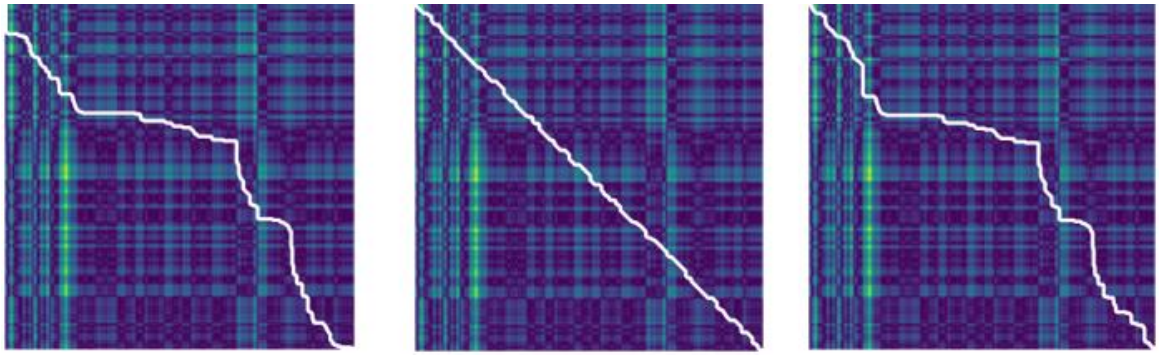

Figura 3.1

3.2.2 Clustering

Per effettuare la *Clustering Analysis* è stato deciso inizialmente di utilizzare l’intero dataset, composto da canzoni di tutti i generi. Tuttavia i risultati ottenuti non sono stati soddisfacenti, e per questo si è deciso di utilizzare un dataset contenente un subset di 100 canzoni ciascuno del genere *Pop*, *Hip-Hop* e *Folk*

Sono state scelte due tipologie di approcci:

- K-means con approccio *shape-based*
- K-means con approccio *apporoximation based*

Per calcolare il valore ottimale di k, l’algoritmo è stato effettuato con un k da 1 a 9, e la metrica utilizzata è stata la DTW (Figura 3.2).

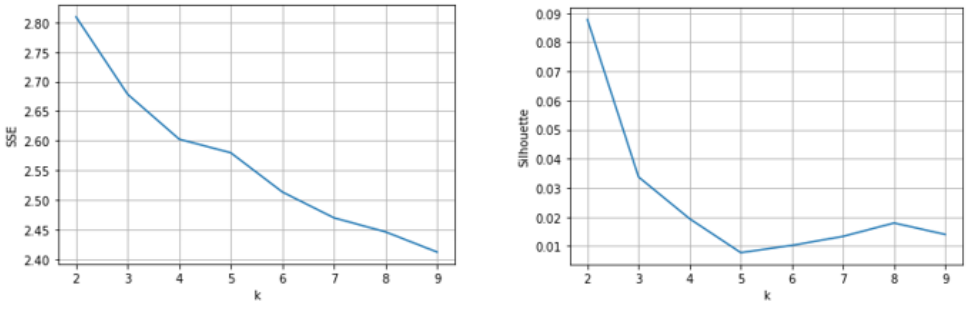

Figura 3.2

3.2.2.1 K-means shape based

Come si osserva dal grafico in Figura 3.3, dove osserviamo le time series dei valori medi di ogni cluster, l'algoritmo ha identificato tre cluster: il primo formato in maggior parte da canzoni del genere *Folk* caratterizzate da un livello dello "spectral_centroid" piu basso, un secondo e un terzo cluster invece formati da canzoni di genere *Pop* ed *Hip-Hop*, caratterizzate in entrambi i casi da un livello più elevato, in media, dello "spectral_centroid".

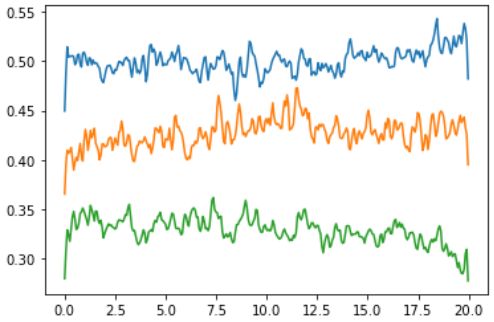

Figura 3.3

Per una valutazione "qualitativa dei cluster" è stato calcolato *SSE* e *Silhouette Score*, quest'ultimo però è stato possibile calcolarlo solo nel caso dell'utilizzo della distanza euclidea, in quanto è stato considerato eccessivamente *computationally expensive* nel caso delle altre metriche.

Di seguito la Tabella 3.3 mostra un riepilogo dei risultati ottenuti con il K-means shape based ottenuto con combinazioni di distanze diverse.

	SSE	Silhouette Score
K = 3, Euclidean	13.15	0.33
K = 3, SoftDTW	275737.5	NA
K = 3, DTW	2.7	NA

Tabella 3.3

3.2.2.2 K-means approximation based

Sono stati utilizzati due algoritmi differenti per l'approssimazione delle time series, il **PAA** (*PiecewiseAggregateApproximation*) e il **SAX** (*SymbolicAggregateApproximation*).

All'interno della Tabella 3.4 osserviamo i risultati ottenuti in termini di SSE e Silhouette (anche in questo caso solo per la distanza euclidea) trovati con le diverse combinazioni di metriche applicando il PAA.

	SSE	Silhouette Score
K = 3, Euclidean	0.12	0.9
K = 3, SoftDTW	196.1	NA
K = 3, DTW	0.08	NA

Tabella 3.4

In Figura 3.4 sono raffigurati i cluster ottenuti con la distanza DTW.

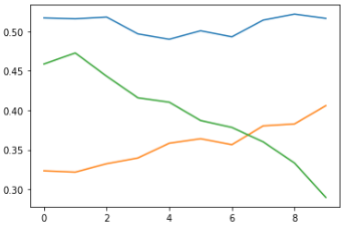

Figura 3.4

In questo caso, a seguito dell'applicazione dell'algoritmo, i centroidi non si trovano nel loro "domain" originale, ma in un domain approssimato formato da 10 time stamps; si è scelto di adottare un numero di segmenti del PAA pari a 8.

Osserviamo come i risultati siano simili all'approccio *shape_based* per uno dei due cluster contenenti tracce "Pop" e "Hip-hop", il quale mantiene un livello elevato dello "spectral centroid". Complessivamente le time series dei cluster ottenuti presentano un livello minore di rumore rispetto a prima dell'applicazione del metodo di approssimazione.

All'interno della Tabella 3.5 osserviamo i risultati ottenuto in termini di SSE e Silhouette (anche in questo caso solo per la distanza euclidea) trovati con le diverse combinazioni di k e metriche trovate applicando il **SAX**.

	SSE	Silhouette Score
K = 3, Euclidean	1.5	0.35
K = 3, SoftDTW	146.9	NA
K = 3, DTW	0.83	NA

Tabella 3.5

In Figura 3.5 sono raffigurati i cluster ottenuto con la distanza DTW.

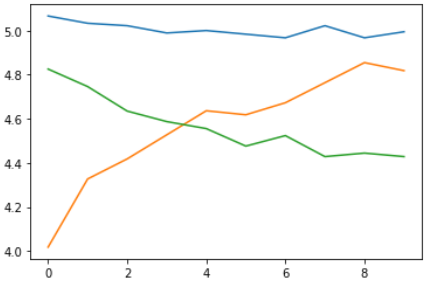

Figura 3.5

Anche in questo caso, a seguito dell'applicazione dell'algoritmo, i centroidi non si trovano nel loro "domain" originale, ma in un domain approssimato formato da 10 time stamps; il numero di segmenti utilizzato nel SAX è pari a 10 mentre il parametro "alphabet_size" scelto è 8. Anche in questo caso osserviamo come dopo l'approssimazione il livello di rumore nei vari cluster sia diminuito notevolmente. L'approccio SAX dà risultati che variano in maniera minore se però confrontati con quelli ottenuti con il metodo di approssimazione PAA: anche in questo caso, infatti, il cluster che contiene time series con livello medio maggiore di "spectral centroid" è molto simile a quello ottenuto in precedenza. Possiamo inoltre affermare che, nelle prove effettuate, utilizzando diverse combinazioni di k (numero di cluster), applicare algoritmi di approssimazione diversi ha portato a risultati anche molto diversi tra loro.

3.3 Motif e Anomaly Discovery

3.3.1 Matrix Profile e Motif Discovery

Si è deciso di svolgere l'analisi dei motif su una sola traccia. Sono state prese in considerazione le tre canzoni più popolari del dataset (ossia le tre con i valori più alti dell'attributo "favorites"), e dopo una prima analisi la traccia *Night Owl* dei *Broke for Free* ha mostrato i risultati più interessanti. La canzone in questione (*track_id* 42377) è la più popolare del dataset, è del 2011 ed appartiene al genere "electronic".

Poiché lo scopo dell'analisi è trovare pattern che si ripetano all'interno della traccia, la prima necessità è stata quella di trovare la lunghezza *m* ideale delle *subpart* da confrontare. Infatti, la ricerca dei motif viene effettuata tramite un approccio *brute-force*, in cui sono confrontate tutte le possibili sequenze di lunghezza *m*.

Per la scelta del parametro ideale sono stati realizzati diversi *matrix profile* con parametri di *m* inclusi tra 2 e 100. Nella Figura 3.6 è riportato un campione dei risultati.

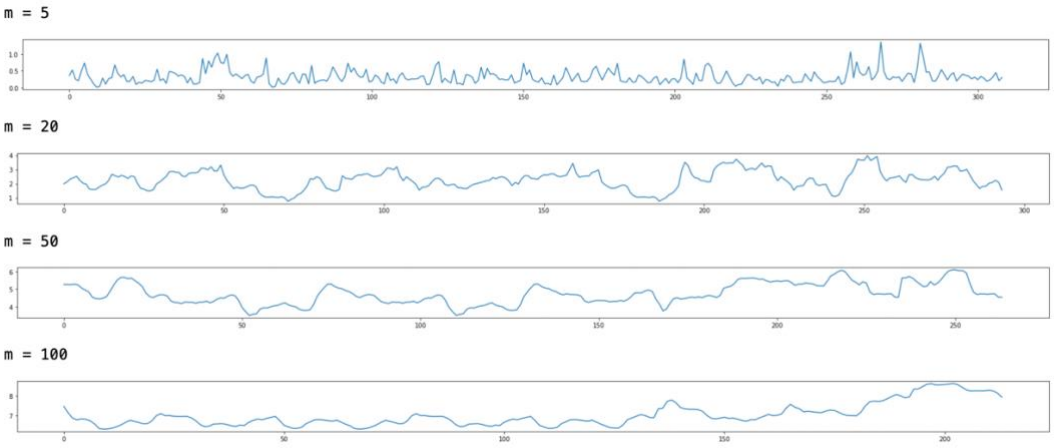

Figura 3.6

Alla luce dei risultati emersi è stato selezionato il parametro 20, in quanto appare come il più preciso nell'indicare la posizione dei motif.

Quindi, una volta stabilita la lunghezza dei motif da ricercare, è stata condotta la *motif discovery*. Anche in questo caso è stato necessario svolgere il tuning di alcuni parametri:

- *max_motifs*: il numero massimo di motif da riportare. Non è stato indicato.
- *radius*: il raggio da un motif entro cui cercare nuovi motif. Sono stati provati valori compresi tra 0,5 e 1,5.
- *n_neighbors*: il numero di vicini da trovare per ogni motif. È stato lasciato il valore di default, ossia "tutti".
- *ex_zone*: la zona di esclusione, ossia la distanza minima che deve intercorrere tra due motif. Sono stati provati valori compresi tra 0 e 10.
-

Nella Figura 3.7 sono riportati alcuni esempi per visualizzare come cambiano i risultati in funzione di questi parametri.

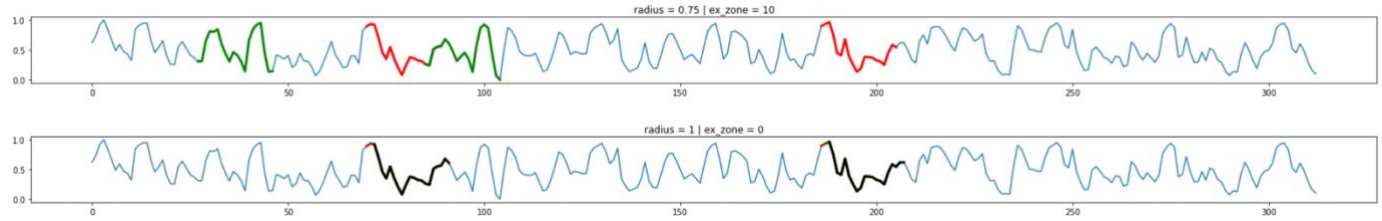

Figura 3.7

I risultati migliori sono stati ottenuti con *radius* = 2 ed *ex_zone* = 10. Nella Figura 3.8 sono evidenziati i motif trovati dall'algoritmo.

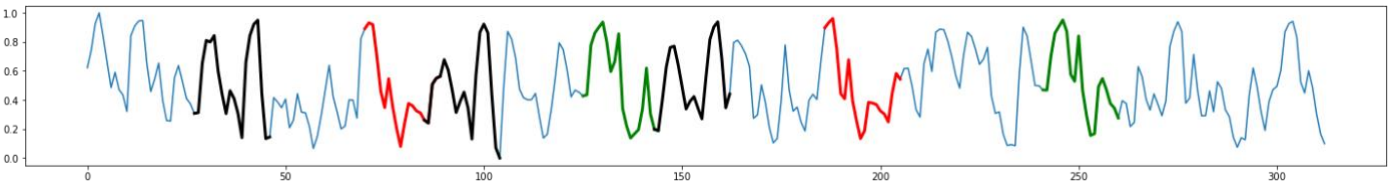

Figura 3.8

È possibile osservare la corrispondenza tra le posizioni dei motif trovati (Figura 3.8) e le "valli" del matrix profile (Figura 3.6). Il motif in rosso (*timestamp* 70 e 186) è quello indicato in maniera più netta.
Nella Figura 3.9 sono stati isolati i motif.

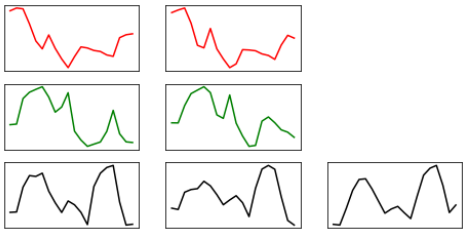

Figura 3.9

3.3.2 Anomaly Discovery

Successivamente è stata svolta l'*anomaly discovery*, per quanto sia difficile stabilire chiaramente cosa sia un'anomalia all'interno di una canzone, e per quanto sia difficile che la suddetta anomalia si verifichi nei primi 20 secondi della traccia.
È stato necessario svolgere il tuning di due parametri:

- *k*: il numero di anomalie da trovare. Sono stati testati valori da 1 a 10.
- *ex_zone*: la distanza minima che deve intercorrere tra due anomalie. Anche qui sono stati provati valori da 1 a 10.

A seconda della combinazione tra parametri sono state riscontrate dalle 0 alle 3 anomalie (nella Figura 3.10 viene riportato un esempio con i parametri ex_zone = 6 e k = 4).

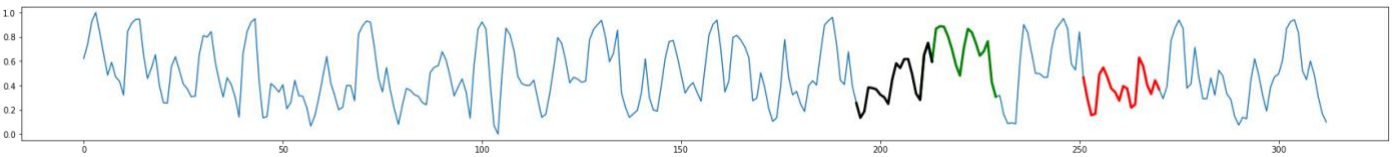

Figura 3.10

I risultati non sono particolarmente convincenti, poiché delle 3 anomalie trovate solo una (riportata in verde) non si sovrappone a parte dei motif estratti in precedenza.
In conclusione, dato anche quanto detto a inizio paragrafo, si può affermare che non ci siano anomalie nella traccia analizzata.

3.4 Classification

Basandoci sulla Time Series si è deciso di classificare il genere musicale: in particolare l'obiettivo della classificazione è riconoscere le canzoni di genere **Hip-Hop** rispetto alle canzoni di altri generi.
Il dataset utilizzato per la classificazione è composto da canzoni appartenenti ai generi:

- Pop, Folk, Electronic, Rock, Instrumental: tutti questi generi compongono la classe 0
- Hip-Hop: classe 1.

Come si può osservare dalla Tabella 3.6 il dataset è sbilanciato.

Genere	n. tracce	%
Hip-Hop (class 1)	495	20,70%
Other genre (class 0)	1898	79,30%

Tabella 3.6

La classificazione è stata implementata con due modalità diverse:

- Una classificazione ha sfruttato l'intera Time Series data da 313 momenti, rappresentati 20 secondi di ogni canzone;
- Una seconda classificazione basata su **shapelets**.

Il data set è stato diviso in 70% training set e 30% test set. Per tutti i classificatori presenti nella sezione 3.4.2 è stato utilizzato il *K-Fold Cross-Validation* implementato con *Repeated Stratified KFold* con k = 5.

3.4.1 Shapelets discovery

Sono state individuate 5 shapelets di lunghezza 31 per la classificazione. Le shapelets sono rappresentate in Figura 3.11 e, come possiamo vedere, si differenziano sia in termini di forma che di valore medio: due shapelets si trovano agli estremi e presentano una forma piatta, mentre le tre shapelets centrali hanno un valore medio compreso tra circa 0.35 e 0.55 con una forma più ondulata.

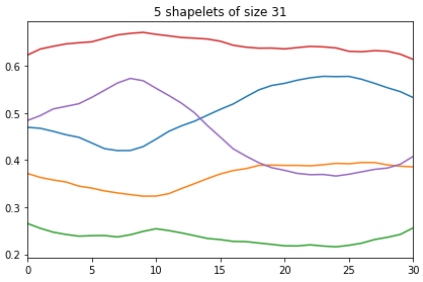

Figura 3.11

Nella Figura 3.12 mettiamo a confronto il posizionamento delle shapelets nelle canzoni di diverso genere musicale.

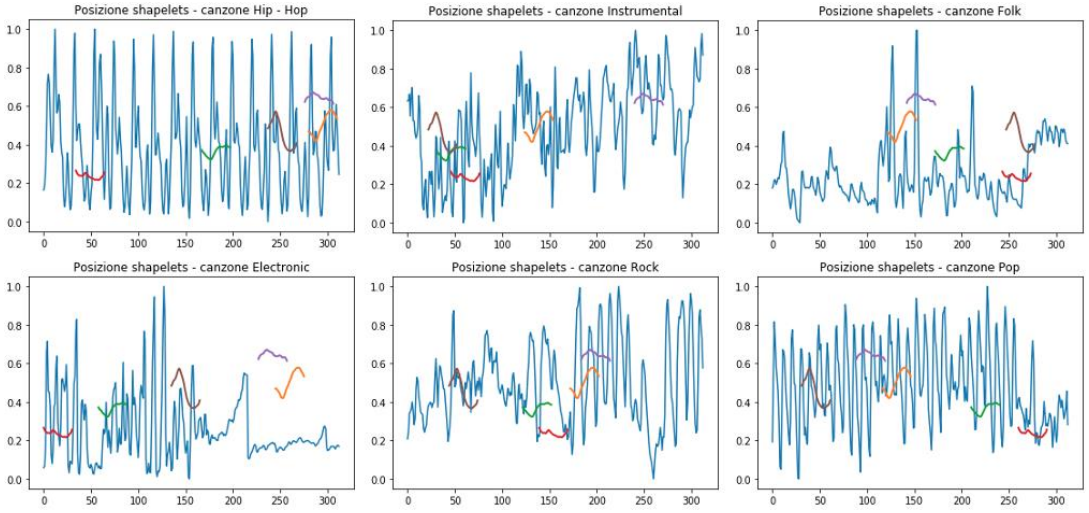

Figura 3.12

Osservando il posizionamento delle shaplets, vediamo come le 3 shapelets che presentano una forma più ondulata siano quelle che si posizionano in modo migliore nella Time Series della canzone Hip-Hop. Tuttavia, le shapelets in questione si posizionano bene anche sulla Time Series della traccia *Pop* presa ad esempio: questo indica che si potrà verificare che delle tracce verranno classificate erroneamente.

3.4.2 Classificatori

Per la scelta dei parametri di tutti i classificatori è stato effettuando il tuning selezionando la combinazione di parametri con il valore di *Mean Validation Score* più elevato.

CLASSIFICATORI SHAPELETS-BASED

K Nearest-Neighbors

parameters: algorithm = brute; metric = **manhattan**; n_neighbors = 14; weights = uniform			
Mean validation score	0.816		
Std score	0.020		
Metrics on training set			
Accuracy 1.0			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	1.00	1.00	1.00
Hip-Hop = 1	1.00	1.00	1.00
Metrics on test set			
Accuracy 0.7981			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.83	0.93	0.88
Hip-Hop = 1	0.53	0.28	0.37
Confusion matrix			
TN = 531	FP = 38		
FN = 107	TP = 42		

parameters: algorithm = brute; metric = **dtw_itakura**; n_neighbors = 15; weights = uniform			
Mean validation score	0.815		
Std score	0.014		
Metrics on training set			
Accuracy 0.8299			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.85	0.95	0.90
Hip-Hop = 1	0.65	0.38	0.48
Metrics on test set			
Accuracy 0.7954			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.83	0.93	0.88
Hip-Hop = 1	0.51	0.28	0.36
Confusion matrix			
TN = 529	FP = 40		
FN = 107	TP = 42		

Decision Tree Classifier

parameters: min_sample_split = 50; min_sample_leaf = 70; max_depth = 8; criterion = Gini			
Mean validation score	0.811		
Std score	0.019		
Metrics on training set			
Accuracy 0.8215			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.84	0.96	0.89
Hip-Hop = 1	0.65	0.30	0.41
Metrics on test set			
Accuracy 0.8064			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.83	0.95	0.89
Hip-Hop = 1	0.57	0.26	0.36
Confusion matrix			
TN = 540	FP = 29		
FN = 110	TP = 39		

Random Forest Classifier

parameters: n_estimators = 160; min_samples_split = 20; min_samples_leaf = 10; max_depth = 3; criterion = Gini			
Mean validation score	0.814		
Std score	0.019		
Metrics on training set			
Accuracy 0.8263			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.84	0.96	0.90
Hip-Hop = 1	0.68	0.30	0.42
Metrics on test set			
Accuracy 0.8092			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.83	0.95	0.89
Hip-Hop = 1	0.59	0.28	0.37
Confusion matrix			
TN = 540	FP = 29		
FN = 108	TP = 41		

Support Vector Machine

parameters: kernel = rbf; gamma = 1; C = 1000			
Mean validation score	0.823		
Std score	0.017		
Metrics on training set			
Accuracy 0.8328			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.84	0.98	0.90
Hip-Hop = 1	0.77	0.27	0.40
Metrics on test set			
Accuracy 0.822			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.82	0.96	0.88
Hip-Hop = 1	0.57	0.20	0.30
Confusion matrix			
TN = 546	FP = 23		
FN = 119	TP = 30		

Logistic Regression

parameters: C = 11.29; solver: liblinear; penalty = l2			
Mean validation score	0.823		
Std score	0.018		
Metrics on training set			
Accuracy 0.8245			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.83	0.97	0.90
Hip-Hop = 1	0.70	0.26	0.38
Metrics on test set			
Accuracy 0.8092			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.82	0.95	0.88
Hip-Hop = 1	0.54	0.20	0.29
Confusion matrix			
TN = 543	FP = 26		
FN = 119	TP = 30		

CLASSIFICATORI SULL'INTERA TIME SERIES

K Nearest-Neighbors

parameters: algorithm = ball_tree; metric = **manhattan**; n_neighbors = 6; weights = uniform			
Mean validation score	0.810		
Std score	0.012		
Metrics on training set			
Accuracy 1.0			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	1.00	1.00	1.00
Hip-Hop = 1	1.00	1.00	1.00
Metrics on test set			
Accuracy 0.7981			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.80	0.99	0.89
Hip-Hop = 1	0.75	0.06	0-11
Confusion matrix			
TN = 566	FP = 3		
FN = 140	TP = 9		

parameters: algorithm = brute; metric = **dtw_itakura**; n_neighbors = 5; weights = uniform			
Mean validation score	0.809		
Std score	0.011		
Metrics on training set			
Accuracy 0.8552			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.86	0.97	0.91
Hip-Hop = 1	0.78	0.42	0.48
Metrics on test set			
Accuracy 0.7758			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.82	0.92	0.87
Hip-Hop = 1	0.42	0.21	0.28
Confusion matrix			
TN = 525	FP = 44		
FN = 117	TP = 32		

Decision Tree Classifier

parameters: min_sample_split = 90; min_sample_leaf = 90; max_depth = 23; criterion = Gini			
Mean validation score	0.793		
Std score	0.002		
Metrics on training set			
Accuracy 0.7934			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.79	1.00	0.88
Hip-Hop = 1	0.00	0.00	0.00
Metrics on test set			
Accuracy 0.7925			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.79	1.00	0.88
Hip-Hop = 1	0.00	0.00	0.00
Confusion matrix			
TN = 569	FP = 0		
FN = 149	TP = 0		

Random Forest Classifier

parameters: n_estimators = 140; min_samples_split = 20; min_samples_leaf = 70; max_depth = 20; criterion = Gini			
Mean validation score	0.793		
Std score	0.001		
Metrics on training set			
Accuracy 0.7934			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.79	1.00	0.88
Hip-Hop = 1	0.00	0.00	0.00
Metrics on test set			
Accuracy 0.7925			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.79	1.00	0.88
Hip-Hop = 1	0.00	0.00	0.00
Confusion matrix			
TN = 569	FP = 0		
FN = 149	TP = 0		

Support Vector Machine

parameters: kernel = rbf; gamma = 0.001; C = 1			
Mean validation score	0.793		
Std score	0.001		
Metrics on training set			
Accuracy 0.7934			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.79	1.00	0.88
Hip-Hop = 1	0.00	0.00	0.00
Metrics on test set			
Accuracy 0.7925			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.79	1.00	0.88
Hip-Hop = 1	0.00	0.00	0.00
Confusion matrix			
TN = 569	FP = 0		
FN = 149	TP = 0		

Logistic Regression

parameters: C = 0.616; solver: liblinear; penalty = l1			
Mean validation score	0.793		
Std score	0.010		
Metrics on training set			
Accuracy 0.8042			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.80	1.00	0.89
Hip-Hop = 1	0.85	0.06	0.12
Metrics on test set			
Accuracy 0.8092			
Classes	**Precision**	**Recall**	**F1-score**
Other genre = 0	0.80	0.99	0.88
Hip-Hop = 1	0.40	0.03	0.05
Confusion matrix			
TN = 563	FP = 6		
FN = 145	TP = 4		

Tabella 3.7

Osservando la Tabella 3.7 riepilogativa dei risultati ottenuti con i classificatori notiamo una marcata differenza tra le prestazioni dei classificatori creati sull'intera Time Series ed i classificatori basati sulle Shapelets, il che evidenzia che i risultati migliori si ottengono implementando i classificatori basandosi sulle shapelets.
Le performance dei vari classificatori shapelets distance-based sono simili tra loro, mentre un classificatore con cui si è raggiunto un risultato di poco migliore è il KNN con la distanza DTW Itakura.

4 Sequential Patterns and Advanced Clustering

4.1 Sequential Pattern Mining

Per il Sequential Pattern Mining sono state utilizzate 3 time series che descrivono l'andamento della variabile "spectral centroid" nei primi 20 secondi della traccia. Sono state selezionate le 3 canzoni più popolari del dataset, ossia *Night Owl* (track_id 42377), *Enthusiast* (track_id 69170), *Kopeika* (track_id 55718) e *Lullaby* (track_id 28553). Come per il paragrafo 3.3.1, la popolarità delle canzoni è stata valutata sulla base dell'attributo "favorites".
Il dataset in questione è stato standardizzato e successivamente trasformato con il metodo SAX, per il quale sono stati utilizzati i parametri *n_paa_segments* = 15 e *n_sax_symbols* = 8. Nella Figura 4.1 sono riportate le tracce dopo la trasformazione.

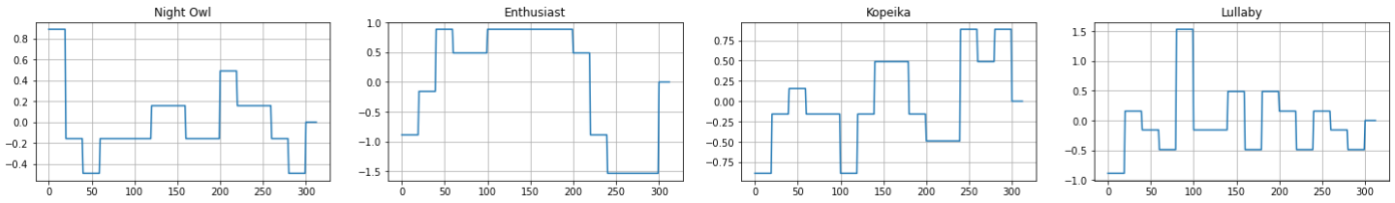

Figura 4.1

Si è poi proceduto a mappare i valori registrati nelle tracce come riportato dalla Tabella 4.1.

Simbolo	Valore
A	-1.5341205443525463
B	-0.887146559018876
C	-0.1573106846101707
D	0.0
E	0.4887764111146695
F	0.887146559018876

Tabella 4.1

Successivamente, allo scopo di realizzare un dataset che si prestasse all'analisi del Sequential Pattern Mining, e che quindi fosse composto da transazioni multiple, sono state create delle transazioni in cui gli item corrispondono al valore di ognuna delle 4 tracce per ogni momento della canzone, per un totale di 313 transazioni di 4 item.
Nella Tabella 4.2 sono riportati i 15 pattern più frequenti con il loro relativo supporto. Inoltre, si sono voluti isolare i pattern più frequenti composti da 3 e 4 elementi.

Pattern	Supporto
[B]	220
[C]	173
[F]	160
[A]	160
[B, C]	120
[E]	120
[F, C]	100
[D]	93
[B, E]	80
[B, F]	80
[B, A]	60
[B, B]	60
[B, F, C]	60
[E, C]	60
[F, F]	60

Pattern	Supporto
[B, F, C]	60
[A, F, C]	40
[B, A, C]	40
[B, E, C]	40
[B, F, F]	40
[A, A, G, B]	20
[A, F, E, C]	20
[B, A, F, C]	20
[B, C, C, E]	20
[B, E, C, B]	20
[B, E, C, G]	20
[B, F, A, C]	20
[B, F, C, F]	20
[B, F, F, B]	20
[D, A, G, E]	20

Tabella 4.2

4.2 Advanced Clustering

4.2.1 X means

Il dataset utilizzato per effettuare il X-Means Clustering è lo stesso dataset utilizzato nei moduli 1 e 2. Al dataset in questione è stata rimossa la colonna "Electronic Labeler" così da avere solamente variabili numeriche continue e ne è stata ridotta la dimensione a due variabili con il PCA, in modo tale da effettuare il Clustering su un dataset bidimensionale.

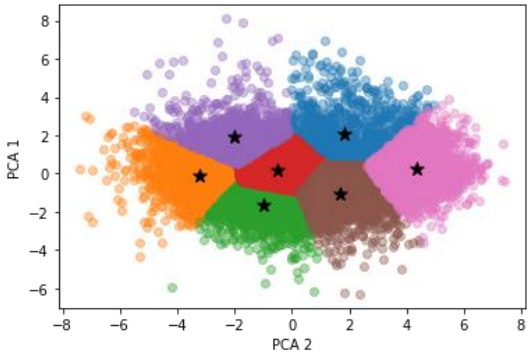

Figura 4.2

SSE	0.4655
Silhouette Score	1870

Tabella 4.3

A differenza del K-Means, l'algoritmo X-Means non necessita che venga specificato il numero di cluster.
Il risultato dell'analisi è presentato nella Figura 4.2, dove vengono rappresentati i 7 cluster trovati dal X-Means all'interno del dataset. Notiamo come i centroidi si trovino tutti in un range [-2, 2] e come non vi siano molti punti sparsi nel grafico. Il nostro clustering con K = 7 ha ottenuto un valore SSE pari a 1870 ed una *Silhouette Score* pari a 0.4655: ciò vuol dire che i cluster non sono molto coesi. Questo valore relativamente basso dipende molto dalle caratteristiche dei dati e dalla natura delle osservazioni che vi sono all'interno: in particolare, il fatto che il dataset contenga tracce che appartengono a 16 generi musicali differenti fa sì che le osservazioni presenti nel medesimo cluster abbiano valori diversi tra loro.

4.3 Transactional Clustering

Per il task del Transactional Clustering è stato deciso di utilizzare lo stesso dataset usato in precedenza per la Rule based classification, composto soltanto da variabili categoriche, dal quale, per l'analisi, sono state eliminate le colonne "*nome_cantante"* e "*Language*". È stato scelto il metodo del **K-Modes**, il metodo di inizializzazione "Huang" ed un numero di inizializzazioni pari a 5. Per la scelta del k ottimale è stato deciso di non utilizzare l'SSE in quanto considerato, adottando la distanza euclidea, non adatto in un dataset interamente con variabili di tipo categorico, è stato deciso di utilizzare una funzione di costo che calcola la distanza tra attributi di tipo categorico come il numero degli errori di corrispondenza. In Figura 4.3 osserviamo dal grafico come il valore ottimale di k sia 4 con un costo pari a 15347.

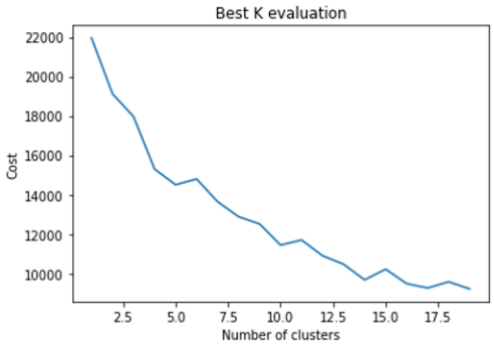

Figura 4.3

Di seguito osserviamo le mode (oggetti rappresentativi) di ogni cluster ottenute dall' algoritmo.

- Cluster0: 'Medium popularity', 'Winter', 'Short duration', 'Rock'
- Cluster1: 'Medium popularity', 'Spring', 'Medium duration', 'Folk'
- Cluster2: 'Low popularity', 'Autumn', 'Medium duration', 'Rock'
- Cluster3: 'High popularity', 'Spring', 'Short duration', 'Electronic'

In Figura 4.4 sono stati quindi combinati i cluster ottenuti con il dataset iniziale, osserviamo come gli item in ognuno degli oggetti rappresentativi dei diversi cluster abbia presenza maggiore nelle colonne dei cluster dei quali il K-modes li ritiene mode rappresentative.

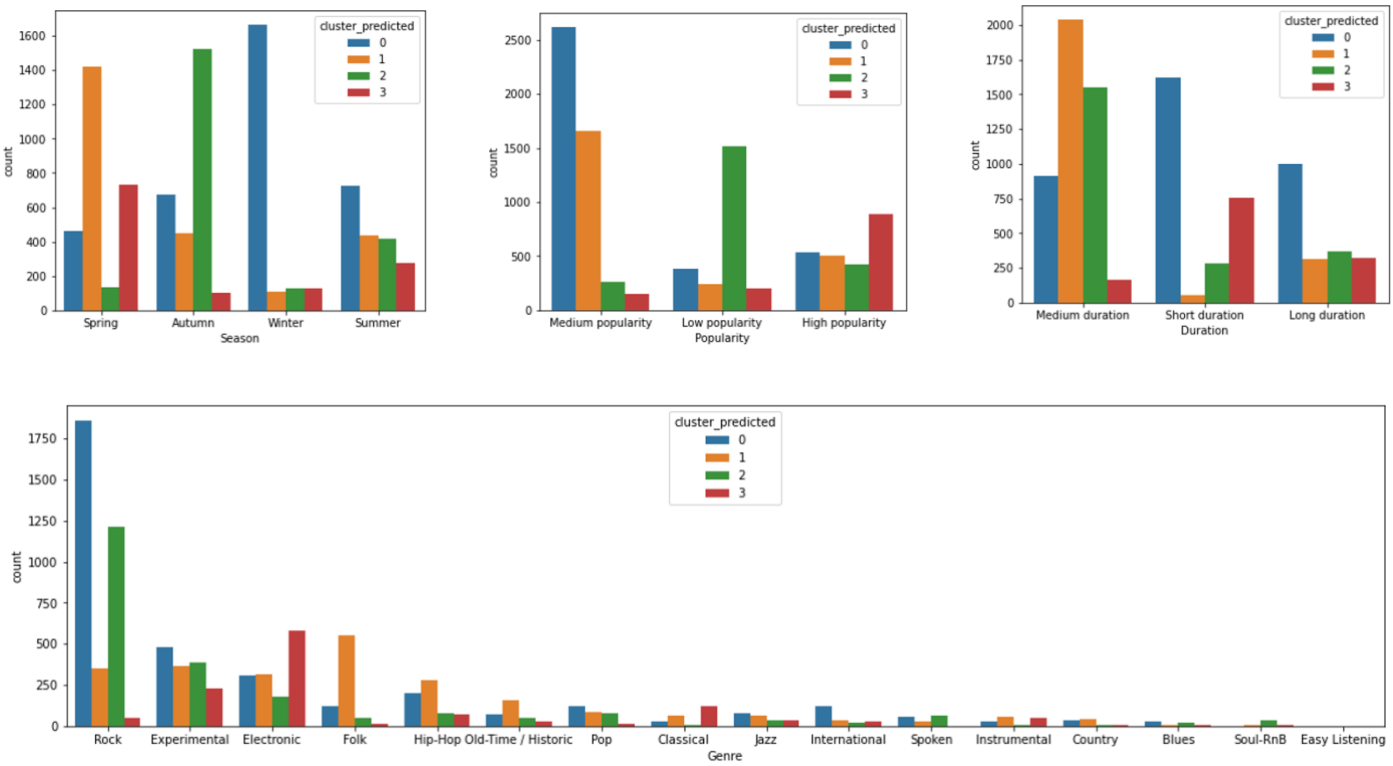

Figura 4.4

5 Explainable AI

Per affrontare il tema dell'*explainability* sono stati selezionati i 3 ensemble classifiers utilizzati nel paragrafo 2.5, ossia Random Forest, Bagging e Boosting. Questi classificatori, infatti, appartengono alla famiglia dei *Black Box Models*, espressione con cui si identificano quei modelli per cui è possibile osservare ed interpretare l'input e l'output, ma non i meccanismi all'interno della "scatola".
Piuttosto che soffermarsi sulla *local explanation*, ossia su una spiegazione circoscritta ad una singola osservazione, si è preferito dare una *global explanation*, ovverosia una spiegazione della logica generale del modello. Il *global explainer* utilizzato è TREPAN.

Applicando TREPAN al **Random Forest**, quest'ultimo implementato con gli stessi parametri utilizzati durante la classificazione, si è potuta ottenere una rappresentazione grafica (sotto forma di Decision Tree) della logica all'interno della *Black Box*. Il risultato è presentato nella Figura 5.1.

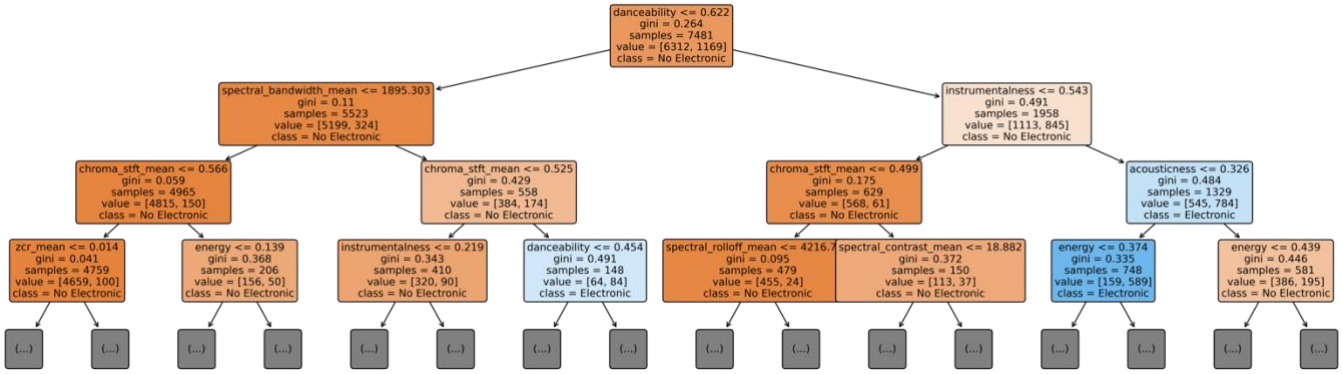

Figura 5.1

La *fidelity*, ossia la capacità dell'*explainer* di spiegare le decisioni del classificatore, si è attestata al 91%. Qualora invece si volesse utilizzare l'*explainer* come classificatore, si avrebbe un'accuracy del 78% (contro l'82% del modello effettivamente utilizzato).

TREPAN è stato successivamente utilizzato per interpretare i 4 *base classifier* utilizzati nell'implementazione del **Bagging**. Per tutti e 4 i classificatori sono stati utilizzati i parametri descritti dal paragrafo 2.5.2. Nella Tabella 5.1 sono riportati i risultati ottenuti in termini di *fidelity* e di *accuracy* (intesa come accuratezza nel caso in cui il modello venisse usato per classificare la classe target).

Base classifier	**Fidelity**	**Accuracy**
Decision Tree	0,81	0,75
SVM	0,75	0,74
Random Forest	0,83	0,77
KNN	0,79	0,75

Tabella 5.1

Nella Figura 5.2 è rappresentato il Decision Tree che spiega come è avvenuta la classificazione per SVM (Bagging).

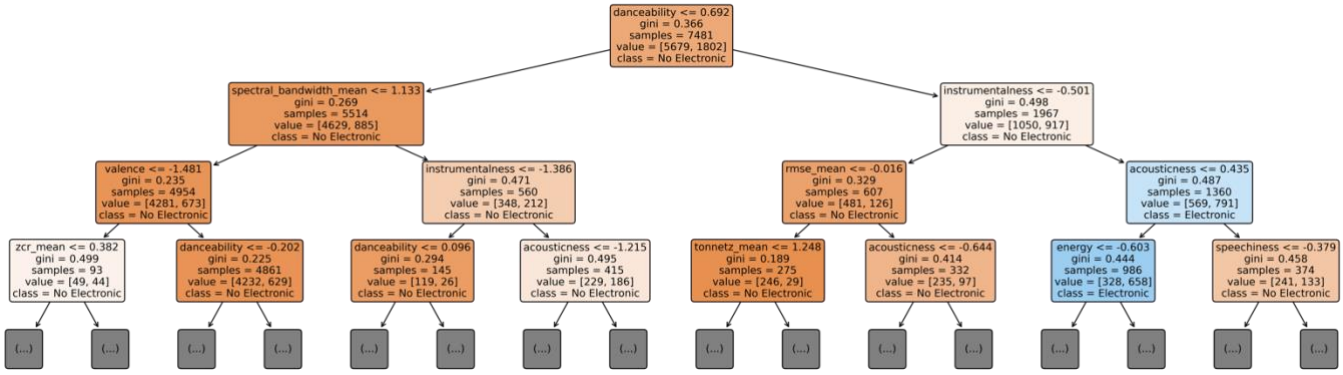

Figura 5.2

Infine, si è applicato lo stesso procedimento per il **Boosting**. Nella Tabella 5.2 sono riportati i risultati ottenuti.

Base classifier	**Fidelity**	**Accuracy**
Decision Tree	0,95	0,78
Random Forest	0,81	0,74

Tabella 5.2