Università di Pisa

Data Mining
Advanced Topics and Applications Project

FMA Dataset

Uno studio musicale

A cura di
Erica Cau, Simona Mazzarino, Federico Mazzoni

Indice

1 Introduzione

In un mondo in cui musica di più generi è ormai accessibile facilmente a tutti, con servizi di *streaming* come *Spotify*, *Apple Music* o *Amazon Music*, i gusti del pubblico si sono fatti sempre più sofisticati. Può risultare così difficile per un'azienda valutare con quale traccia musicale accompagnare la pubblicità di un proprio prodotto. **Quali possibili canzoni oggi incontrano le preferenze del pubblico?** Il presente progetto tenta di rispondere a questi bisogni delle aziende, analizzando nella prima parte ***le caratteristiche che rendono un album musicale popolare*** e focalizzandosi nella seconda parte sulle singole tracce musicali di due generi radicalmente diversi fra loro (il Rock e il Folk).
Per rispondere a questi interrogativi, si è attinto al ***Free Music Archive*** (FMA), una raccolta di dati musicali liberamente accessibile e scaricabile, composta da 106,574 tracce appartenenti a 161 generi musicali differenti (nato, del resto, proprio per questo tipo di ricerche[1]).

2 Data Understanding e Data Preparation

2.1 Data Transformation

Per i nostri scopi è stato utilizzato un subset del dataset FMA, composto da 25,000 tracce appartenenti a 16 generi musicali. A seguito della pulizia del dataset, si è poi proceduto all'applicazione di vari metodi di classificazione e regressione.
Il dataset originale era suddiviso in quattro macrocategorie: *tracks*, *genres*, *features* e *echonest*. Per la classificazione è stato utilizzato il subset *tracks* raggruppando i vari record per album, riducendone il numero a 5055, e selezionando gli attributi più rilevanti, ovvero ***album_favorites***, ***track_duration***, ***track_listens***, ***artist_favorites***, ***genre***, ***album_number_tracks***, ***album_type***, ***album_favorites***, ***album_listens***, ***track_bit_rate***.
Il dataset includeva anche caratteristiche delle singole tracce calcolate in base ad alcune misurazioni specifiche dell'azienda *Echonest*, acquistata da Spotify nel 2014. A causa della grande presenza di *missing values* e del fatto che le nostre analisi si sono concentrate in questa fase sugli album piuttosto che sulle singole tracce, nel presente progetto non si farà uso di questi attributi.
Alcune *feature* selezionate erano riferite alla singola traccia, per cui, nella fase di raggruppamento dei record per album, è stato necessario calcolare dei nuovi valori che si riferissero all'album: per *track_bit_rate* e *track_listens* si è calcolata la media, per *track_duration* si è calcolata la somma della durata di ciascuna traccia e, infine, per *artist_favorites* si è considerato il valore massimo. Successivamente, l'attributo *track_bit_rate* è stato trasformato in *quality*, assegnando dei valori da 0 a 4, seguendo gli standard dell'industria.[2] L'attributo *album_type* classificava gli album in quattro cate-

[1] Pur non essendo esente da difetti, come lo sbilanciamento nei confronti del *Rock*, come già evidenziato nel *paper* originale: https://arxiv.org/pdf/1612.01840.pdf

[2] $x < 128000$: 0, $128000 \leq x < 192000$: 1; $192000 \leq x < 256000$: 2; $256000 \leq x < 320000$: 3; $320000 < x$: 4.

gorie (Single, Album, Radio Program, Live Concert)[3] e presentava 121 missing values. Per risolvere il problema è stato consultato il sito del produttore musicale *cdbaby*[4], che riporta i criteri utilizzati da Spotify per classificare un album: Single: l'album ha massimo 3 tracce e una durata inferiore ai 30 minuti; EP: l'album ha dalle 4 alle 6 tracce e una durata inferiore ai 30 minuti; Album: l'album ha almeno 7 tracce e/o una durata superiore ai 30 minuti. È stato quindi introdotto il nuovo valore EP e si è effettuata la riclassificazione degli album, che avevano originariamente valore Single, Album o Nan, secondo questi nuovi criteri.
In seguito, è stato applicato lo *one hot enconding* gli attributi *album_type* e *genre* ottenendo 21 nuove feature. Infine, è stata definita la variabile target è stata definita ***popularity***, calcolata in base ai valori di *album_favorites*: i 235 album con *album_favorites* > 5 sono stati classificati come *popolari*, assumendo valore 1, mentre i restanti 4820 sono stati etichettati come *non popolari* e quindi con valore 0.

2.2 Outlier Detection

Per l'***outlier detection*** si sono in primo luogo applicati i *boxplot*. I risultati più rilevanti sono stati individuati in *track_listens* e *album_listens*, rispettivamente con 567 e 529 *outlier* (con un livello di contaminazione di 0.11 e 0.10). Le nostre successive analisi si sono quindi soffermate su queste due feature.

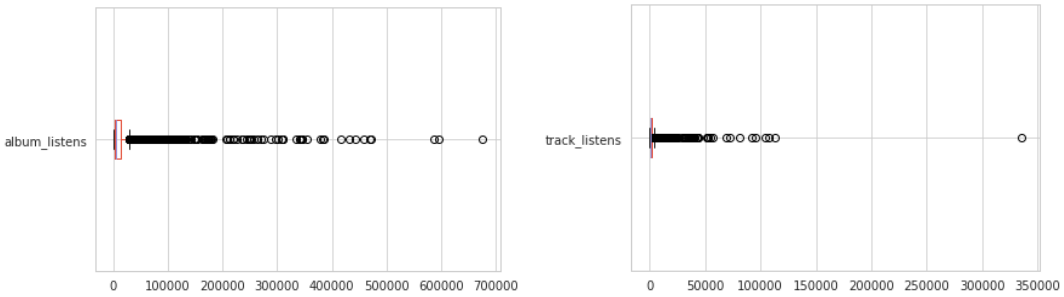


Figura 1: Boxplot delle feature *album_listens* e *track_listens*

2.2.1 Grubbs' Method

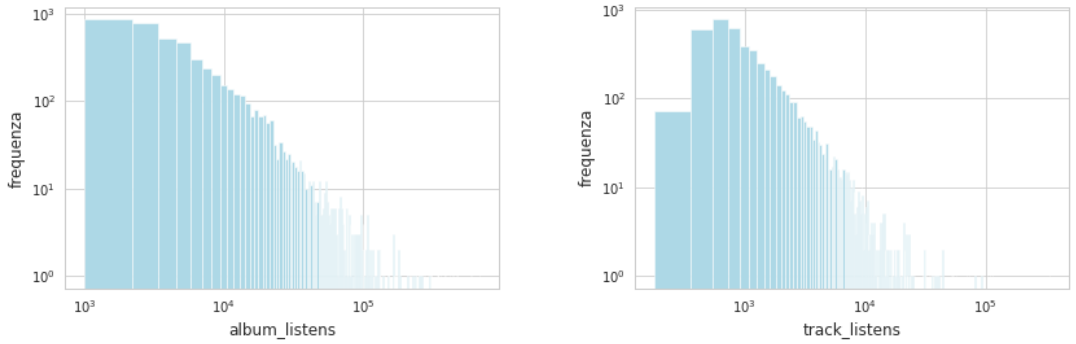
Il metodo di **Grubbs** ha largamente sovrastimato la presenza di *outlier* individuandone 2412 per l'attributo *track_listens* e 1423 per *album_listens*. Questo risultato può essere spiegato dalla distribuzione dei dati delle due feature: entrambe, infatti, seguono una *power law* come si vede nelle figure 2a e 2b, mentre il metodo di Grubbs è pensato per essere applicato a distribuzioni normali.

2.2.2 Local Outlier Factor

Analizzando i due attributi *album_listens* e *track_listens* con il metodo density-based **Local Outlier Factor** (LOF) sono stati ottenuti risultati migliori rilevando 506 *outlier* di cui 37 *popular* e 469 *not popular*. Lo score assegnato dal LOF agli *outlier* oscilla

[3]Oltre a queste quattro categorie era presente anche la categoria *Contest*, che però non possedeva alcun record.

[4]Sito del produttore musicale cdbaby: `https://support.cdbaby.com/hc/en-us/articles/360008275672-What-is-the-difference-between-Single-EP-and-Albums-`



(a) **(b)**

Figura 2: Distribuzione delle features *album_listens* e *track_listens*

da -1.1743021992592586 a -7.451876319522422 e come si può notare in figura 3 gli *outlier* con *score* più alto si concentrano nella sezione del grafico con *track_listens* e *album_listens* con valori massimi rispettivamente di 50000 e 200000. Questo spiega la scarsa presenza di album popolari (che possiamo aspettarci avere più alti ascolti) fra gli *outlier* rilevati. Di particolare interesse la sezione del grafico con valori di *album_listens* compresi fra 200000 e 400000: essa è ad alta densità pur presentando pochi dati, che quindi non sono classificati come *outlier* dal LOF (a differenza, come vedremo, degli altri metodi).

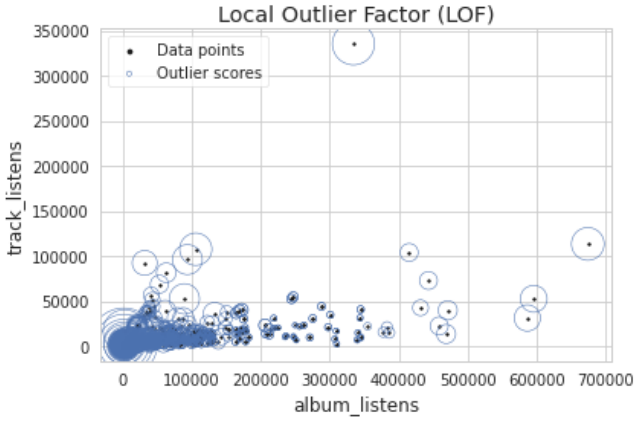


Figura 3: Local Outlier Factor

2.2.3 Angle-Based Outlier Detection

Il metodo **Angle-Based Outlier Detection** (ABOD) ha rilevato 496 *outlier*, con un punteggio minimo di -1.83^{-24} e uno massimo di -5.66^{-13}. L'ABOD è stato lanciato utilizzando i medesimi iperparametri utilizzati per il LOF (numero di vicini pari a 20, livello di contaminazione - risultato ottenuto dai boxplot - settato a 0.1), ma i risultati sono stati tuttavia molto differenti: dei 496 *outlier* rilevati, solo 160 sono in comune con il LOF. Al netto di una maggioranza di album non popolari classificati come *outlier* (363), si è comunque notata una maggior presenza di *outlier popular* rispetto al LOF (133 contro 37).

2.2.4 K-Nearest Neighbors

Il metodo ABOD è generalmente utilizzato per dataset multidimensionali: la bontà dei risultati ottenuti con sole due dimensioni (*track_listens* e *album_listens*) è stata confermata dall'applicazione del più classico **K-Nearest Neighbors**: maggiore la distanza dal proprio KNN, maggiore il punteggio assegnato a ciascun punto. Sempre impostando un numero di vicini pari a 20, il KNN ha rilevato 497 *outlier* (con un valore minimo di 2256.67 ed uno massimo di 380690.20) di cui 450 in comune con l'ABOD.

2.2.5 Isolation Forest

L'ultimo metodo di *outlier detection* sperimentato è stato l'**Isolation Forest**, sia nella sua versione standard, sia nella sua versione *Extended*. Le due applicazioni dell'algoritmo, lanciate impostando 200 alberi e un *sample_size* di 512, hanno portato risultati molto simili: fra i 500 punti con *outlier score* più alto si sono notati 460 in comune fra le due foreste. Come si può notare in figura 4 le differenze riguardano i punti localizzati fra le zone di confine. In generale i risultati hanno confermato quelli ottenuti da ABOD e KNN.

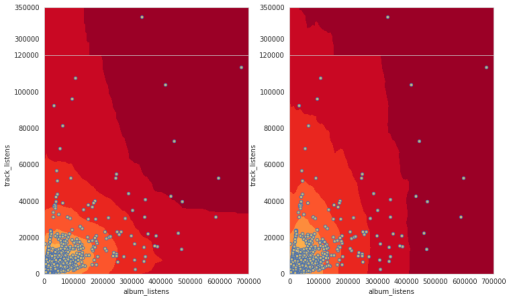


Figura 4: Isolation Forest ed Extended Isolation Forest

2.2.6 Considerazioni finali

Come si può notare dalla tabella 1 i risultati ottenuti dal LOF sono gli unici che si discostano dagli altri. In figura 5 si comparano i risultati ottenuti dal KNN e quelli del LOF, riportando in dettaglio la sezione del grafico con valori di *album_listens* compresi fra 200000 e 400000, dove si concentrano molti degli *outlier* rilevati dal LOF.

Outlier Detection								
	Popular	**Not Popular**	**Totale**	**In comune LOF**	**In comune ABOD**	**In comune KNN**	**In comune IF**	**In comune EIF**
LOF	37	469	506	-	160	148	151	142
ABOD	133	363	496	160	-	450	427	436
KNN	139	358	497	148	450	-	463	470
IF	135	365	500	151	427	463	-	460
EIF	138	362	500	142	436	470	460	-

Tabella 1: Tabella riassuntiva degli *outlier* identificati

Considerando la natura del dataset e gli obiettivi della nostra ricerca, gli *outlier* rilevati dai vari metodi non possono essere rimossi acriticamente: per definizione, gli album popolari sono degli "*outlier*" e il dataset è inoltre già particolarmente sbilanciato a favore degli album non popolari. Si è quindi deciso di rimuovere i dati con *track_listens* e *album_listens* superiori rispettivamente a 100000 e 40000, considerati *outlier* da tutti i metodi (e che costituiscono il *top 1%* degli *outlier*), e di rimuovere i dati considerati *outlier* anche da un solo metodo con popularity = 0, per rendere il dataset meno sbilanciato, dopo aver verificato che tali dati avessero un *track_listens* o un *album_listens* inferiori rispettivamente a 50000 e 200000 (ovvero, che fossero nella sezione più popolata

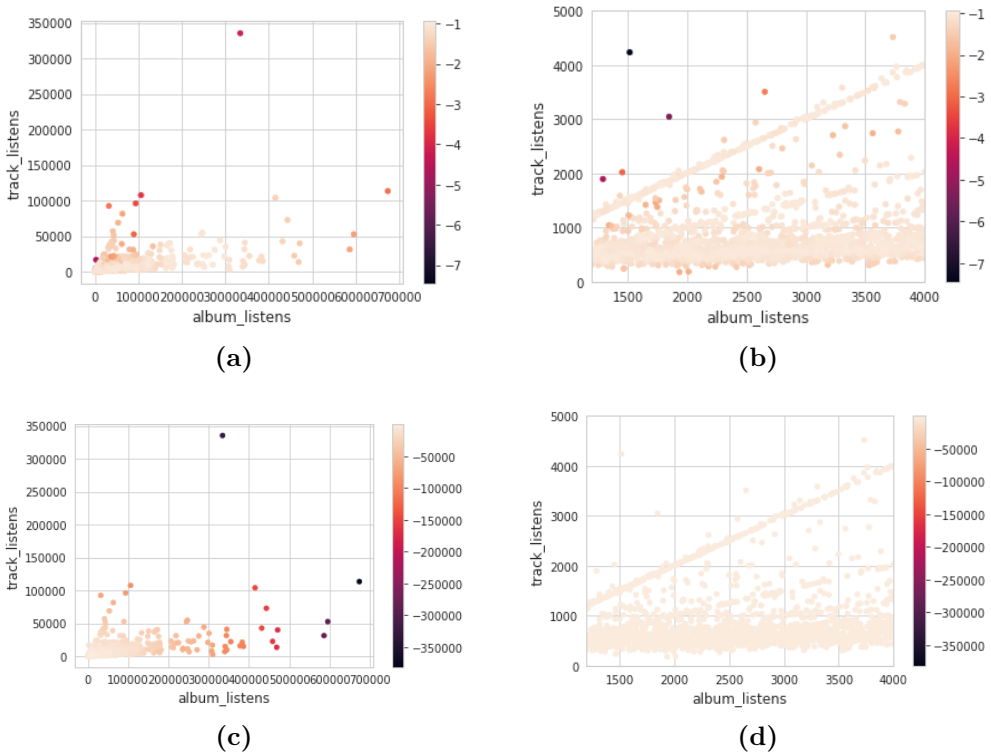


Figura 5: Scatterplot degli *outlier* individuati rispettivamente da LOF e KNN

del grafico in modo tale che la loro rimozione non rendesse il dataset *biased*). Per questo secondo punto sono stati in totale rimossi 781 album (di cui 469 rilevati dal LOF, che si è dimostrato qui particolarmente utile).

popularity	album_listens	tracks_listens	Posizione LOF	Posizione ABOD	Posizione KNN	Posizione IF	Posizione EIF
1.0	674347	113.310.833.333	10	2	1	1	1
1.0	595875	52.460.071.429	14	4	3	5	2
1.0	586634	31.185.000.000	17	3	4	8	3
1.0	471199	39.722.583.333	82	8	5	6	5
1.0	468967	13.411.000.000	89	11	6	19	7
1.0	458874	22.302.750.000	114	12	7	13	8
1.0	443286	72.685.000.000	123	6	8	4	6
1.0	431787	42.479.333.333	289	10	10	7	10
1.0	415275	103.803.000.000	137	5	9	3	9
1.0	334514	335.215.000.000	5	1	2	2	4
0.0	105913	107.451.000.000	6	9	18	9	16

Tabella 2: Top 1% degli *outlier*

2.3 Conclusioni

A conclusione dell'operazione di *Data Preparation* è stato ottenuto un dataset formato da 4263 record e 28 attributi (di cui 16 dati dallo *one hot encoding* dei generi e 5 dallo *one hot encoding* di *Album_type*). Di seguito presentiamo la matrice di correlazione con tutte le variabili (includendo anche *Album_favorites*, che, come detto precedentemente, ha costituito la base per la creazione della variabile target *popularity*).

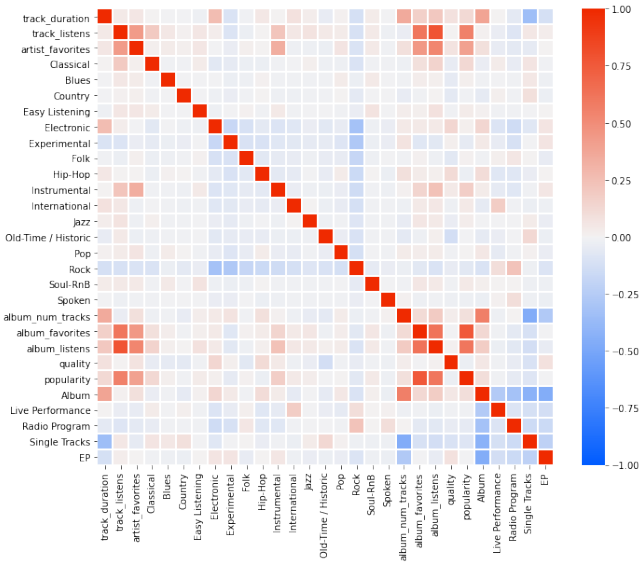


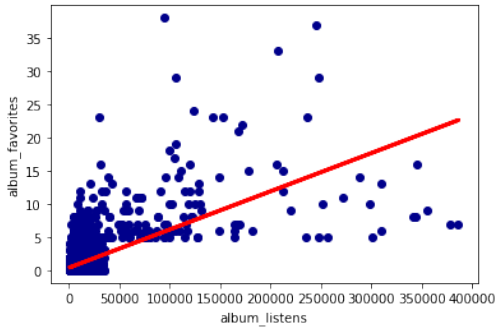
Figura 6: *Correlation matrix* delle feature usate nell'analisi dopo le operazioni di *data cleaning* e *data transformation*

3 Regressione

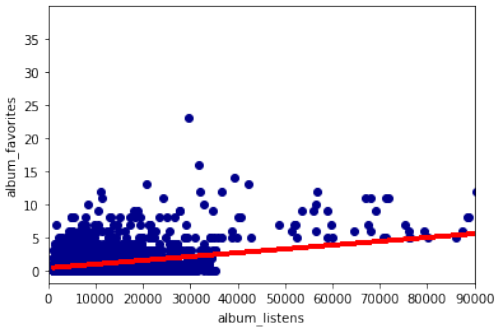
In questa sezione abbiamo affrontato due problemi di regressione lineare, utilizzando per entrambi come variabile indipendente *album_favorites*, ovvero l'attributo usato per la creazione della variabile target. Gli attributi *album_listens* e *track_listens* sono stati invece utilizzati come variabili dipendenti per ciascuno dei due problemi, ottenendo buoni risultati in termini di misura dell'errore ma una retta che apparentemente non riesce a visualizzare correttamente la relazione fra *album_listens* o *track_listens* e *album_favorites*. L'andamento della retta è, infatti, largamente influenzato dalla distribuzione sbilanciata dei valori (come già visto in Figura 2), con una maggior densità di dati con bassi valori di *album_listens* o *track_listens*.
Dopo aver effettuato la regressione lineare per gli attributi *album_listens* e *track_listens* presi singolarmente, i due problemi sono stati generalizzati in una *multiple linear regression* con *track_listens*, *album_num_tracks* e *track_duration* (ovvero, tutte le feature continue ad eccezione di *album_listens*). Per effettuare una multiple linear regression includendo anche *album_listens* (che presente forte collinearità con le altre *feature* continue) si è ricorso al metodo ***Ridge***.
Infine, è stato applicato il metodo ***Lasso*** all'intero dataset, sfruttando la sua natura di *selection operator*. Applicando il metodo Lasso con alpha=2 si sono ottenuti i risultati migliori (come si può osservare nella tabella 3, valori superiori di alpha incrementavano i risultati in forma lieve). Si è notato che il coefficiente di nessuna delle feature è stato ridotto a 0 dal metodo Lasso: ciò significa che tutte le feature hanno, in una qualche misura, contribuito alla regressione.

	Linear regression		Multiple linear regression		
	album_listens	**track_listens**	**track_listens, track_duration, album_num_tracks**	**Ridge**	**Lasso**
R2	0,398	0,374	0,399	0,44	0,415
MSE	3,368	3,499	3,359	3,132	0,035
MAE	0,997	1,065	0,996	0,989	0,073

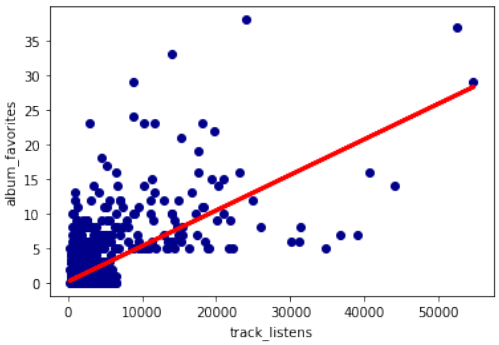
Tabella 3: Tabella riassuntiva dei risultati ottenuti con l'applicazione degli algoritmi di regressione



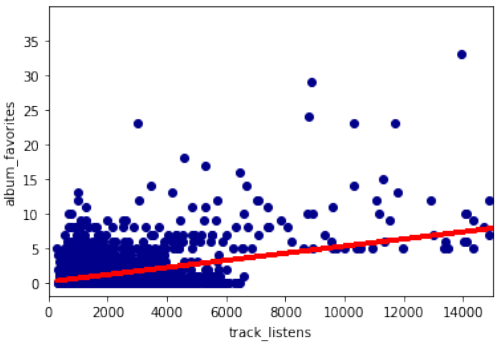
(a) Grafico della regressione con *album_listens*



(b) Zoom sul grafico della regressione con *album_listens*



(c) Grafico della regressione con *track_listens*



(d) Zoom sul grafico della regressione con *track_listens*

Figura 6: Regressione lineare ottenuta usando gli attributi *album_listens* e *track_listens* e rispettivi ingrandimenti dei grafici

4 Classificazione

4.1 Introduzione

Nei seguenti paragrafi abbiamo individuato un problema di classificazione e si è cercato di risolverlo attraverso diversi metodi di classificazione, usando anche algoritmi di *sampling* per bilanciare i dati. Il problema che ci siamo posti è stato ricercare nel dataset quali fossero gli album popolari e quali fossero le principali caratteristiche che

li rendono tali. I metodi utilizzati per le nostre analisi sono stati i ***Decision Tree*** e la sua variante *ensemble*, ovvero il ***Random Forest***, il ***Naïve Bayes Classifier*** con l'approccio *Gaussian*, *Categorical* e *Bernoulli*, la ***Logistic Regression***, il ***Rule-based Classifier***, le ***Support Vector Machine*** (SVM), il ***Multilayer Perceptron*** e, infine, le ***Deep Neural Network***.

4.2 Decision Tree Classifier & KNN

I primi metodi utilizzati per la classificazione sono stati i ***Decision Tree*** e il **KNN**[5]. Dalla **PCA** in figura 7, ottenuta dopo aver effettuato un *train test validation split* con proporzione 60-20-20, si può notare la forte discrepanza tra album *popular* e *not popular* (224 contro 4039) e la loro particolare distribuzione, che conferma la nostra intuizione del precedente capitolo: statisticamente, gli album popolari sono degli *outlier*. Per questo motivo i risultati ottenuti dal *Decision Tree* (con i parametri migliori cercati attraverso un algoritmo di ***Grid Search***) hanno presentato risultati poco soddisfacenti nella corretta identificazione degli album popolari, risultati confermati dal basso F-1 score del metodo KNN.

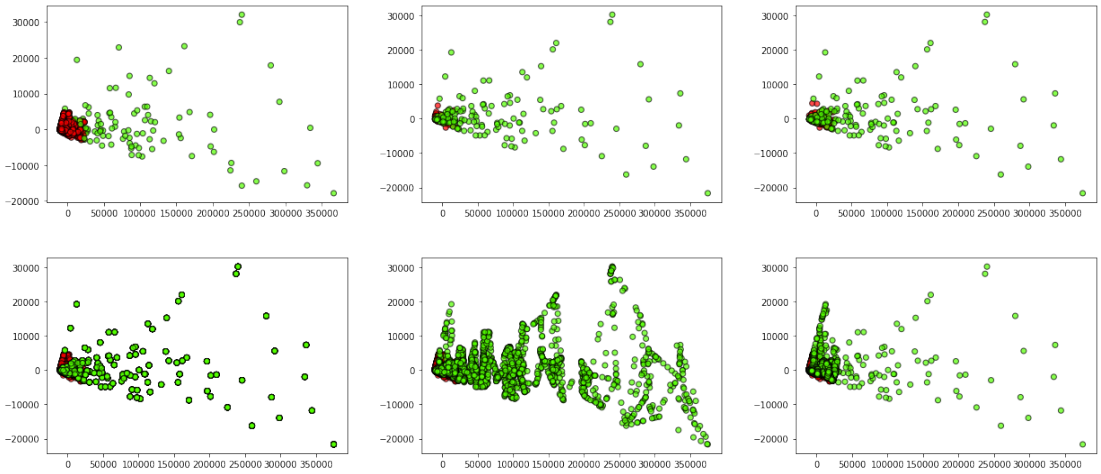


Figura 7: PCA applicata sul dataset originale e sul dataset dopo l'applicazione di *RUS, CNN, ROS, SMOTE-NC* e *ADASYN*

Si sono quindi applicati vari metodi di *undersampling* e *oversampling*, ripetendo per ciascuno la ricerca dei parametri migliori con la *grid search*. Sia i due metodi di *undersampling* utilizzati (***Random Undersampling*** (RUS) e ***Condensed Nearest Neighbors*** (CNN)) sia i tre di *oversampling* (***Random Oversampling*** (ROS), ***Synthetic Minority Oversampling TEchnique*** (SMOTE) nella sua variante ***Nominal Continuos*** (NC), preferita alla tradizionale per la la presenza di attributi binari o categorici nel dataset, e ***Adaptive Synthetic*** (ADASYN)) hanno nettamente migliorato la *recall* a scapito della *precision*. I risultati complessivamente migliori, anche per il KNN, sono stati ottenuti dal *Random Oversampling*.

[5]Il KNN è stato applicato con i seguenti parametri: n_neighbors=20, leaf_size = 10, metric='euclidean'. Prima dell'applicazione si è inoltre proceduto allo scaling del dataset, analogamente a quanto mostrato più avanti con il metodo Support Vector Machine.

Decision Tree & KNN						
	OG (4039, 224)	**RUS (157, 157)**	**CNN (262, 157)**	**ROS (2827, 2827)**	**SMOTE-NC (2827, 2827)**	**ADASYN (2841, 2827)**
DT: min_samples_split	3	20	4	15	7	16
DT: min_samples_leaf	29	4	5	3	2	6
DT: max_depth	13	13	13	18	14	11
DT: criterion	Gini	Gini	Gini	Gini	Gini	Gini
DT: Precision	0,98 - 1	0,99 - 0,27	0,99 - 0,39	0,99 - 0,82	0,99 - 0,63	0,99 - 0,49
DT: Recall	1 - 0,55	0,87 - 0,88	0,93 - 0,84	0,99 - 0,88	0,98 - 0,76	0,95 - 0,79
DT: F-1	0,99 - 0.71	0,93 - 0,42	0,96 - 0,53	0,99 - 0,85	0,98 - 0,69	0,97 - 0,60
KNN: AUC	0.9475 (+/- 0.00)	0.8807 (+/- 0.03)	0.8746 (+/- 0.02)	0.8533 (+/- 0.02)	0.8442 (+/- 0.02)	0.7410 (+/- 0.08)
KNN: F-1	0.4865 (+/- 0.00)	0.5961 (+/- 0.12)	0.5109 (+/- 0.10)	0.8454 (+/- 0.02)	0.8352 (+/- 0.03)	0.7357 (+/- 0.08)

Tabella 4: Tabella riassuntiva dei risultati ottenuti con i modelli *Decision Tree* e *KNN* con diversi dataset ottenuti per mezzo di tecniche di *imbalanced learning*. Per ogni metodo utilizzato sono stati riportati i parametri che hanno garantito i migliori risultati dopo la prima fase di *tuning*.

Analizzando l'albero ottenuto con il ROS (figura 9) si nota che i principali attributi utilizzati per classificare un album come popolare o meno sono i tre continui: *album_listens*, *track_listens* e *artist_favorites*. Alcuni dei generi e *track_duration* vengono utilizzati nei successivi nodi per ridurre quanto più possibile l'indice di Gini.

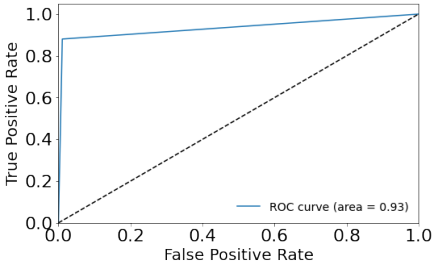


Figura 8: *ROC curve* ottenuta dal metodo *Decision Tree* sul dataset con *Random Oversampling*

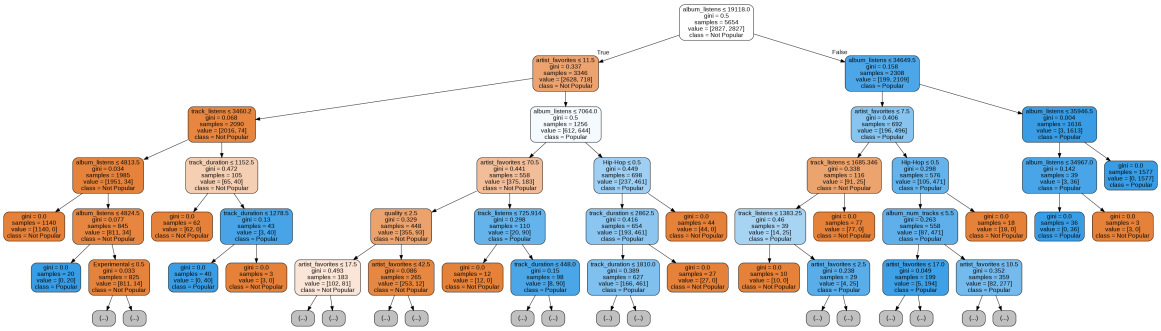
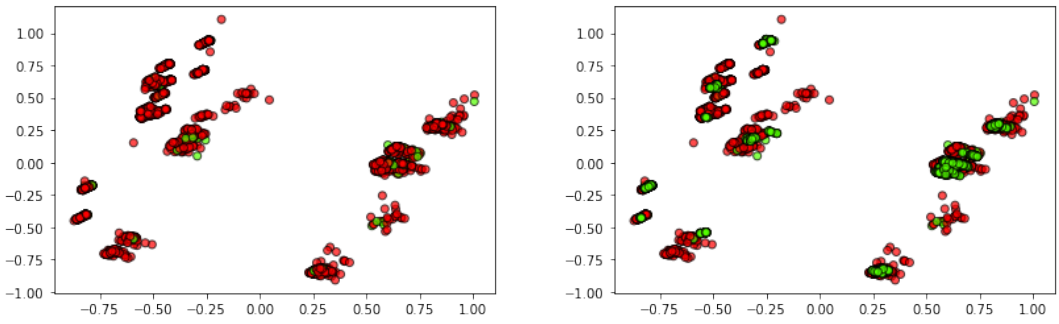


Figura 9: Albero ottenuto con l'algoritmo *Decision Tree* in seguito all'applicazione del *Random Oversampling*

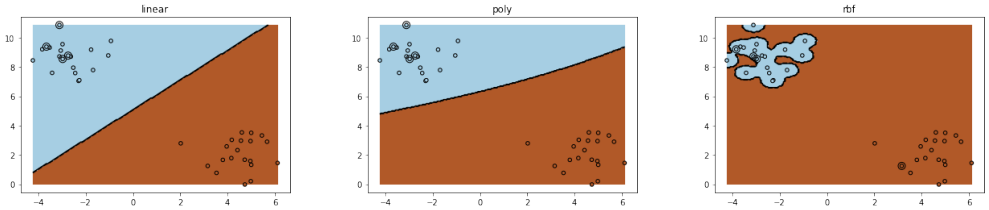
4.3 Support Vector Machine

Il modello ***Support Vector Machine*** permette di suddividere i vari dati per mezzo di un *iperpiano*, a seconda del *kernel* selezionato in modo lineare o non-lineare. Poiché il modello si basa sullo studio e la divisione della distribuzione dei punti nello spazio, prima della sua applicazione si è proceduto allo *scaling* dei dati: in figura 10a la PCA del dataset scalato.
Attraverso il metodo *grid search* si sono poi cercati i parametri migliori per il nostro dataset, sia con un kernel lineare, sia con un kernel di tipo *poly* o *rbf*. A diverso kernel corrisponde una diversa conformazione dell'iperpiano, come risulta in figura 11.
La ***Linear SVM*** ha prodotto una classificazione con alta *precision*, ma *recall* solo



(a) **(b)**

Figura 10: PCA del dataset dopo l'applicazione dello scaling e dopo la successiva applicazione dello SMOTE-NC



(a) **(b)** **(c)**

Figura 11: Grafici mostranti le diverse conformazioni assunte dall'iperpiano con kernel *linear*, *poly* e *rbf*

leggermente superiore al 50%. L'applicazione dello SMOTE-NC e il bilanciamento i pesi delle due classi hanno invertito i risultati, con una *recall* medio-alto ma una *precision* molto bassa. Simili risultati sono stati ottenuti con le SVM non lineari: l'utilizzo della *grid search* ha restituito come miglior valore per la kernel per il dataset originario il *rbf*, mentre per il dataset a cui si è applicato lo SMOTE-NC, il miglior valore per la kernel è risultato essere il *poly*.
Gli scarsi risultati ottenuti con il dataset originario (con e senza i pesi bilanciati) in fase di test e validation corrispondono a quelli ottenuti in fase di training. Lo stesso non si è verificato con il dataset a cui è stato applicato lo SMOTE-NC: in fase di training si sono notati, infatti, ottimi risultati sia per la linear SVM sia per la non-linear, che, tuttavia, non sono stati confermati successivamente.

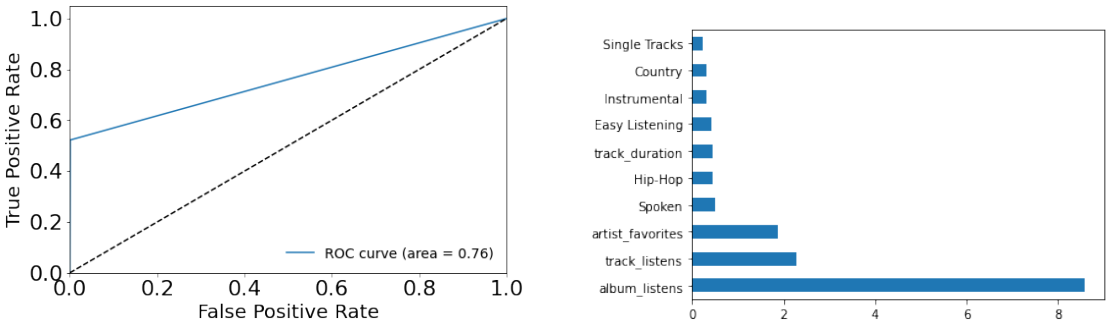
	Linear SVM (OG Dataset)	**Linear SVM (Pesi bilanciati)**	**Linear SVM (SMOTENC)**	**Non-Linear SVM (OG Dataset)**	**Non-Linear SVM (Pesi Bilanciati)**	**Non-Linear SVM (SMOTENC)**
C	10	10	50	100	100	100
Tolerance	1	1	1	0,001	0,001	0,001
Gamma	-	-	-	0.01	0.01	1
Kernel	(Linear)	(Linear)	(Linear)	RBF	RBF	Poly
Precision	0,97 - 0,94	0,99 - 0,33	0,99 - 0,28	0,97 - 0,97	0,98 - 0,47	0,98 - 0,32
Recall	1 - 0,51	0,91 - 0,78	0,89 - 0,76	1 - 0,51	0,96 - 0,63	0,93 - 0,60
F-1	0,99 - 0,66	0,95 - 0,46	0,94 - 0,41	0,99 - 0,67	0,97 - 0,54	0,95 - 0,41

Tabella 5: {Tabella riassuntiva dei risultati ottenuti con le SVM sul *test set* e diversi dataset ottenuti per mezzo di tecniche di *imbalanced learning*.

Si sono, dunque, verificati problemi di *overfitting* (il modello viene addestrato in modo troppo specifico sul dataset di training) e di *data sparseness* (i dati reali a nostra disposizione sono troppo pochi per addestrare correttamente il modello). In tabella 6 riportiamo, come esempio, i risultati ottenuti con la Linear SVM in fase di training.
Per quanto non particolarmente efficace come classificatore, la *Linear SVM* ha comunque fornito alcuni dati interessanti per quanto riguarda l'importanza degli attributi utilizzati per la classificazione: come nel caso del *Decision tree* spiccano sugli altri i tre attributi continui e a seguire i generi, come si può vedere in Figura 12b.

	Linear SVM (SMOTE-NC) Risultati del Train
C	50
Tolerance	1
Precision	0,89 - 0,94
Recall	0,94 - 0,89
F-1	0,92 - 0,91

Tabella 6: Risultati con SVM e SMOTE-NC



(a) **(b)**

Figura 12: Curva di ROC e prime 10 *feature* più rilevanti per la SVM con kernel lineare

4.4 Naïve Bayes

L'algoritmo ***Naïve Bayes*** presenta tre profili: *Gaussian, Categorical* e *Bernoulli*.
Applicando il profilo ***Gaussian*** alle feature continue (*album_listens, track_listens, artist_favorites, track_duration, album_num_tracks*) si sono ottenuti buoni risultati solo per i record con *popularity = 0*. Applicando uno *smoothing* pari a 0.01 si ha avuto un deciso aumento di *Precision* per gli album popolari, a fronte di una lieve diminuzione della *recall*.
Metodologicamente, questa applicazione dell'algoritmo Naive Bayes non è del tutto corretta in quanto alcune feature, come visto nella *correlation matrix*, mostrano correlazione fra loro (in particolare *album_listens*).
Effettuando la classificazione con solo *album_listens* si notano dei risultati migliori per quanto riguarda la *recall* della classe popolare, ma peggiori applicando lo *smoothing*. Effettuando la classificazione con i soli attributi track_listens, artist_favorites, track_duration si è notato invece un peggioramente complessivo (che non riportiamo in tabella per motivi di spazio).
Si è poi applicato il metodo ***Bernoulli*** alle feature binarie (i 16 generi e i 5 *album_type*) e il metodo ***Categorical*** alle feature binarie e categoriche (alle 21 del *Bernoulli* si è aggiunta la feature *quality*), in questo caso sia in forma *one hot encoded* sia effettuando

in *label encoder*. In entrambi i casi a un'alta *precision* si è accompagnata una *recall* quasi nulla per gli album popolari. I risultati identici fra i due metodi lasciano intendere inoltre una scarsa importanza dell'attributo *quality*.
Risultati migliori con il profilo *Categorical* sono stati ottenuti con una discretizzazione approssimata delle feature *album_listens*, *track_listens* e *artist_favorites* (valgono qui analoghe considerazioni sulla correlazione di queste feature). Si è così notato un netto miglioramento della *recall*, rimasta tuttavia inferiore al 60%. Questo risultato conferma l'importanza di *album_listens*, *track_listens* e *artist_favorites* per la classificazione, come già rilevatato con i *Decision tree*.

Naive Bayes Classifier						
	Gaussian	**Gaussian (smooth = 0.01)**	**Bernoulli & categorical**	**Categorical discretizzazione**	**Gaussian album_listens**	**Gaussian album_listens (smooth = 0.01)**
Precision	0.98 - 0,50	0,98 - 0,93	0,95 - 1	0,97 - 1	0,98 - 0,67	0,98 - 0,71
Recall	0,96 - 0,64	0,99 - 0,57	1 - 0,01	1 - 0,51	0,98 - 0,60	0,99 - 0,55
F-1	0,97	0,98 - 0,67	0,97 - 0,03	1 - 0,67	0,98 - 0,63	0,98 - 0,62

Tabella 7: Tabella riassuntiva dei risultati ottenuti con l'applicazione del Naive Bayes Classifier

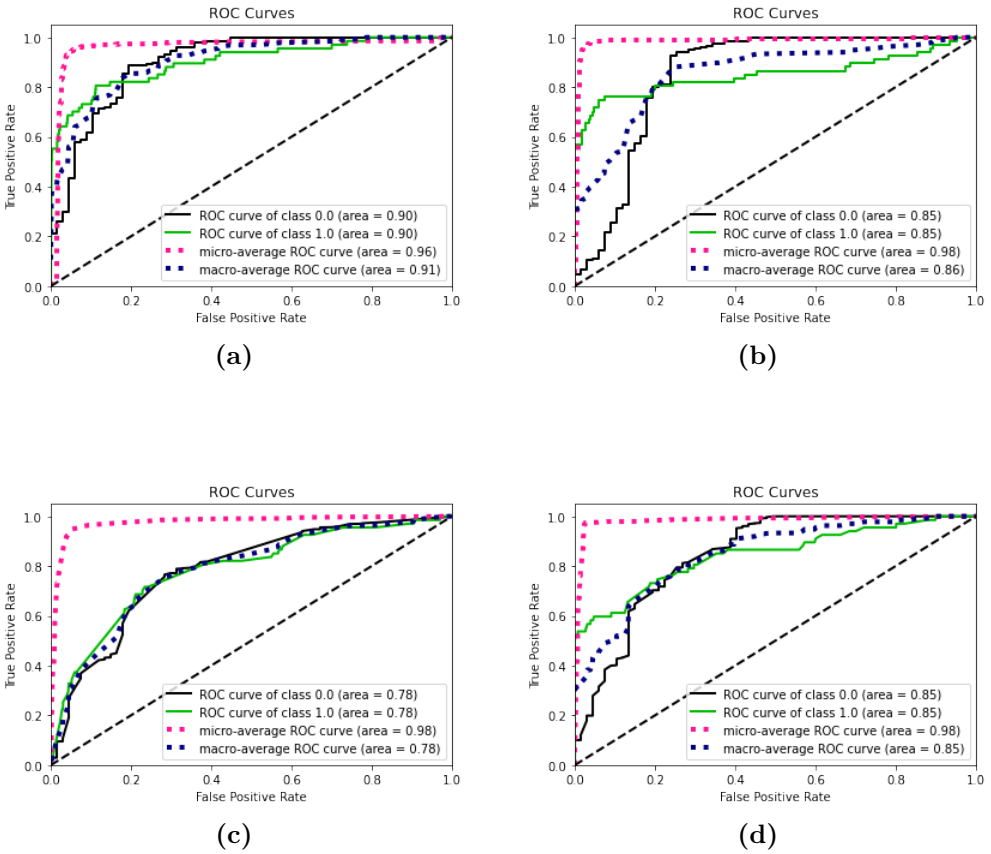


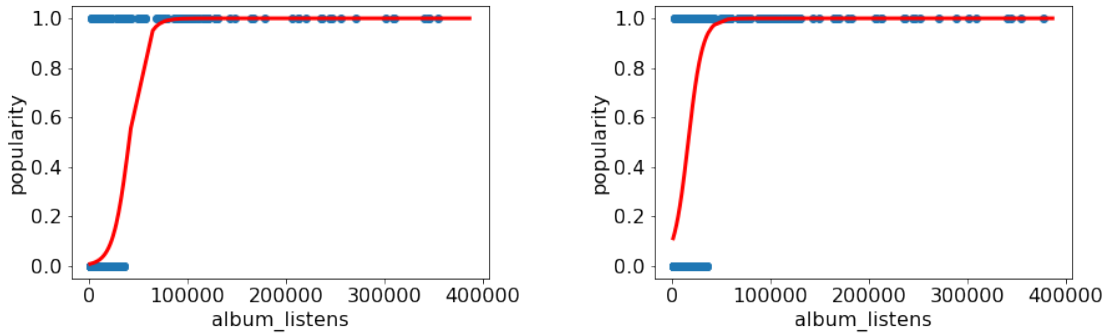
Figura 13: Curve di ROC dei classificatori Naïve Bayes nei profili Gaussian, Gaussian con smooth, Bernoulli e Categorical con feature binarie e categoriche

4.5 Logistic Regression

In seguito è stata applicata, come metodo di classificazione, la ***logistic regression***, ottenendo risultati meno buoni di quelli ottenuti in precedenza con la regressione multilineare.
Utilizzando come variabile indipendente *album_listens* sono stati ottenuti per la classe *popular* risultati soddisfacenti per quanto riguarda la *precision*, ma poco superiori al 50% per la *recall*. Modificando gli iperparametri dell'algoritmo impostando i pesi delle classi *popular* e *not popular* come bilanciati, a un miglioramento della *recall* si è accompagnato un netto abbassamento della *precision*.

Logistic Regression		
	Logistic	**Logistic (pesi bilanciati)**
Precision	0,97 - 1	0,99 - 0,31
Recall	1 - 0,55	0,89 - 0,82
F-1	0,99 - 0,71	0,94 - 0,45

Tabella 8: Risultati regressione logistica con pesi bilanciati



(a) **(b)**

Figura 14: Regressione logistica con pesi non bilanciati e con pesi bilanciati

4.6 Rule-based classifier

Grazie al modulo **Wittgenstein** si è cercato di applicare una **classificazione basata su delle regole**. Impostando un k=1 e una *prune_size* pari a 0.33, Wittgenstein, e la sua implementazione dell'algoritmo **RIPPER** (basato sulla massimizzazione della FOIL con ogni regola) ci ha permesso di estrarre 9 regole dal dataset. Come con il metodo *Support Vector* si è notato un calo delle prestazioni passando da *Train* a *Test* (in particolare per la *Precision* della classe minoritaria).

Wittgenstein		
	Train	**Test**
Precision	0,97 - 0,83	0,97 - 0,66
Recall	0,99 - 0,53	0,99 - 0,51
F-1	0,98 - 0,65	0,98 - 0,57

Tabella 9: Risultati ottenuti sul train e sul test set con l'applicazione del RIPPER

Si nota ovviamente che gli album più popolari sono quelli con più alti valori di *album_listens*, *artist_favorites* e *track_listens*, ma sono più interessanti le regole relative ai generi, ai tipi album e la *quality*.
Dalla regola 2 il tipo di album più popolare sembra essere l'album "canonico", a discapito di EP, Single e antologie di Live Concert o di Radio Program. La qualità degli album (espressa in bitrate nel dataset originario) non sembra incidere sulla popolarità: la regola 3 indica infatti come popolari album con bassa qualità ma alti ascolti, ma di

contro la regola 6 mostra come popolari album di con alta *track_duration* e discreta qualità. Interessante l'ultima regola riportata in tabella 10, che mostra la popolarità del genere Jazz.

Rule-based classifier - RIPPER	
1	album_listens=19049-377713 AND track_listens=2997.57-43983.4 AND artist_favorites=23-153
2	album_listens=19049-377713 AND artist_favorites=153-963 AND Album=1 AND track_listens=2997.57-43983.4
3	album_listens=19049-377713 AND track_listens=2997.57-43983.4 AND quality=1.0
4	album_listens=19049-377713 AND track_listens=2997.57-43983.4 AND album_num_tracks=10-13
5	album_listens=19049-377713 AND track_duration=920-1117 AND album_num_tracks=8-10
6	track_listens=1508.64-2021.0 AND Album=1 AND quality=3.0 AND track_duration=1713-2333
7	artist_favorites=7-13 AND Jazz=1 AND track_listens=2997.57-43983.4

Tabella 10: Regole estratte per mezzo dell'algoritmo RIPPER

Simili regole sono state dedotte utilizzando il software ***Orange***. A differenza di Wittgenstein, *Orange* implementa un algoritmo di tipo CN2: similmente agli alberi di classificazione, le regole cercano di creare insiemi minimizzando la misura di disordine. Le regole estratte con *Orange*, presenti in tabella 11, evidenziano l'importanza data ai generi Electronic, International e Folk.

Rule-based classifier - Orange	
1	album_listens>=35857.0
2	track_listens>=4852.166666666667 AND artist_favorites>=20.0
3	Electronic!=0 AND track_listens>=3557.0
4	International!=0 AND artist_favorites>=8.0 AND album_listens>=16073.0
5	Folk!=0 AND track_listens>=925.0

Tabella 11: Regole estratte per mezzo del software Orange

4.7 Neural Network

4.7.1 Multilayer Perceptron

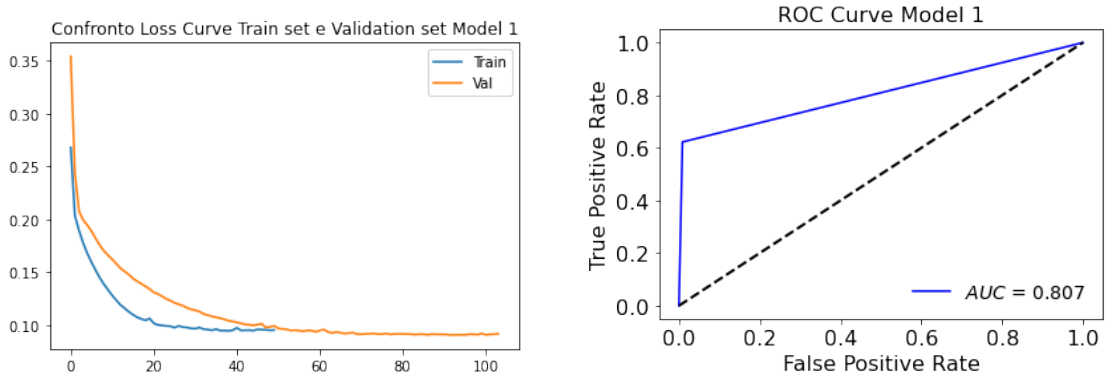
In questo paragrafo è stato utilizzato come classificatore il ***Multilayer Perceptron***, ovvero una rete formata da neuroni aventi, oltre allo strato di input e di output, anche degli *hidden layer*. È stato quindi utilizzato il metodo di *grid search* per la selezione dei migliori valori da assegnare ai parametri *hidden layer*, *learning rate*, *momentum*, *solver* e *activation function*. I valori migliori, come si può vedere in tabella 12, risultano essere 23, 43, 32 come dimensioni dei vari layer, *constant* come *learning rate*, 0 come *momentum* e, infine, *identity* come *activation function* e *adam* come *solver*. Quest'ultimo, in particolare, basato su un'ottimizzazione del tradizionale *SGD* (Stochastic Gradient Based) è ottimizzato per l'utilizzo su grandi dataset.

Multilayer Perceptron		
	Model 1	**Model 2**
Hidden Layer	23, 43, 32	128, 64, 32
Learning Rate	constant	adaptive
Momentum	0	0.9
Solver	adam	adam
Activation	identity	identity
Precision	0.98 - 0.80	0.98 - 0.76
Recall	0.99 - 0.62	0.99 - 0.64
F-1 score	0.98 - 0.70	0.98 - 0.70

Tabella 12: Migliori modelli di Multilayer Perceptron

Successivamente si è provato a implementare un secondo modello aumentando le dimensioni degli *hidden layer* e modificando il *learning rate* e il valore del *momentum*

per verificare se si verificassero dei miglioramenti nella *recall*, ma la rete ha restituito all'incirca gli stessi risultati.



(a) **(b)**

Figura 15: Confronto tra loss curve del train set e validation set e ROC curve del *Model 1*

4.7.2 Deep Neural Network

I problemi menzionati con gli altri metodi (alta *precision* ma bassa *recall* o viceversa) sono stati riscontrati anche testando delle *Deep Neural Network*, addestrate con Keras e Tensorflow con 700 epoche, un *batch_size* pari a 120 e un numero variabile di layer a seconda del modello. Anche in questo caso è stata prima effettuata una *grid search* con gli stessi parametri sovramenzionati e successivamente sono stati creati tre diversi modelli, con i parametri migliori.

Parametri DNN				
	Numero di Hidden Layer	**Dimensioni Hidden Layer**	**Funzione d'Attivazione Hidden Layer**	**Optimizer**
Modello 1	2	128, 64	tahn	adam
Modello 2	1	128	relu	adagrad
Modello 3	3	128, 64, 32	tahn	adagrad

Tabella 13: Parametri per le DNN dopo la fase di *tuning*.

Ciascuno dei modelli è stato testato: senza pesi, con pesi bilanciati guardando alla distribuzioni della classe *popularity* (0: 0.528, 1: 9.40), con pesi bilanciati arbitrariamente (0: 1, 1: 2), con pesi bilanciati ed *early stopping* e con pesi arbitrari ed *early stopping*. Dopo aver testato tutti e tre i modelli in queste varie configurazioni sul *validation set*, è stata applicata sul *test set* solo la configurazione migliore per ciascuno dei tre modelli (guardando in particolare alla *Recall* della classe minoritaria). In tabella 14 riportiamo risultati migliori così ottenuti (e due esempi di configurazioni con alti risultati di *Precision* a scapito della *Recall*).

Risultati DNN					
	Pesi	**Early Stopping**	**Precision**	**Recall**	**F-1**
Modello 1	Bilanciati	Sì	0,99 - 0,19	0,80 - 0,84	0,88 - 0,31
Modello 2	Bilanciati	Sì	0,98 - 0,17	1 - 0,68	0,88 - 0,27
Modello 3	Bilanciati	No	0,99 - 0,22	0,83 - 0,86	0,90 - 0,35
Modello 2	Arbitrari	Sì	0,95 - 1	1 - 0,05	0,97 - 0,10
Modello 3	No	No	0,96 - 0,92	1 - 0,30	0,98 - 0,45

Tabella 14: Migliori configurazioni dei modelli e risultati ottenuti in fase di *validation*. Nelle ultime due righe della tabella sono presentati esempi di configurazioni focalizzate sulla *precision*.

I risultati sono stati confermati dall'applicazione sul *test test*, con un'unica significativa differenza notata nel *Modello 3*. Si è così verificata la bontà dei parametri e l'assenza di *overfitting*.

Risultati DNN					
	Pesi	**Early Stopping**	**Precision**	**Recall**	**F-1**
Modello 1	Bilanciati	Sì	0,99 - 0,20	0,81 - 0,87	0,89 - 0,33
Modello 2	Bilanciati	Sì	0,98 - 0,23	0,87 - 0,71	0,92 - 0,34
Modello 3	Bilanciati	No	0,99 - 0,29	0,90 - 0,76	0,94 - 0,42

Tabella 15: Risultati finali sul test set ottenuti con le Deep Neural Network

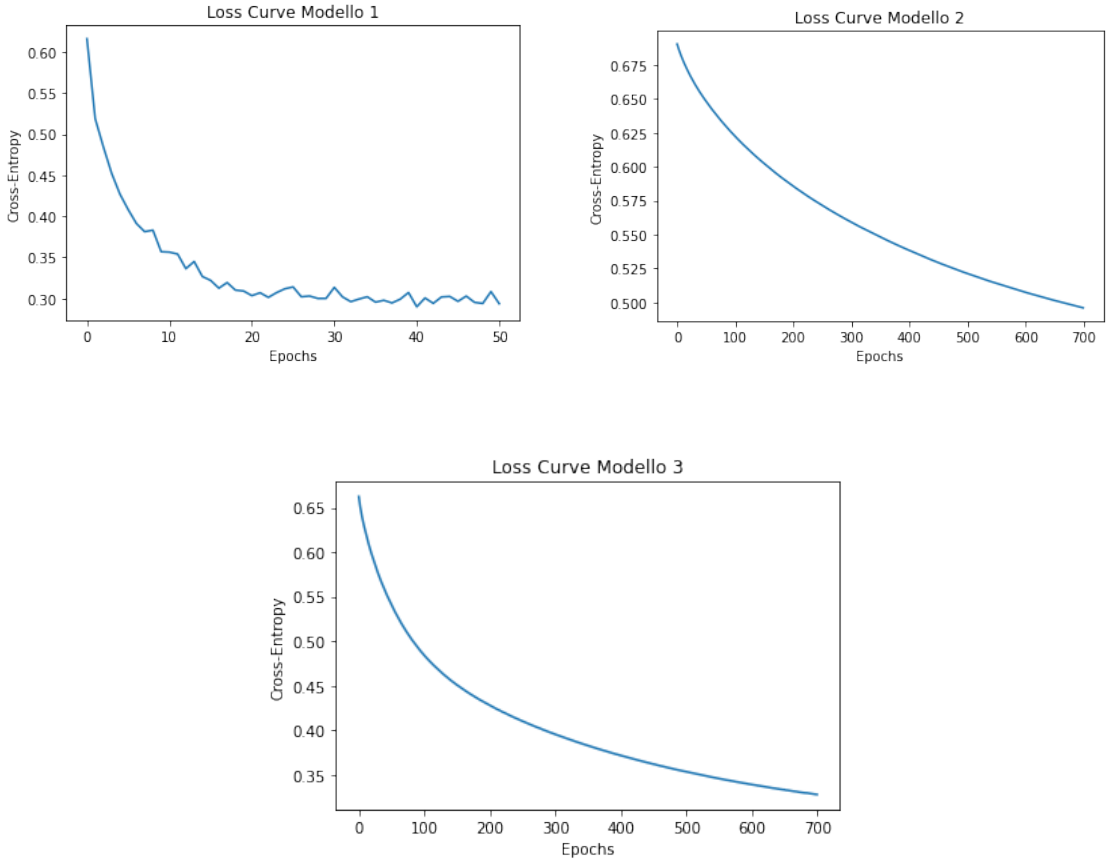


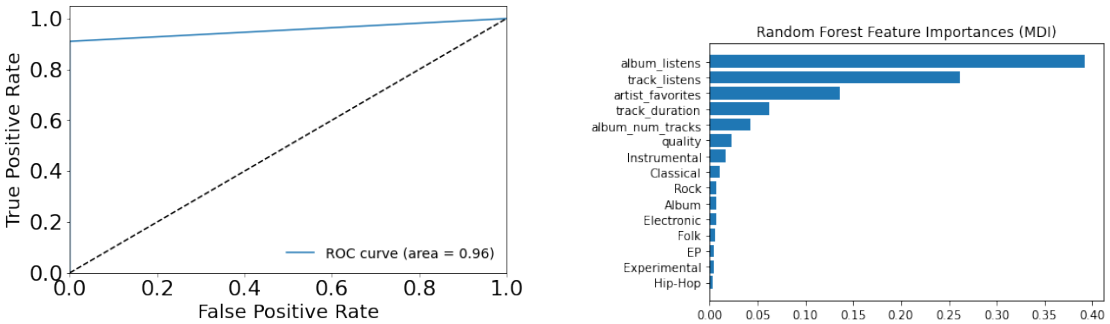
Figura 16: Loss curve dei Modelli 1, 2 e 3.

4.8 Ensemble classifier

Infine, si sono provati dei ***metodi Ensemble*** su alcuni classificatori precedentemente testati. Il metodo *Random Forest* ha nettamente migliorato il *Decision Tree*, portandolo a prestazioni superiori anche a quelle originariamente ottenute applicando il *Random Oversampling* al dataset. Sono state inoltre estratte le feature più importanti per il Random Forest e il rispettivo peso.
Risultati diversi sono stati ottenute dalle Support Vector Machine: l'applicazione del ***Bagging*** ha portato a risultati discreti trovando un bilanciamento fra *precision* e *recall*, ma lo stesso non è avvenuto con il ***Boosting***, i cui risultati sono comparabili a quelli originali. Una possibile spiegazione è data dal fatto che il metodo *Bagging* unisce i risultati di diversi classificatori compensandoli a vicenda, mentre il *Boosting* attribuisce più importanza ai classificatori con i risultati migliori, ma non riesce a fornire complessivamente buoni risultati se mediamente i classificatori non hanno buone prestazioni.
Nella tabella che segue compariamo i risultati ottenuti dai classificatori *ensembled* con le controparti standard già discusse nei rispettive paragrafi.

Ensembled classifier					
	SVM Originale	**SVM Bagging**	**SVM Boost**	**Decision Tree**	**Random Forest**
Precision	0,97 - 0,94	0,98 - 0,88	0,96 - 1	0,98 - 1	1 - 1
Recall	1 - 0,51	1 - 0,62	1 - 0,27	1 - 0,55	1 - 0,91
F-1 score	0,99 - 0,66	0,99 - 0,73	0,98 - 0,42	0,99 - 0,71	1 - 0,95

Tabella 16: Tabella riassuntiva dei risultati ottenuti con le metodologie ensembled a confronto con quelli ottenuti dai metodi standard già discussi nei rispetti paragrafi. In tutti i casi sono stati utilizzati 90 classificatori e gli stessi parametri dei metodi standard



(a) **(b)**

Figura 17: ROC curve del *Random Forest* e prime 15 feature più rilevanti

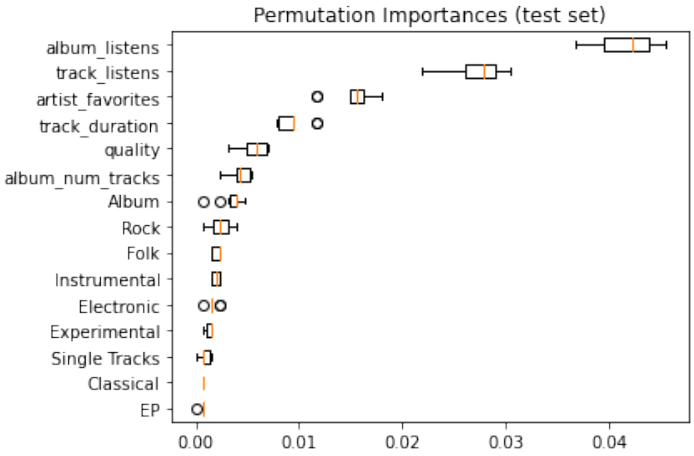


Figura 18: Grafico di permutazione dell'importanza delle feature

4.9 Considerazioni conclusive sui classificatori

I risultati migliori si sono ottenuti con il *Decision tree*, confermati poi dal *Random forest*. In tutti gli altri casi si è sempre notato, nel caso della classe *popular*, una buona *precision* a scapito della *recall* o viceversa.
Tuttavia, per le possibili applicazioni pratiche di questa analisi questo non è necessariamente un problema: ad esempio, nel caso in cui un'azienda volesse selezionare degli album ben apprezzati dal pubblico per una campagna pubblicitaria (ricordiamo che la feature *popularity* è una binarizzazione di *album_favorites*) è preferibile avere alti valori di *precision* pur al netto di una *recall* con valori nella media.
Per quanto riguarda le varie *feature*, si è notata l'importanza degli attributi continui (in particolare *album_listens*, confermata sia dalla *feature selection* del *Decision Tree* e delle *Support Vector Machine*, sia dalla buona regressione lineare, sia dai risultati inizialmente deludenti ottenuti con il Categorical Naive Bayes e migliorati inserendo le feature continue discretizzate. Anche i generi (come Hip-Hop, Classical o Jazz) giocano un ruolo importante, come si può notare dal *Decision tree* e dalle regole inferite con RIPPER e CN2. Non si è tuttavia rilevato un genere che influenzasse pesantemente la popolarità, forse a causa dello sbilanciamento del dataset a favore di alcuni generi: ad esempio, solo 17 album appartengono al genere Blues, mentre 1515 al Rock.[6] Fra gli *album_type*, invece, i *Rule-based Classifier* e i *Decision tree* hanno assegnato un ruolo importante solo al valore *Album* (coerentemente con la matrice di correlazione e con l'importanza data ad *album_num_tracks* e *track_duration*).

5 Time Series

Dopo aver analizzato le tracce raggruppate per album, la nostra analisi si è in seguito concentrata sull'**andamento delle tracce** stesse. Per questa analisi è stato utilizzato un dataset comprendente 1000 tracce, 500 di genere Rock e 500 di genere Folk (due dei generi più determinanti per la popolarità per quanto vista prima). L'attributo *popularity* è stato definito partendo da *track_favorites*: sono popolari le tracce con un *track_favorites* maggiore o uguale a 10. Di ciascuna traccia è stato calcolato l'andamen-

[6] Sbilanciamento già presente nel dataset originario, come già notato nell'introduzione

to temporale con lo smoothing RMS sfruttando la libreria ***librosa***, con un *sampling rate* di 22050, generando 1292 diverse feature.
L'analisi, in questa sezione, ha cercato di rispondere a due domande:

- poiché il genere ha, come visto, un ruolo determinante nella popolarità, se è possibile dedurre il genere di una traccia dalla sua *time series*;
- se è possibile dedurre la popolarità di una traccia dalla sua *time series*.

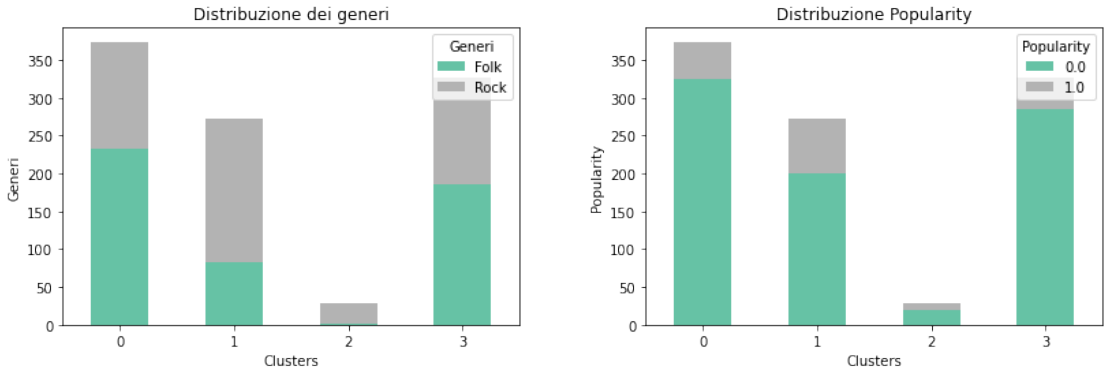
5.1 Clustering

5.1.1 Shape-based clustering

Si è proceduto al clustering delle 1000 canzoni utilizzando l'algoritmo ***KMeans***. Il numero dei cluster è stato fissato a 4 dopo aver utilizzato il *Knee Method* con due differenti metriche. La procedura è stata fatta sia con metrica **Euclidea**, sia con **DTW**: come previsto i risultati migliori, che minimizzano la ***inter-cluster distance***, sono stati ottenuti con la DTW. Focalizzandoci sui risultati ottenuti con quest'ultima metrica, notiamo tre cluster piuttosto grandi (con 374, 326 e 272 elementi) e dalla composizione eterogenea in quanto a generi e *popularity*. Un'eccezione è però costituita dal cluster 2, con sole 28 canzoni tutte (tranne una) di genere rock.

K-Means clustering		
	Euclidean	**DTW**
SSE	372.22	113.025

Tabella 17: Valori di SSE ottenuti con l'applicazione del K-Means

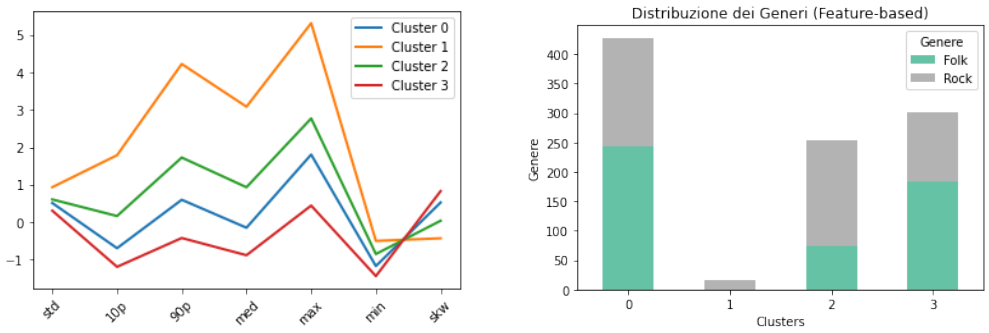


(a) **(b)**

Figura 19: Distribuzione all'interno dei quattro cluster della popolarità e dei due generi (*shape-based).*

5.1.2 Feature-based e Approximation-based Clustering

Si è poi effettuato un *clustering* basato sulle ***feature*** del dataset. Sono state considerate la deviazione standard, il decimo percentile, il novantesimo percentile, la mediana, il massimo, il minimo e la simmetria. Anche in questo caso accanto a tre cluster con 284, 301 e 254 canzoni (e dalla composizione eterogenea, sia per i generi che per la *popularity*) si è formato un più piccolo cluster con sole 17 canzoni rock.



(a) **(b)**

Figura 20: Parallel coordinates e distribuzione dei generi nei quattro cluster *feature-based.*

Dal grafico delle *parallel coordinates* notiamo che il cluster 1 presenta le canzoni con i più alti valori, discostandosi dagli altri. Le canzoni di questo cluster sono quindi caratterizzate da alta **RMS (Root Mean Squared) energy**. I valori più bassi di asimmetria suggeriscono che le *time series* in questo cluster abbiano l'intensità focalizzata sulla destra della curva. Un possibile sottogenere corrispondente a questa descrizione potrebbe essere l'hard rock.
Un simile risultato è stato ottenuto dopo aver approssimato attraverso **PAA** il dataset, generando delle *time series* più compatte composte ciascuna da 30 frammenti (uno per secondo). In questo caso, il cluster più piccolo comprendeva 14 canzoni, tutte di genere rock. Si è verificato che le 14 canzoni rock di questo cluster facessero parte delle 17 del cluster ottenuto con il metodo *feature-based*, a loro volta tutte incluse fra le 27 canzoni rock del cluster più piccolo ottenuto effettuando il clustering direttamente delle *time series*.

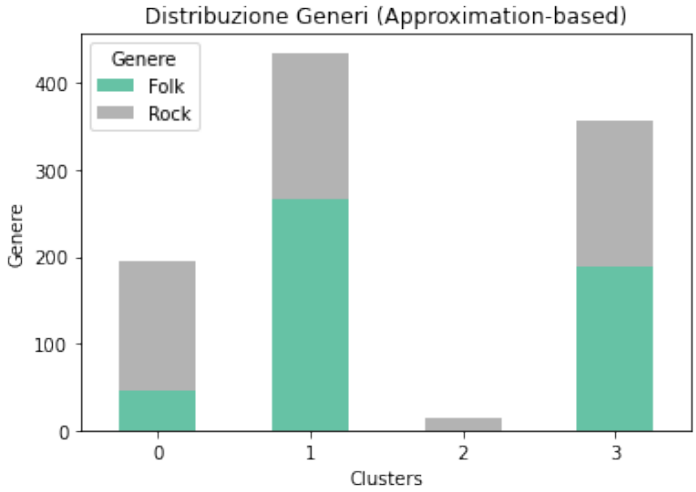


Figura 21: Distribuzione all'interno dei quattro cluster *(approximation-based)* dei due generi.

SSE complessivo			
	K-Means con DTW	**Feature-based**	**Approximation-based**
SSE	**113.025**	**1584.19**	**2.89**

Tabella 18: SSE complessivi ottenuti con l'applicazione di diversi metodi di clustering sulle Time Series

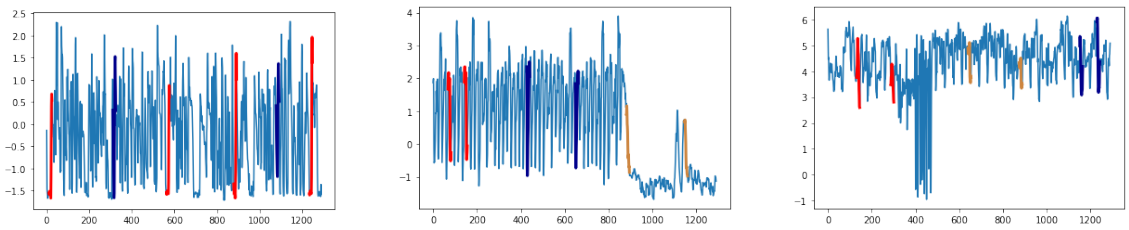
Complessivamente questi risultati suggeriscono che non sia possibile dedurre la popolarità di una canzone dai primi 30 secondi della *time series*, né il macrogenere (rock o folk). L'operazione potrebbe invece essere possibile con uno specifico sottogenere (nel nostro caso, ad esempio, un sottogenere particolare del rock).

5.2 Motif e discord

L'analisi dei ***motif*** e dei ***discord*** non è stata realizzata su un campione casuale, ma su delle *time series* giudicate rappresentative dell'intero dataset. Per trovare queste *time series* "prototipiche" in primo luogo il dataset è stata diviso in due, realizzando un dataset con 500 canzoni rock e un dataset con 500 canzoni folk. Su ciascuna metà è stato effettuato il *clustering* con algoritmo *K-Means* e metrica DTW, analogamente a quanto visto nella sezione precedente. Per ciascun genere sono stati così ottenuti 3 cluster con i rispettivi centroidi. Scegliendo per ciascun cluster la *time series* che minimizzasse la distanza con il rispettivo centroide si è riusciti così a individuare i **medoidi** di ciascun cluster. In totale si sono quindi ottenute sei *time series* prototipiche, tre per il rock e tre per il folk.

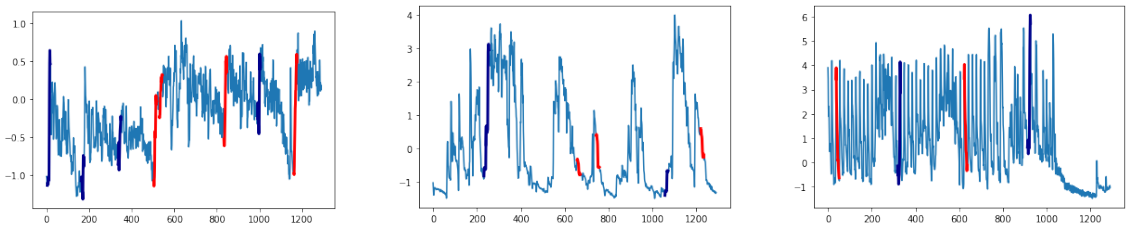
Cluster						
	Rock0	**Rock1**	**Rock2**	**Folk0**	**Folk1**	**Folk2**
Elementi nel cluster	272	211	17	149	209	242

Particolarmente degno di nota risulta essere il cluster *Rock2*, contenente 17 canzoni rock già osservate nei mini-cluster della sezione precedente, che avevamo ipotizzato potessero appartenere al genere *hard rock*.



(a) Motif Rock0 **(b)** Motif Rock1 **(c)** Motif Rock2

Figura 22: *Time series* più rappresentative delle canzoni rock con relativi motif



(a) Motif Folk0 **(b)** Motif Folk1 **(c)** Motif Folk2

Figura 23: *Time series* più rappresentative delle canzoni folk con relativi motif

Osservando più nel dettaglio i motif rilevati, si è notato che quelli delle canzoni rock presentano un andamento cuneiforme, mentre quelli delle canzoni folk rettilineo.

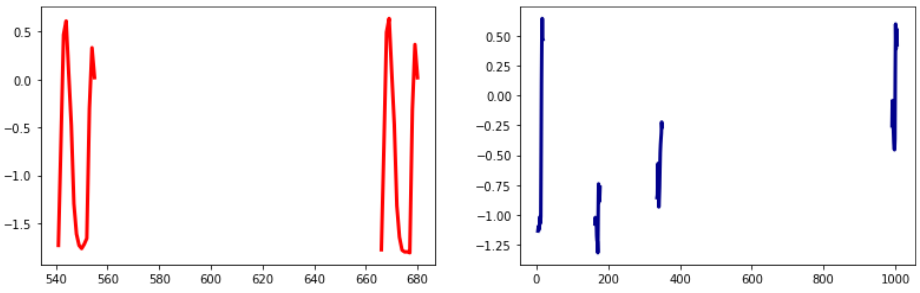
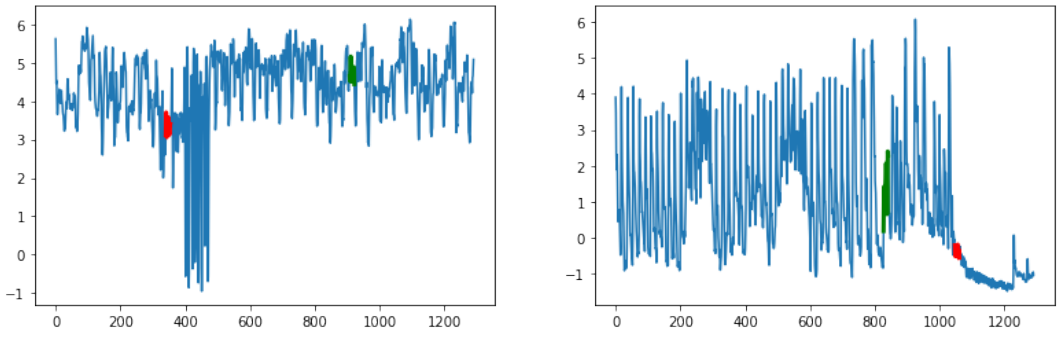


Figura 24: Dettaglio dei motif individuati nel cluster Rock2 e nel cluster Folk0.

Analogamente si è proceduto con le **anomalie**.



(a) **(b)**

Figura 25: Anomalie individuate nelle *time series* Rock2 e Folk2

5.3 Classificazione

I task di classificazione sono stati effettuati con una suddivisione in train, test e validation set, con una proporzione 60 - 20 - 20. È stato in primo luogo creato uno ***shapelet*** con un ottimizzatore SGD e con 300 epoche di apprendimento. Per quanto riguarda la classificazione del rock o del folk sono stati ottenuti in generale risultati sufficienti, con un F-1 score attorno al 60%.
Simili risultati sono stati ottenuti anche effettuando la classificazione in base alla **distanza dagli shapelet**, direttamente sulle *time series* stesse (utilizzando come classi-

ficatore un KNN con distanza DTW) e analizzando le feature delle *time series* (le stesse già usate per il *clustering*).

Classificazione Folk - Rock					
	Shapelet	**Shapelet Distance Based KNN**	**Shapelet Distance Based DT**	**Feature-Based Decision Tree**	**KNN**
Precision	0.68 - 0.61	0.60 - 0.60	0.57 - 0.56	0.62 - 0.66	0.67 - 0.59
Recall	0.53 - 0.75	0.61 - 0.59	0.51 - 0.61	0.71 - 0.56	0.45 - 0.78
F-1	0.68 - 0.68	0.60 - 0.60	0.54 - 0.58	0.66 - 0.61	0.54 - 0.67

Tabella 19: Risultati ottenuti con l'applicazione di diversi algoritmi di classificazione con shapelet per il riconoscimento dei generi Folk e Rock.

Nella tabella a lato è riportato come esempio quanto ottenuto con gli shapelet. Tentativi di classificare le *time series* in base non al genere ma alla variabile *popularity* (dopo aver bilanciato il dataset con un *random undersampling* - l'applicazione dello SMOTE avrebbe generato delle nuove tracce effettivamente inesistenti) hanno portato in tutti i casi a risultati drasticamente peggiori.

Classificazione popularity	
	Shapelet
Precision	0.80 - 0.12
Recall	0.57 - 0.12
F-1	0.67 - 0.17

Tabella 20: Risultati ottenuti con la classificazione della *popularity*

5.4 Sequential Pattern Mining

I due dataset di 500 *time series* l'uno sono stati utilizzati anche nell'individuazione dei ***pattern* ricorrenti** all'interno delle *time series*, riducendo così i tempi di esecuzione e ottenendo pattern distinti per il rock e il folk.
Per ciascuno dei due generi, l'algoritmo **SAX** ha analizzato una delle 3 time series prototipiche, create per i task presentati al paragrafo 5.2, generando, per ciascun genere, un set di etichette utilizzate per discretizzare l'andamento delle tracce del genere corrispondente. Questa discretizzazione è stata fatta su una time serie precedentemente scalata con il TimeSeriesScalerMeanVariance (std = 1, mean = 0). In seguito, è stato scelto un massimo di 40 etichette per entrambi i generi, ottenendo una buona approssimazione della *time serie* modello ed estraendo 32 etichetti per il Rock e 31 per il Folk. Questa approssimazione è stata utilizzata per computare i pattern ricorrenti utilizzando come dataset le sole tre *Time Series* prototipiche di ciascun genere.

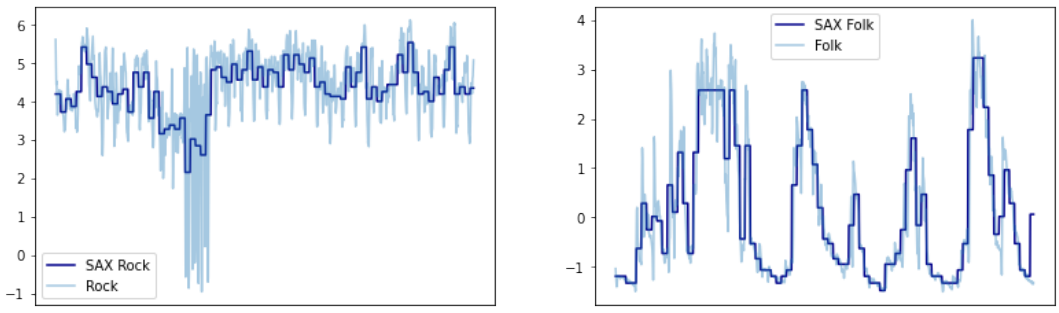
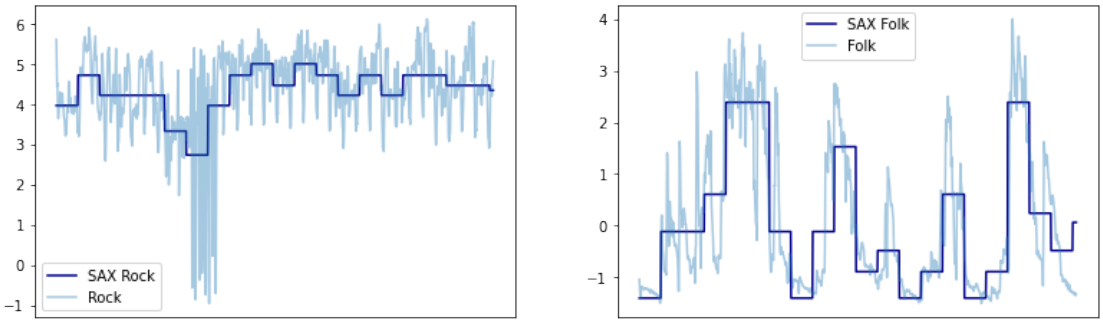


Figura 26: Approssimazione delle time series prototipiche per il genere Rock (27a) e Folk (27b) con 40 etichette.

Sono stati così trovati 718 *pattern* frequenti per il Rock e 308 per il Folk (in entrambi i casi con *support* pari a 3 - sono, quindi, *pattern* comuni a tutti i tre prototipi). Poiché i prototipi sono le *Time Series* più caratteristiche dei due generi, possiamo inferire dal diverso numero di pattern ottenuti che le canzoni Rock del *dataset* originale sono più simili fra loro rispetto alle canzoni Folk, che invece presentano maggiore variabilità intragenere. Questa caratteristica può essere notata anche osservando il numero di canzoni presenti in ciascun cluster alla base dei prototipi (come visto in 5.2).
Di seguito due *sequential pattern* prototipici per ciascun genere con *support* massimo.

Frequent Pattern prototipi Rock (closed)
13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18
13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18
Frequent Pattern prototipi Folk (closed)
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15

In seguito è stato effettuato il *sequential pattern mining* sull'intero *dataset* di 500 canzoni per genere, approssimando tuttavia ulteriormente la *time series* modello, riducendo il numero massimo di etichette a 10, per velocizzare la computazione. In questo modo sono state ottenute 8 etichette per il genere Rock e 9 etichette per il genere Folk.



(a) **(b)**

Figura 27: Approssimazione della time series prototipica dei generi Rock (28a) e Folk (28b) con 10 etichette.

	Etichette	
	Rock	**Folk**
0	-1.6448536269514729	-1.0364333894937898
1	-1.0364333894937898	-0.6744897501960817
2	-0.38532046640756773	-0.38532046640756773
3	-0.12566134685507402	-0.12566134685507402
4	0.0	0.0
5	0.12566134685507416	0.12566134685507416
6	0.38532046640756773	0.38532046640756773
7	0.6744897501960817	1.0364333894937898
8	-	1.6448536269514722

Tabella 21: Etichette ottenute con l'applicazione del SAX.

Successivamente è stato applicato l'algoritmo ***PrefixSpan*** con support a 400, ottenendo però un numero ridotto di pattern frequenti (12 per il rock, 16 per il folk); si è poi provato a diminuire la *threshold*, impostandola a 300, individuando così 55 pattern ricorrenti per il rock e 68 per il folk.
Si sono dunque estratti i 20 pattern più frequenti per entrambi i generi. I più interessanti sono riportati nelle tabelle seguenti.

Frequent Pattern Rock (closed)	
Support	**Pattern**
500	4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
471	6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
470	3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
423	5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
422	3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
393	3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
393	2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
372	2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4

Frequent Pattern Folk (closed)	
Support	**Pattern**
500	4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
477	6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
468	2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
423	3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
419	2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
411	5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
402	3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4
399	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4

Tabella 22: Frequent pattern per Rock e Folk.

Come si può notare, il pattern più frequente, presente in tutte le tracce sia di genere rock che di genere folk, è rappresentato dal valore ripetuto 4, ovvero il valore medio, 0. Analizzando i pattern successivi, vediamo che i valori più alti (7 per il rock, 8 per il folk) non sono presenti in nessuno dei 20 pattern più frequenti. È inoltre interessante evidenziare come il rock presenti quasi in tutte le sue tracce il suo secondo valore più alto (6), a differenza del folk (nei cui pattern più frequenti il 7 è assente). Generalizzando, possiamo affermare che il genere rock presenta delle intensità più elevate rispetto al genere folk; un esempio in questo senso può essere l'ultimo pattern folk presentato in tabella, con i più bassi valori d'intensità.

5.5 Considerazioni conclusive sulle *time series*

Dall'analisi delle *time series* non è stato possibile migliorare i risultati ottenuti dall'analisi degli album per quanto riguarda la *popularity*. I vari metodi di *clustering* non

hanno poi permesso di ottenere delle buone classificazioni per i generi Rock e Folk (da noi selezionati per questa analisi).
Con l'analisi dei *motif* e attraverso il *sequential pattern mining* sono tuttavia emerse alcune differenze fra Rock e Folk. Il risultato più significativo è stato però ottenuto in fase di *clustering*, dove è emersa l'esistenza di un piccolo gruppo di canzoni riconducibili probabilmente a un sottogenere del rock (si è ipotizzato l'*hard rock*). Alla luce di questi risultati, future possibili analisi potrebbero concentrarsi non tanto sui macrogeneri (come Rock o Folk), quanto su sottogeneri specifici (come l'hard rock), o sottogeneri *ibridi* (come il folk-rock), meglio riconosciuti dall'analisi dei primi 30 secondi della *time serie*.

6 Tecniche avanzate di clustering

In questa sezione sono state provate alcune tecniche avanzate di clustering, per verificare alcuni dei risultati ottenuti in precedenza.

6.1 OPTICS

Nella sezione 3 era stato utilizzato il metodo di *outlier detection* LOF, ottenendo risultati drasticamente differenti da quelli registrati con gli altri metodi (come l'ABOD o l'Isolation Forest) a causa della particolare densità del dataset. In questa sezione si è quindi deciso di applicare il metodo di clustering OPTICS, per rianalizzare i risultati del LOF. Impostando un *min_samples* pari a 5 sono stati ottenuti ben 322 cluster, molti dei quali con pochissimi elementi. Per quanto poco utile per un'analisi degli album, questa quantità di cluster è un buon indice dei vari livelli di densità che caratterizzano il dataset. L'OPTICS ha poi rilevato 2423 elementi come *rumore*, fra cui molti degli album con bassi valori di *album_listens* o *track_listens* classificati come *outlier* dal LOF (e non dagli altri metodi di *outlier detection*). I risultati dell'OPTICS sono stati più precisi del LOF per quanto riguarda la zona del dataset con valori di *album_listens* compresi fra 200000 e 400000 *album_listens*, rilevando del rumore in mezzo ai piccoli cluster.
Altri valori del parametro *min_samples* hanno ridotto il numero dei piccoli cluster rilevati, ma non modificato radicalmente la fotografia del rumore.

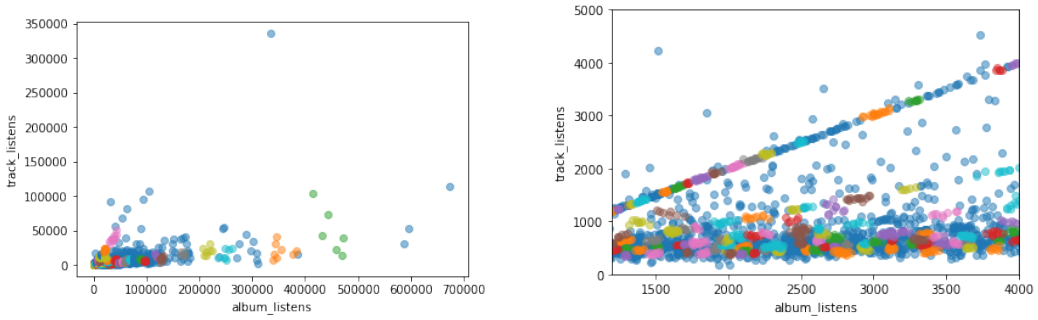


Figura 28: Scatterplot ottenuto in seguito all'applicazione dell'algoritmo OPTICS. A differenza del LOF è stato rilevato del rumore con valori di *album_listens* attorno a 300000. Nel dettaglio si notano i tanti piccoli cluster e la presenza di rumore dove il LOF aveva individuato degli outlier.

6.2 X-Means

Varie applicazioni del *K-Means* alle *time series* delle varie tracce avevano rilevato un costante piccolo cluster formato da sole tracce rock con alti valori di RMS.
Il metodo di clustering *X-Means* ha confermato la presenza di questo piccolo gruppo di canzoni - accanto a sei grandi cluster eterogenei, ha infatti rilevato un piccolo cluster con 12 canzoni rock, le stesse già viste negli analoghi piccoli cluster rilevati dal K-Means.

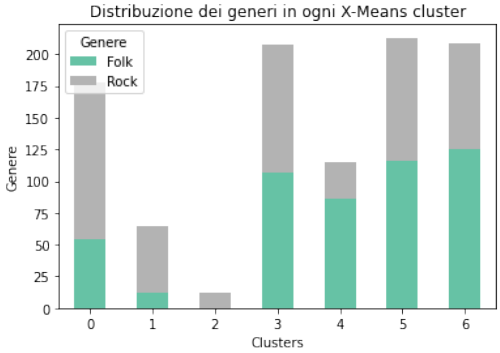


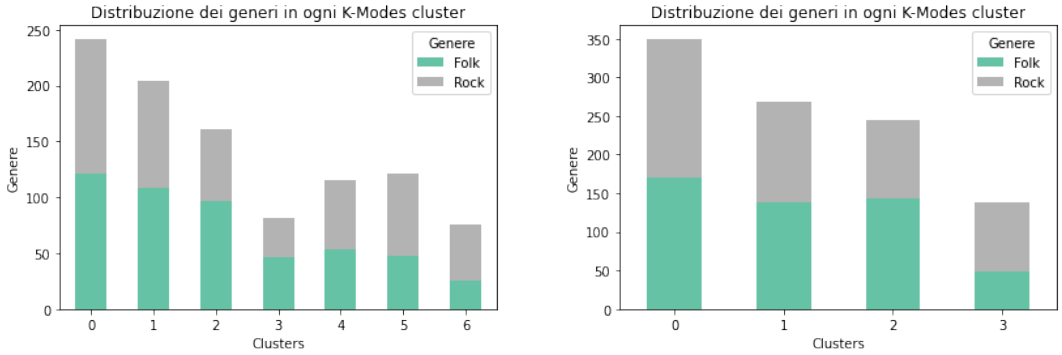
Figura 29: Distribuzione dei generi nei sette cluster individuati

Tabella comparativa delle canzoni 'hard rock'						
ID	**K-Means DTW (Cluster 2)**	**Feature (Cluster 1)**	**Approximated (Cluster 2)**	**X-Means (Cluster 2)**		**Prototipo (Rock 2)**
459	**Sì**	**No**	**No**	No		No
825	Sì	Sì	Sì	Sì		Sì
9491	Sì	No	No	No		No
28692	Sì	Sì	Sì	Sì		Sì
42844	Sì	Sì	No	No		Sì
54163	Sì	No	No	No		No
56552	Sì	No	No	No		No
57892	Sì	Sì	No	No		Sì
58539	Sì	No	No	No		No
58541	Sì	No	No	No		No
58542	Sì	No	No	No		No
61814	Sì	Sì	Sì	Sì		Sì
62586	Sì	No	No	No		Sì
63257	Sì	No	No	No		No
66534	Sì	Sì	Sì	Sì		Sì
66535	Sì	Sì	Sì	Sì		Sì
66536	Sì	Sì	Sì	Sì		Sì
66537	Sì	Sì	Sì	Sì		Sì
66538	Sì	Sì	Sì	Sì		Sì
66539	Sì	Sì	No	No		No
71711	Sì	No	No	No		No
74908	Sì	Sì	Sì	No		Sì
88959	Sì	No	No	No		No
91868	Sì	Sì	Sì	Sì		Sì
91869	Sì	Sì	Sì	No		Sì
93983	Sì	Sì	Sì	Sì		Sì
93985	Sì	Sì	Sì	Sì		Sì
93986	Sì	Sì	Sì	Sì		Sì

Tabella 23: Comparazione delle canzoni incluse nei diversi microcluster

6.3 K-Modes

Il metodo di clustering *K-Modes* è stato applicato al dataset utilizzato nei task precedenti, preprocessato in modo simile a quanto fatto per il *pattern mining*. I valori di tutte le *time series* del dataset sono stati ridefiniti con delle etichette da 0 a 13 estratte da una delle tracce.
Sono state provate diverse quantità di cluster, ottenendo sempre raggruppamenti delle canzoni eterogenei a livello di genere. In questo caso non è emerso il piccolo cluster delle canzoni hard rock, forse perché l'approssimazione con le 13 etichette ha impedito ai valori sopra la media delle loro *time series* di contraddistinguersi.
Riportiamo quanto ottenuto con 7 cluster (il numero suggerito dall'*X-Means*) e 4 cluster (il numero precedentemente utilizzato con il *K-Means*).



(a) **(b)**

Figura 30: Distribuzione dei generi nei 7 e 4 cluster ottenuti con l'applicazione del K-Modes con $k = 7$ e $k = 4$.

7 Explainability

A conclusione del progetto sono stati utilizzati LIME e SHAP. Le nostra analisi si sono concentrate sul *Random Forest*, che aveva dato i risultati migliori, come visto in Sezione 3, applicando i tre metodi sopra citati su un record selezionato casualmente che ha mostrato interessanti risultati di *Prediction probabilities.*
Applicando il metodo LIME, si può notare che alti valori di *album_listens*, *track_listens* e *artist_favorites* influenzano pesantemente la decisione verso la classe popolare. Risulta però poi determinante il tipo di album (Album, EP o Single Tracks) e il Genere. Come già notato precedentemente, l'*Album_type Album* (cioè con almeno 7 tracce o una durata superiore di 30 minuti) risulta correlato con *Popularity* positiva (di contro, il valore *Single Tracks* è determinante per una *Popularity* negativa; era stato, del resto, già notata in Sezione 3 che la correlazione fra il numero di tracce e la popolarità). Per quanto riguarda i generi, il fatto che l'album fosse di musica classica è stato determinante per la *popularity* negativa. La qualità in termini di bitrate dell'album sembra inoltre avere un ruolo importante.

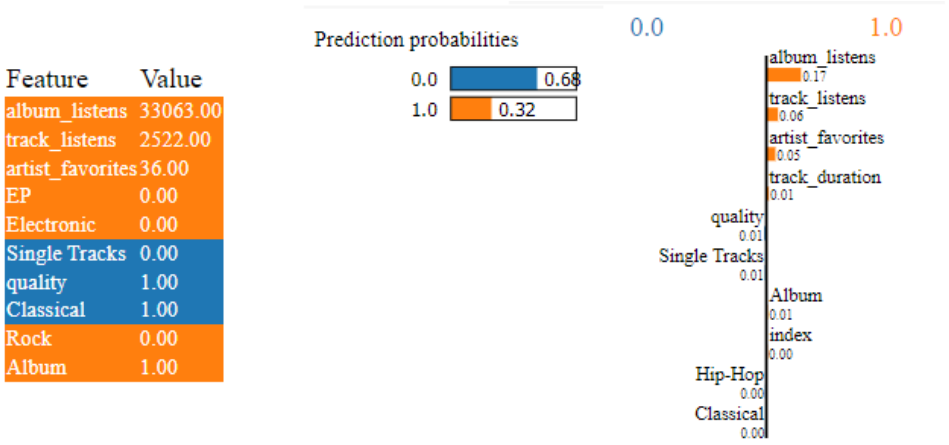


Figura 31: Tabelle ottenute con l'applicazione del LIME su un record *not popular*

Con la successiva applicazione del metodo SHAP è stato ulteriormente analizzato il peso di *Single Tracks* e del genere *Rock*, importanti rispettivamente per *Popularity* negativa e *Popularity* positiva. Ricordiamo però il carattere *biased* del dataset nei confronti del Rock.

(a)

(b)

Figura 32: Grafico ottenuto con l'applicazione del metodo SHAP sul *test set* con *single tracks* e genere Rock.

8 Conclusioni finali

In sintesi, le varie fasi del progetto hanno restituito risultati coerenti fra loro e complessivamente buoni per gli obiettivi prefissati. Nella prima parte del *paper* sono emersi dei risultati significativi per quanto riguarda la classificazione degli *album*, identificando come *feature* più rilevanti per i compiti di classificazione alcuni specifici generi e *album_listens* e *track_listens* (risultato poi confermato dall'*Explainability*). Come notato, gli scarsi valori ottenuti della *Precision* al netto di una buona *Recall* non inficiano la bontà dei risultati per le eventuali applicazioni in ambito aziendale prefigurate all'inizio del progetto.
Nella seconda parte l'analisi della *time series* non ha portato complessivamente a risultati rilevanti, con l'eccezione del gruppo di canzoni identificato come *hard rock*. Eventuali

futuri sviluppi del progetto potrebbero considerare i sottogeneri musicali e gli attributi inclusi nel subset *Echonest* dell'FMA Dataset - ad esempio i valori di *Danceability* e *Acousticness* potrebbero essere messi in relazione con la *time series* della singola traccia. Questo approccio inserirebbe l'analisi della *time series* nel più ampio contesto dell'analisi degli attributi della singola traccia e potrebbe portare a risultati più interessanti di quelli qui ottenuti analizzando la sola *time series* (ad esempio, le tracce dello stesso genere - o sottogenere - potrebbero avere simili valori di *Acousticness*, permettendo un più efficace divisione in *cluster*).