
Progetto Machine Learning

SPOTIFY AWARD

GIANLUCA GIUDICE - 830694

MARCO GRASSI - 829664

Contents

1	Introduzione	3
1.1	Descrizione del problema	3
1.1.1	Spotify	3
1.1.2	Premi per i singoli musicali	3
1.2	Approccio al problema	4
1.3	Struttura del codice	4
2	Dataset	5
2.1	Acquisizione del dataset	5
2.1.1	Dataset spotify - Kaggle	5
2.1.2	Premi canzoni - Wikipedia	5
2.2	Descrizione del dataset	7
2.2.1	Spotify	7
2.2.2	Premi	8
2.3	Data integration	8
2.3.1	Record linkage con MongoDB	8
2.4	Analisi esplorativa	9
2.4.1	Assunzioni sul dominio	9
2.4.2	Creazione dataset bilanciato	13
2.4.3	Preprocessing	13
2.4.4	Distribuzione dei valori	13
2.4.5	Artisti nelle canzoni	17
2.4.6	Correlazione tra features	19
2.4.7	Principal component analysis	20
2.5	Scelta delle features	21
2.5.1	Variabili scartate	21
2.5.2	Rappresentazione one hot encoding degli artisti	22
2.5.3	Variabili categoriche	22
2.5.4	Coordinate PCA	22
2.5.5	Riassunto feature finali utilizzate	23
3	Campagna sperimentale	24
3.1	Approccio	24
3.1.1	10-folds cross validation	24
3.1.2	Model selection	25
3.2	Misure di performance	26
3.3	Support Vector Machine	26
3.4	Decision Tree	26
3.4.1	Scelta del modello	26
3.4.2	Decision Tree Plot	27
3.4.3	Complexity Parameter	27

<i>CONTENTS</i>	2
3.4.4 Risultati esperimenti	28
3.5 Modelli a confronto	32
3.5.1 Confronto secondo metrica ROC	32
3.5.2 Confronto Timings	33
4 Conclusioni	34

Chapter 1

Introduzione

In questo capitolo viene introdotto il problema, il dominio di riferimento e l'approccio adottato per la risoluzione.

1.1 Descrizione del problema

In questo elaborato consideriamo i singoli musicali. Al giorno d'oggi le canzoni vengono spesso ascoltate dagli utenti tramite piattaforme di streaming musicale.

L'obbiettivo del lavoro è quello di analizzare le features dei singoli musicali così da prevedere se una canzone diventerà o meno di successo. Nella sezione successiva verrà meglio specificato cosa si intende per **singolo musicale di successo** (subsection 1.1.2).

1.1.1 Spotify

Spotify è un servizio di riproduzione digitale in streaming di musica, podcast e video, con accesso immediato a milioni di brani e altri contenuti di artisti provenienti da tutto il mondo. Questa piattaforma viene utilizzata da milioni di utenti per ascoltare canzoni e nello specifico singoli musicali.

Da questo servizio è possibile ottenere migliaia di brani musicali, infatti Spotify mette a disposizione una API da cui è possibile scaricare informazioni su brani con associate alcune caratteristiche. Pertanto oltre alle classiche informazioni di un brano come "titolo" o "artisti" si avrà a disposizione una serie di caratteristiche come ad esempio quanto una canzone è "energica" o "ballabile".

1.1.2 Premi per i singoli musicali

Come vedremo in subsection 2.2.1 il dataset ottenuto da Spotify mette a disposizione un campo "popularity" che indica quanto può essere considerata popolare. Tuttavia questa caratteristica non ci sembra adeguata per identificare una canzone come di successo oppure no.

Per questo motivo consideriamo una canzone di successo in base alle certificazioni ottenute, rispettivamente "disco d'oro" o "disco di platino". Queste premi sono dei riconoscimenti vinti da un brano musicale e storicamente fanno riferimento al numero di copie vendute da un singolo. Tuttavia con la costante crescita dell'utilizzo di servizi per lo streaming di brani musicali, da qualche anno questi riconoscimenti vengono assegnati anche considerando il numero di streaming sulle diverse piattaforme, tra cui Spotify.

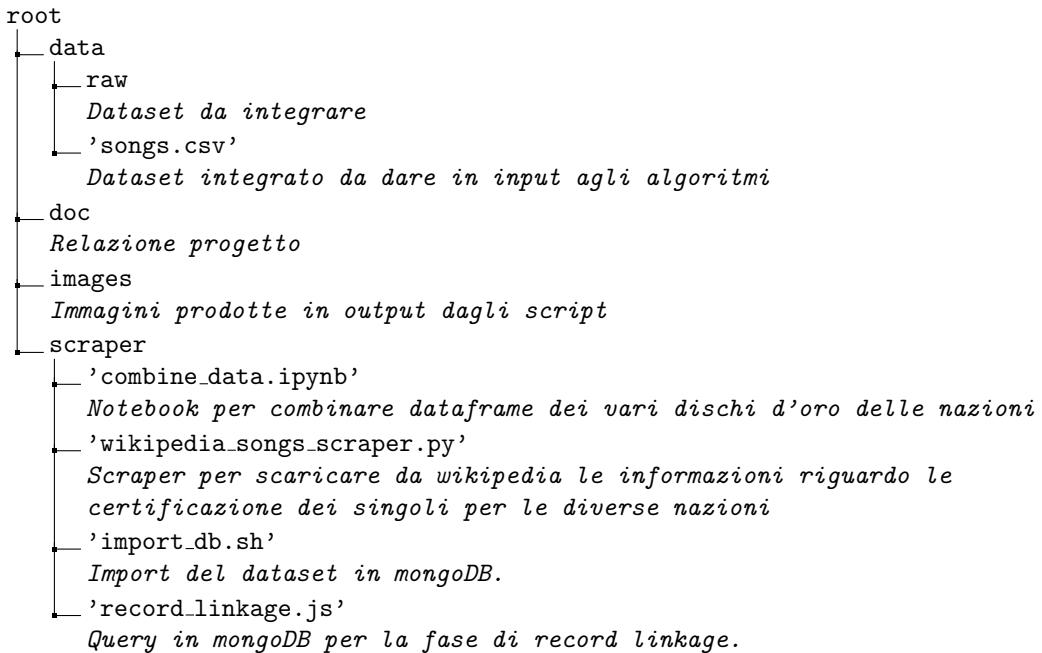
Riteniamo che questo riconoscimento sia una metrica oggettiva per considerare un singolo come di successo.

1.2 Approccio al problema

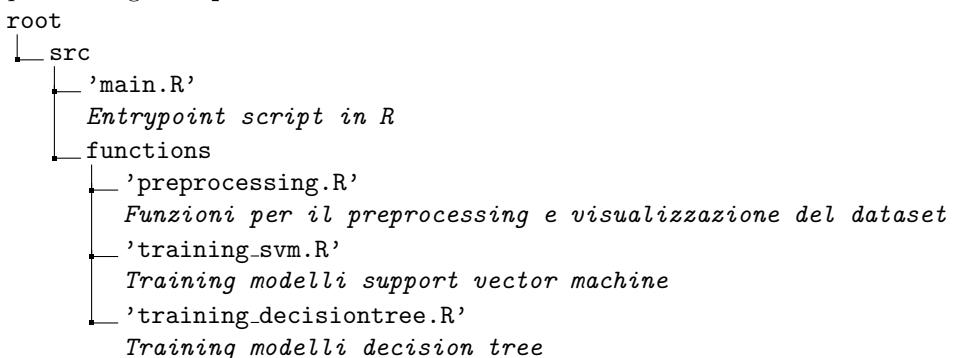
Il problema viene approcciato come un **task di classificazione binaria**. Dato un singolo musicale vogliamo prevedere se questo sarà di successo o no. Pertanto sviluppiamo modelli supervisionati di machine learning partendo da un dataset etichettato, in questo modo è possibile classificare i brani musicali.

1.3 Struttura del codice

Di seguito viene brevemente spiegata la struttura del codice, inoltre viene sotto indicata la working directory e l'entry point del programma.



In particolare gli **script in R** si trovano in:



N.B.: Per come è stato progettato il codice, la working directory è la `root/` e non la cartella `src/`. L'**entry point del programma** è lo script `src/main.R`

Chapter 2

Dataset

In questo capitolo viene analizzato il dataset. Viene quindi spiegato da dove è stato preso il dataset, descritte le covariate e viene effettuata un'analisi esplorativa.

2.1 Acquisizione del dataset

Il dataset completo contenente sia le informazioni sui singoli musicali che le etichette associate ad ogni istanza non è disponibile da una sola sorgente. Pertanto abbiamo ottenuto le informazioni necessarie da diversi sorgenti per poi integrarle in un unico dataset.

2.1.1 Dataset spotify - Kaggle

Per quanto riguarda i brani con le relative informazioni e caratteristiche del brano, Spotify mette a disposizione un'API da cui si possono ottenere questi dati.

Con questa tecnica sono stati ottenuti i dati relativi ai brani, un utente ha quindi caricato il dataset sul sito kaggle.com. Il dataset è disponibile su kaggle all'url indicato¹.

Il dataset contenente queste informazioni è il file: `data/raw/to_integrate/data.csv`.

2.1.2 Premi canzoni - Wikipedia

Le informazioni riguardo i premi delle canzoni, ovvero le varie certificazioni vinte, sono disponibili su wikipedia.org. Questa informazione costituisce di fatto l'etichetta per classificare ogni brano musicale come di successo o non di successo..

Nello specifico siamo interessati a:

- Singoli certificati oro
- Singoli certificati platino
 - 1. Singoli certificati 1 volta platino
 - 2. Singoli certificati 2 volte platino
 - 3. ...
 - 4. Singoli certificati N volte platino

Inoltre le varie certificazioni dei singoli vengono considerati in base al paese. I paesi da noi presi in considerazione sono:

¹**Dataset Spotify:** <https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks>.

- Italia
- Australia
- Stati Uniti d'America
- Regno Unito
- Canda
- Danimarca

Si noti come una certificazione può essere consegnata in diversi paesi alla stessa canzone.

Certificazioni disco d'oro

Per quanto riguarda i dischi d'oro, facciamo riferimento a questa pagina su wikipedia². Fissato quindi uno stato (ad esempio l'Italia) è possibile visualizzare la pagina contenente la lista dei singoli che hanno vinto quel particolare premio. La lista è un elenco di url che puntano alla pagina wikipedia della canzone, un esempio a questo url³.

Certificazioni disco di platino

Ragionamento analogo viene fatto per i dischi di platino, con l'unica differenza che la pagina è questa⁴, inoltre dal momento che i singoli posso vincere più volte un disco di platino, si considerano non solo i singoli che hanno vinto una volta il disch di platino ma anche quelli che l'hanno vinto N volte.

Scraping

Ottenuti quindi i puntatori alle canzoni certificate, è possibile accedere alla pagina wikipedia del singolo, la quale contiene una tabella riassuntiva della canzone. Un esempio di tabella viene mostrato nella Figura 2.1.

²**Disco d'oro per stato:**

https://it.wikipedia.org/wiki/Categoria:Singoli_certificati_oro_per_stato.

³**Singoli certificati disco d'oro in Italia:**

https://it.wikipedia.org/wiki/Categoria:Singoli_certificati_disco_d%27oro_in_Italia.

⁴**Singoli certificati disco di platino per stato:**

https://it.wikipedia.org/wiki/Categoria:Singoli_certificati_platino_per_stato.

Chico (singolo)

Da Wikipedia, l'enciclopedia libera.

Chico è un singolo del rapper italiano Gué Pequeno, pubblicato il 31 luglio 2020 come secondo estratto dal sesto album in studio *Mr. Fini*.^[2]

Indice [nascondi]	
1	Classifiche
1.1	Classifiche settimanali
1.2	Classifiche di fine anno
2	Note
3	Collegamenti esterni

Classifiche [modifica modifica wikitesto]									
Classifiche settimanali [modifica modifica wikitesto]	Classifiche di fine anno [modifica modifica wikitesto]								
<table border="1"> <thead> <tr> <th>Classifica (2020)</th> <th>Posizione massima</th> </tr> </thead> <tbody> <tr> <td>Italia^[3]</td> <td>5</td> </tr> </tbody> </table>	Classifica (2020)	Posizione massima	Italia ^[3]	5	<table border="1"> <thead> <tr> <th>Classifica (2020)</th> <th>Posizione</th> </tr> </thead> <tbody> <tr> <td>Italia^[4]</td> <td>10</td> </tr> </tbody> </table>	Classifica (2020)	Posizione	Italia ^[4]	10
Classifica (2020)	Posizione massima								
Italia ^[3]	5								
Classifica (2020)	Posizione								
Italia ^[4]	10								

Chico

Artista Gué Pequeno
Featuring Rose Villain e Luchè
Tipo album Singolo
Pubblicazione 31 luglio 2020
Durata 3:33
Album di *Mr. Fini*
provenienza
Genere Pop rap
Etichetta Island
Produttore Sixpm
Formati Streaming
Certificazioni
Dischi di platino Italia (3)^[1]
(vendite: 210 000+)
Gué Pequeno - cronologia
 Singolo precedente Singolo successivo
Saigon *Bla Bla*
(2020) (2020)
Luchè - cronologia
 Singolo precedente Singolo successivo
Come me *Maserati (Reloaded)*
(2019) (2020)

Figure 2.1: Esempio di tabella per un singolo musicale su wikipedia

Viene quindi creato uno script per effettuare scraping delle tabelle, il codice è nella directory `scraping/wikipedia_songs_scraper.py`.

Successivamente si integrano le informazioni dei diversi stati, e tipi di certificazione vinti. Il risultato di questa operazione è il file `data/raw/to_integrate/awards_cleaned.csv`.

2.2 Descrizione del dataset

In questa sezione vengono elencate e descritte le features del dataset. Trattandosi di un problema supervisionato, ad ogni istanza viene associata la corretta etichetta, ovvero la classe positiva per i brani musicali che hanno vinto un premio e classe negativa per i brani che non hanno vinto un premio.

2.2.1 Spotify

Di seguito viene descritto il dataset proveniente da kaggle, quindi quello contenente le informazioni dei brani.

ATTRIBUTO	DESCRIZIONE	TIPO
id	Identificativo della canzone (generato da spotify)	Intero
name	Titolo della canzone	Stringa
artists	Lista degli artisti che compaiono nel brano	Stringa
year	Anno del brano	Intero
duration_ms	Durata della traccia in millisecondi	Intero
acousticness	Metrica riguardante quanto un brano risulta "acustico"	Float [0, 1]
danceability	Metrica riguardante quanto una traccia è ballabile	Float [0, 1]
energy	Energia del brano	Float [0, 1]
instrumentalness	Contenuto relativo di strumenti musicali nella traccia	Float [0, 1]
valence	Metrica riguardante la "positività" della traccia	Intero
liveness	Durata relativa della traccia suonata in una performance dal vivo	Float [0, 1]
loudness	Rumorosità della traccia in decibel (dB)	Float [-60, 0]
release_date	Anno di rilascio del brano	Intero
speechiness	Contenuto relativo di voce umana nella traccia	Float [0, 1]
tempo	BPM della traccia	Float
key	Scala musicale utilizzata	Factor {0, 1, ..., 11}
mode	Indica se la traccia parte con una progressione armonica	Booleano
explicit	Indica se la traccia è esplicita oppure no (linguaggio volgare)	Booleano
popularity	Popolarità della traccia	Float [0, 100]

2.2.2 Premi

Si noti come la distinzione tra tipo di premio vinto e lo stato in cui è stata ottenuta la certificazione per uno specifico brano, viene fatta solo nel contesto della attività di scraping, in quanto i brani sono così rappresentati sul sito di wikipedia. Tuttavia da questo punto in poi non verrà più tenuto conto di questa informazione, infatti un singolo verrà considerato come **vincitore di un premio** (quindi di successo) oppure come **non vincitore di un premio** (non di successo). I brani musicali vincitori di un premio appartengono alla classe positiva, viceversa quelli che non hanno vinto un premio alla classe negativa.

Il risultato dello scraping della tabella di wikipedia è il seguente:

ATTRIBUTO	DESCRIZIONE	TIPO
title	Titolo della canzone	Stringa
artists	Artisti presenti nella traccia	Stringa
date	Data di rilascio della traccia	Data
genre	Genere musicale del brano	Stringa
award	Premio vinto dal singolo	Stringa {Oro, 1-platino, 2-platino, ...}
nation	Paese in cui è stato vinto il premio	Stringa (Sigla del paese)

2.3 Data integration

Date queste due sorgenti dati, è necessario integrare i dati allo scopo di avere un dataset etichettato, le label saranno appunto se un brano musicale ha vinto un premio (quindi è di successo) oppure no.

La strategia di entity resolution adottata è considerare un singolo musicale come una singola entità basandosi sul titolo e gli artisti di una canzone. Se il nome del brano musicale è lo stesso e gli artisti coincidono, allora il brano è il medesimo.

2.3.1 Record linkage con MongoDB

A questo scopo i due dataset vengono importati in mongoDB, ogni istanza è rappresentata da un documento. Per la fase di record linkage vengono effettuati i seguenti passi:

1. Import dei dataset in mongodb, inizialmente in due collezioni diverse.
2. Normalizzazione dei dati, i campi di join vengono trasformati in lowercase.

3. Creazione indici.
4. Entrambe le collezioni hanno il campo "artisti" il quale consiste in una stringa contenente la lista degli artisti separati dal carattere ", ". Viene quindi eseguito l'unfold del campo "artista" in entrambe le collezioni, costruendo una lista di artisti per ogni documento facendo uno split sul carattere ", ".
5. Viene eseguita la join sul campo titolo considerando un match valido se l'intersezione tra gli insiemi di artisti dei documenti delle due collezioni non è vuota.
6. Viene ritrasformato il campo artista appiattendo la lista e rappresentando l'insieme degli artisti di un brano come una stringa, separando ogni artista con il carattere ", ".
7. Dump del database in un file .csv, questo è il dataset integrato e verrà usato per il training dei modelli.

2.4 Analisi esplorativa

In questa sezione vengono analizzate e discusse le covariate del dataset. Dopo la fase di record linkage le feature del dataset finale sono:

Covariata	id	name	artists	year	duration_ms	acousticness
Utilizzata	No	No	Si	No	Si	Si
Covariata		danceability	energy	instrumentalness	valence	
Utilizzata		Si	Si	Si	Si	
Covariata		liveness	loudness	release_date	speechiness	tempo
Utilizzata		Si	Si	No	Si	Si
Covariata		key	mode	explicit	popularity	award
Utilizzata	Si	Si	Si	No	Label	

Table 2.1: Tabella riassuntiva di tutte le features del dataset.

Sotto viene riportata una tabella riassuntiva riguardo il numero di istanze nel dataset. Il significato dei threshold viene spiegato nella sezione successiva (subsection 2.4.1).

DATASET	# ISTANZE
Dataset completo	151762
Dataset ($release_date \geq 2005 \& popularity > 25$)	26134
Dataset bilanciato ($release_date \geq 2005 \& popularity > 25$)	3670

Table 2.2: Numero di istanze nel dataset.

2.4.1 Assunzioni sul dominio

Anno di uscita singoli

Viene per prima cosa analizzata la distribuzione degli anni di uscita nel dataset.



Figure 2.2: Distribuzione anno delle canzoni considerando tutto il dataset.

Le diverse certificazioni come "Disco d'oro" e "Disco di platino" vengono rilasciate considerando il numero di streaming oltre che alle vendite solo da pochi anni. Inoltre con il passare del tempo i trend musicali cambiano, un fattore determinante per fare diventare una canzone di successo.

Con questa giustificazione si ritiene che sia meglio considerare solo i brani musicali dopo un certo anno di uscita. Prendiamo quindi in considerazione solo le canzoni dopo il 2005.

La distribuzione delle canzoni dal 2005 in poi è rappresentata in Figure 2.2. Viene di seguito mostrato il boxplot delle canzoni dopo quella data distinguendo tra classe positiva e negativa, così da vedere se esistono differenze.

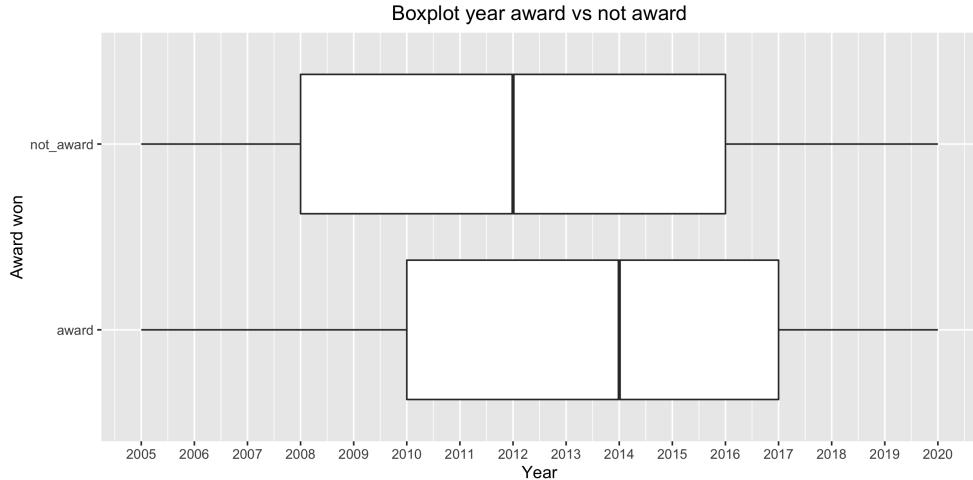


Figure 2.3: Boxplot anno di rilascio, distinguendo tra le classi.

Popolarità

Un altro aspetto che viene analizzato a parte, riguardo il quale è necessario fare ulteriori assunzioni, è il campo popolarità.

Il dataset fornito da Spotify contiene un campo "popularity". Tuttavia come spiegato nella subsection 1.1.2, questo campo non viene utilizzato per etichettare una canzone come di successo, si utilizza invece l'informazione delle certificazioni vinte da una canzone (ottenute da wikipedia).

Il campo "popularity" non verrà usato nella fase di training dei modelli, proprio perché è un dato che non si conosce a priori nel momento in cui un singolo esce, ed è chiaramente influente nel determinare se una canzone vincerà o meno una certificazione e quindi se viene considerata di successo.

L'informazione sulla popolarità viene calcolata sul numero di streaming dopo un determinato lasso di tempo. Stimare questo valore, dal momento che non è conosciuto fin da subito, è di fatto un problema di regressione ed è molto simile al task di classificazione preso in esame, pertanto il campo viene scartato.

Anche se il campo non viene effettivamente utilizzato, è interessante analizzare la distribuzione dei valori di popolarità delle canzoni nel dataset. Inoltre assumiamo che per il problema trattato, si vogliano classificare delle canzoni che non sono completamente sconosciute. Sarebbe infatti irrealistico pensare che un singolo musicale del tutto sconosciuto vinca dal nulla una certificazione, risultando come brano di successo. Per questo motivo non si tratta di una assunzione molto restrittiva.

Si assume di voler utilizzare questi modelli per classificare brani che iniziano ad essere un minimo conosciuti o hanno almeno il potenziale di diventare popolari. Si noti come un brano di poco popolare non implica assolutamente che questo vinca un disco d'oro o di platino.

Di seguito viene analizzata la distribuzione della popolarità delle canzoni nel dataset.

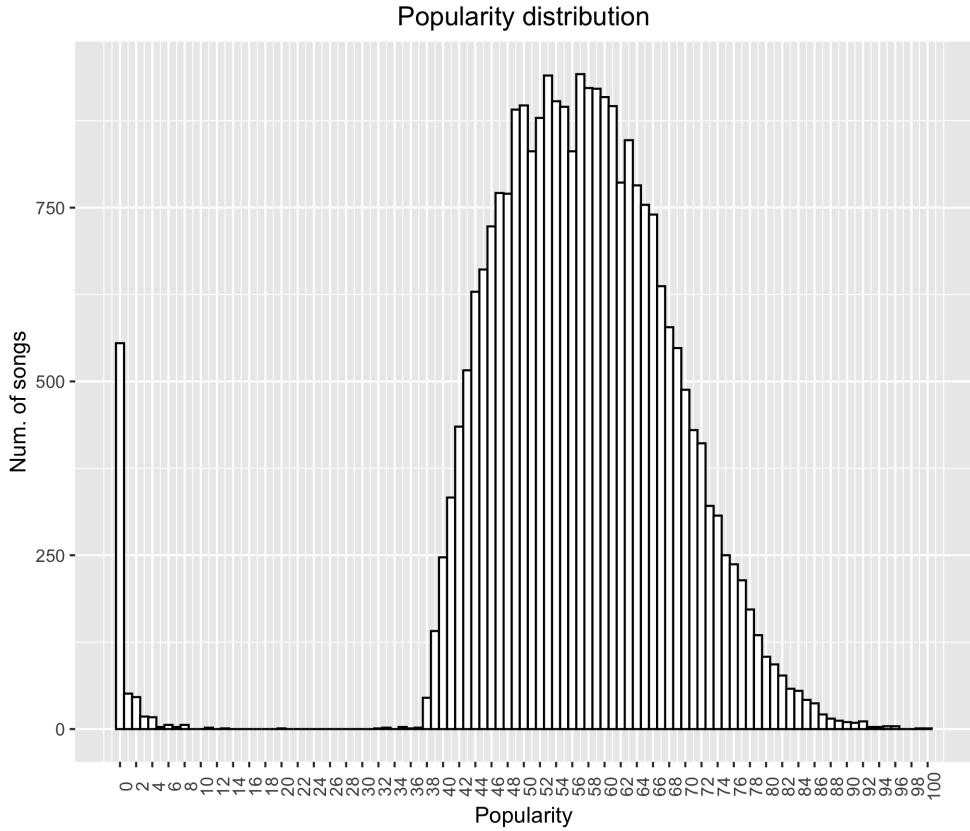


Figure 2.4: Distribuzione popolarità brani musicali.

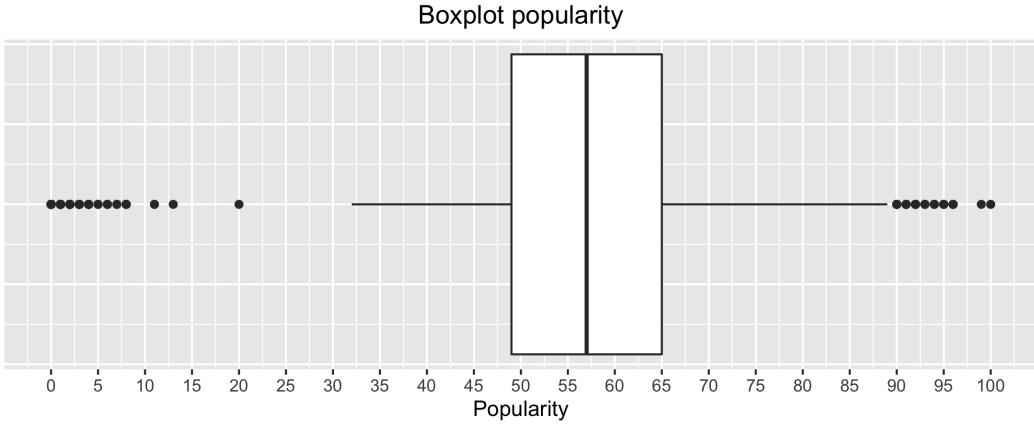


Figure 2.5: Boxplot popolarità dei brani musicali.

Analizzando la distribuzione dei valori della popolarità si considerano il primo, secondo e terzo quartile, rispettivamente: $q_{1/4} = 49$; $q_{2/4} = 57$; $q_{3/4} = 65$. Inoltre i valori di media e deviazione standard sono: $\mu = 57.83$; $\sigma = 10.12$.

Analizzando la distribuzione dei valori della popolarità viene scelto 25 come threshold, valore 3 volte più estremo della deviazione standard. Si considerano quindi solo i singoli con il valore "popularity" maggiore del threshold. In questo modo il dataset rispecchia l'assunzione

sopra spiegata riguardo la popolarità minima, tuttavia si scartano solo i valori davvero estremi per non rendere questa assunzione troppo restrittiva.

Dopo queste operazioni di riduzione del numero di istanze del dataset in base alle soglie, il numero di istanze diventa 26134.

2.4.2 Creazione dataset bilanciato

Dopo aver ridotto il dataset in base alle diverse soglie si vuole vedere il numero di istanze appartenente alla classe positiva e negativa. Per classe positiva si intendono i brani che hanno vinto un premio e sono quindi di successo, viceversa per la classe negativa.

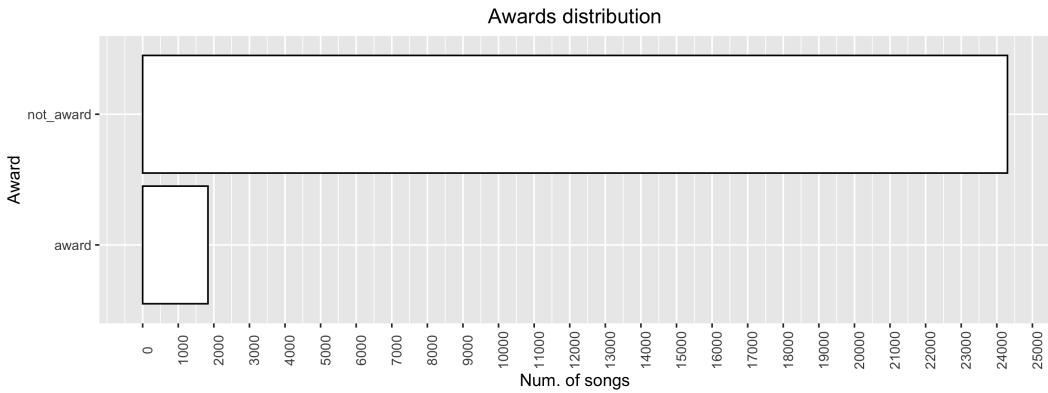


Figure 2.6: Istanze positive e negative.

Undersampling

Dalla Figura 2.6 emerge che il dataset è sbilanciato, il rapporto tra classe negativa e positiva è circa 14 : 1. Dal momento che questo può creare problemi in fasi di training, si sceglie di costruire un dataset bilanciato a partire da tutte le istanze. Come strategia per la costruzione di questo nuovo dataset vengono scelte casualmente un numero di istanze negative pari al numero di istanze positive.

2.4.3 Preprocessing

Per quanto riguarda le variabili numeriche queste vengono standardizzare, ovvero per ogni covariata si trasformano gli individui in modo da ottenere dei valori tali che $\mu = 0$; $\sigma = 1$. Questa è sicuramente una best practice ma è cruciale per quanto riguarda la PCA, tecnica che viene utilizzata e spiegata nella sezione subsection 2.4.7.

Inoltre nella fase di preprocessing tutti gli artisti vengono trasformati in lowercase. Questo è necessario nel momento in cui si vuole tenere traccia degli artisti nelle canzoni con una rappresentazione one hot encoding (subsection 2.5.2).

2.4.4 Distribuzione dei valori

Variabili numeriche

Di seguito viene mostrata la distribuzione delle covariate, distinguendo tra singoli che hanno vinto un premio (Classe positiva) e quelli che non hanno vinto un premio (Classe negativa).

Dal pairplot della Figure 2.7 possiamo notare come non esiste una netta distinzione nei dati tra brani di successo e brani non di successo. Questo sarà sicuramente un problema in fase

di classificazione e potrebbe potrare a basse performance dei modelli. Riteniamo quindi che sia necessario considerare altre features per ben distinguere brani musicali di successo da quelli non di successo.

Il campo "popularity" ben distingue le due classi, tuttavia come già discusso in section 2.4.1, questa covariata non verrà utilizzata per la creazione dei modelli.

Inoltre esiste correlazione tra alcune covariate, come ad esempio tra "energy" e "loudness", questo rispecchia intuitivamente l'idea di come una canzone più energica è anche più rumorosa. La correlazione tra le variabili viene meglio analizzata in subsection 2.4.6.

CHAPTER 2. DATASET

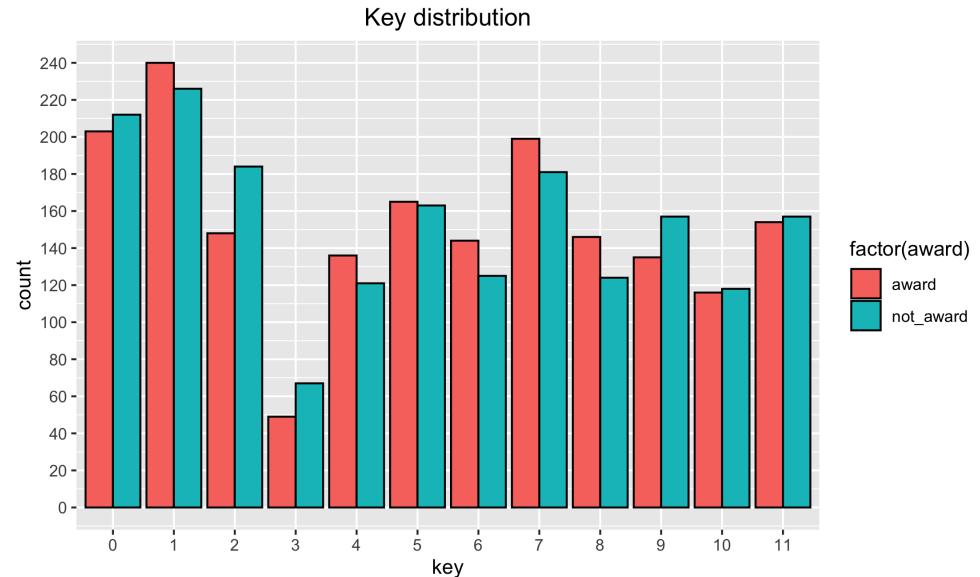
15



Figure 2.7: Pairplot delle covariate.

Variabili categoriche

Di seguito si esplorano le variabili categoriche del dataset, distinguendo tra classe positiva e negativa.



(a) Variabile key.

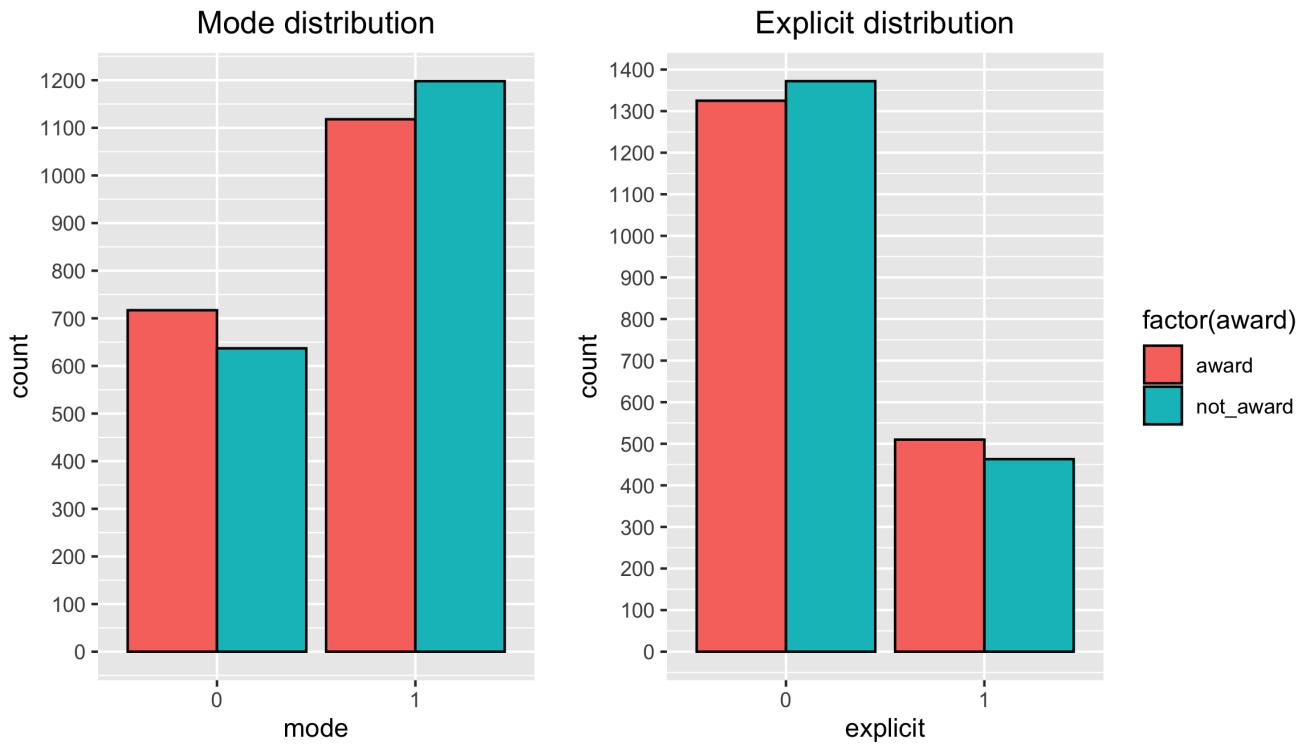


Figure 2.8: Distribuzione delle variabili categoriche.

2.4.5 Artisti nelle canzoni

Un'altra caratteristica che si ritiene importante per riconoscere una canzone come di successo, è quali artisti sono presenti in una canzone.

Frequenza artisti

Vine ora considerato il numero di occorrenze di ogni artista tra tutte le canzoni, si analizza quindi la frequenza delle occorrenze.

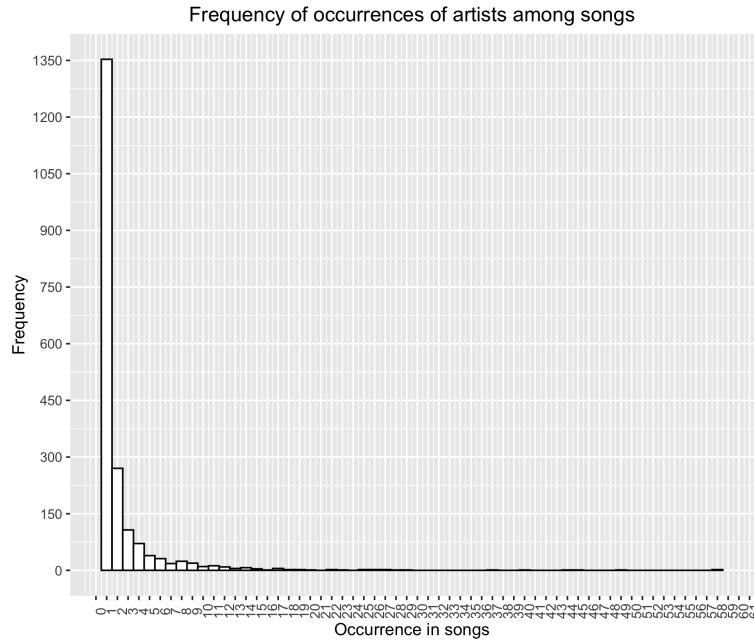


Figure 2.9: Frequenza delle occorenze degli artisti nelle canzoni.

Come possiamo notare da questo grafico, la maggior parte degli artisti hanno fatto solo una canzone. Questo aspetto verrà tenuto in considerazione per la rappresentazione one hot encoding degli artisti, come spiegato in subsection 2.5.2

Wordcloud

Si vuole ora analizzare se esiste una differenza tra artisti che hanno realizzato brani di successo rispetto a quelli non di successo. Viene prima di tutto mostrato il wordcloud degli artisti tra tutte le canzoni, senza fare distinzione tra classe positiva e negativa.

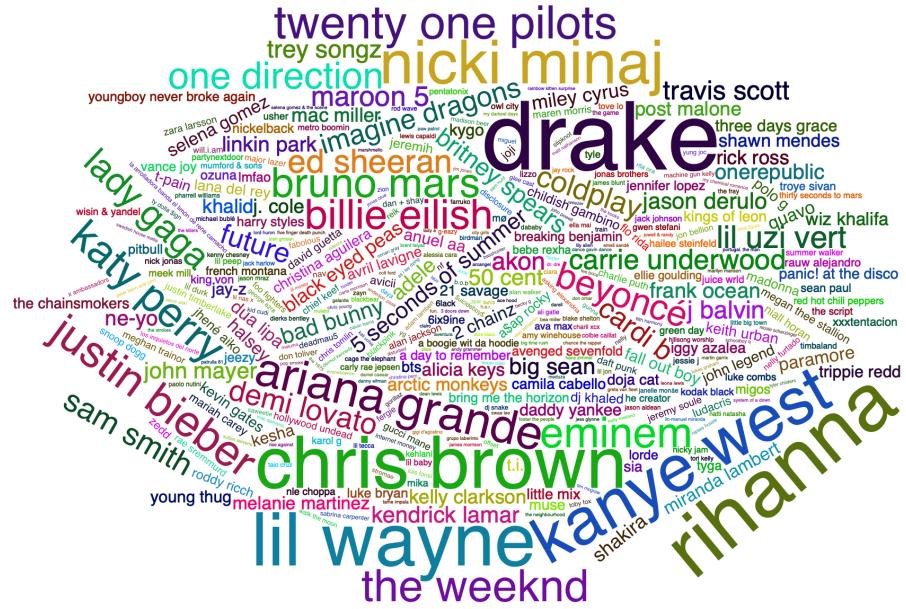


Figure 2.10: Wordcloud artisti considerando entrambe le classi.

Viene quindi mostrato il worcloud degli artisti andando a considerare solo le canzoni di successo:

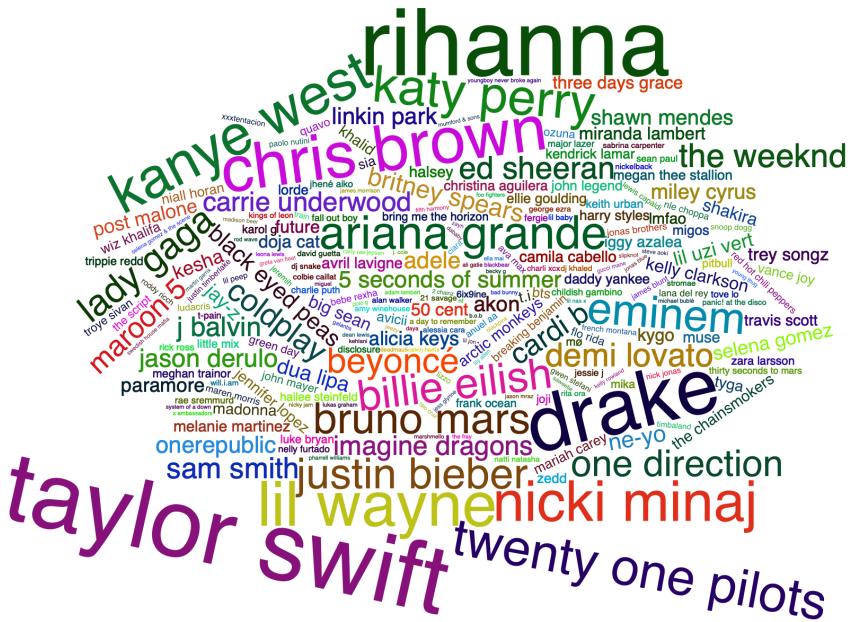


Figure 2.11: Wordcloud artisti considerando solo la classe positiva.

Analogamente il worcloud degli artisti considerando solo le canzoni non di successo:

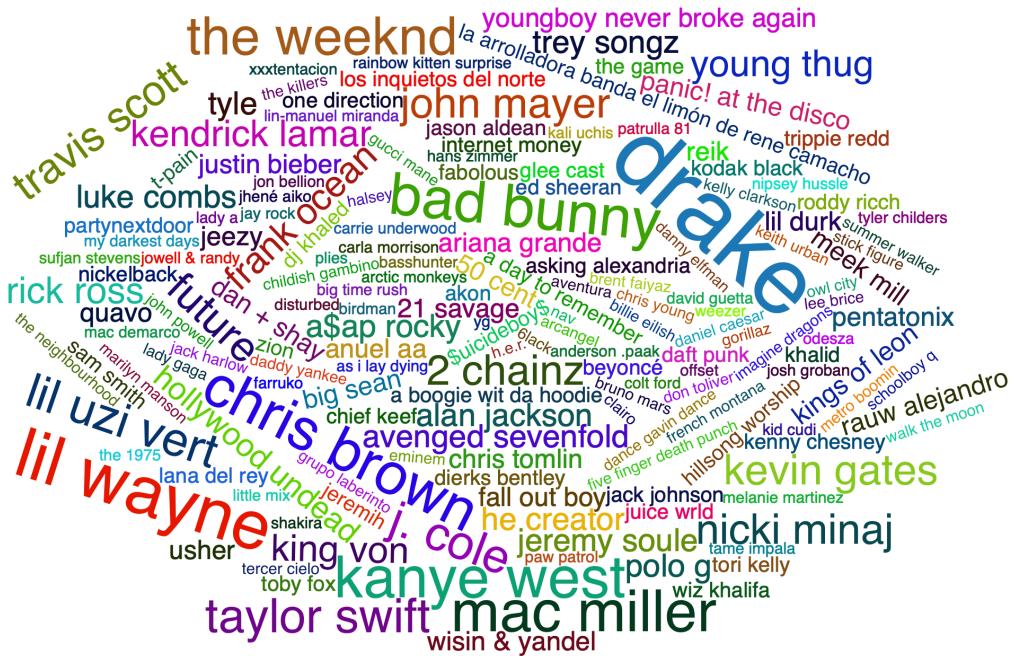


Figure 2.12: Wordcloud artisti considerando solo la classe negativo.

Dai grafici emerge che molti artisti compaiono sia nella classe positiva che in quella negativa. Tuttavia ci sono alcuni artisti e.g. "Taylor Swift" o "Rihanna" che spiccano nella classe di successo mentre sono meno frequenti nei brani non di successo. A fronte di questa analisi riteniamo che considerare gli artisti all'interno di un brano possa essere utile per distinguere le due classi. Viene quindi utilizzata una rappresentazione one hot encoding (spiegato in subsection 2.5.2) per utilizzare questa informazione durante la fase di training dei modelli.

2.4.6 Correlazione tra features

Si considera ora la correlazione tra le features numeriche. A questo scopo calcoliamo la matrice di correlazione e la rappresentiamo qui sotto per mezzo di una heatmap:

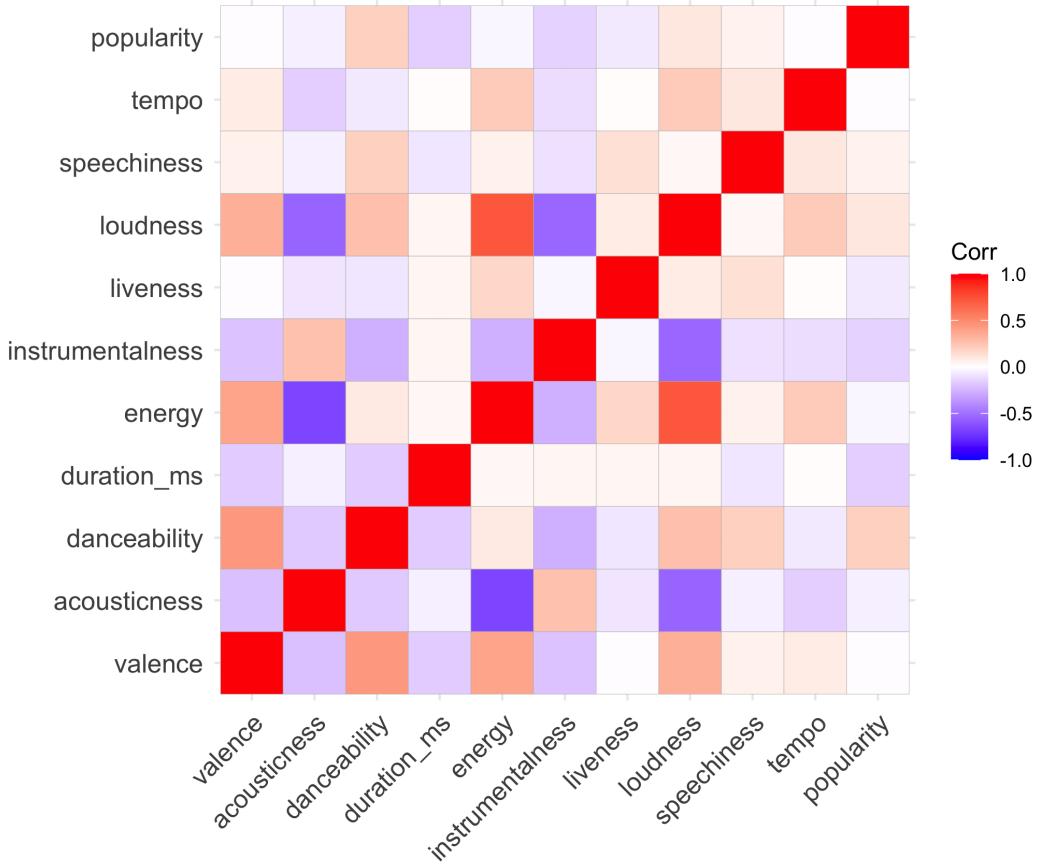


Figure 2.13: Matrice di correlazione.

Da questa immagine si può notare come alcune covariate, ad esempio "energy" e "loudness", sono correlate positivamente, mentre altre covariate come "acousticness" e "energy" oppure "acousticness" e "loudness" sono correlate negativamente. Questo rispecchia il senso comune di energia di una canzone, e ci si aspetta che al crescere di questa aumenta anche la loudness di un brano.

2.4.7 Principal component analysis

Dal momento che esiste correlazione tra le covariate, viene utilizzata la tecnica PCA con lo scopo di ridurre la dimensione del dataset spiegando la maggior parte della varianza.

La PCA viene effettuata solo sulle features numeriche, si noti inoltre come il dataset è stato precedentemente standardizzato, in questo modo i valori numerici hanno media 0 e varianza 1. Questa è una condizione necessaria per applicare correttamente la tecnica PCA.

A partire dalla matrice di covarianza viene effettuata la decomposizione agli autovalori, si ottengono quindi gli autovettori con gli autovalori associati. Ordinando gli autovalori in ordine decrescente viene mostrata la varianza spiegata da ogni componente principale, oltre che alla varianza spiegata cumulata.

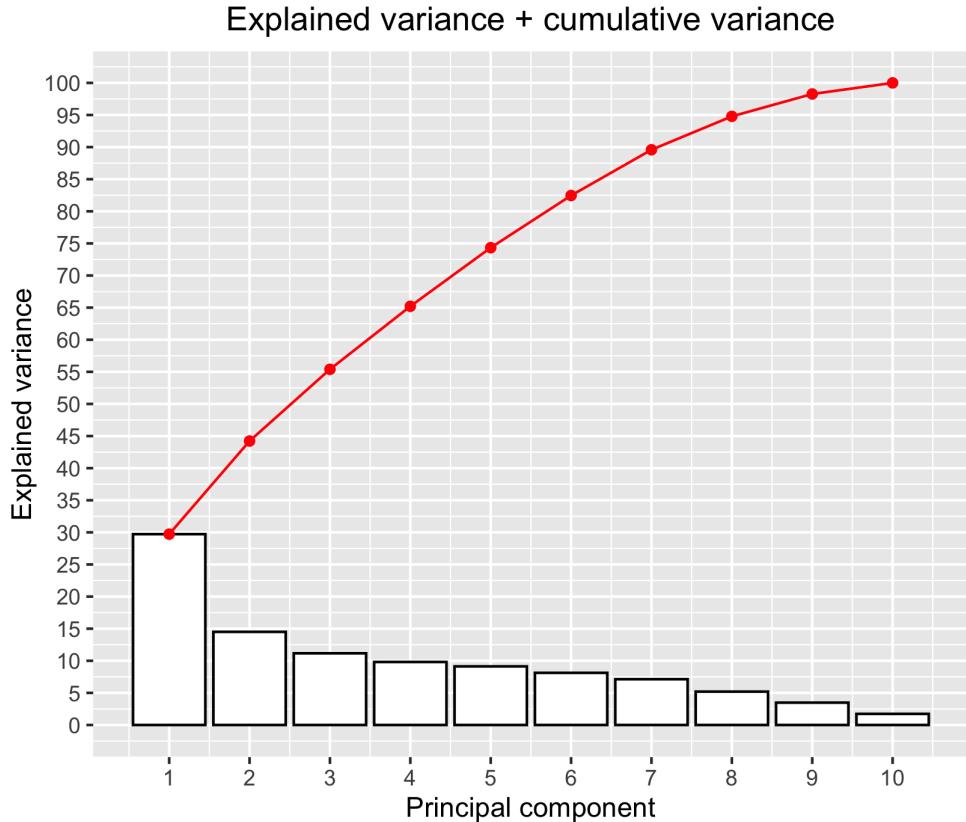


Figure 2.14: Varianza spiegata dalle componenti principali.

In tabella viene riportata la varianza spiegata cumulata:

PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
29.72	44.22	55.39	65.20	74.33	82.45	89.58	94.78	98.26	100.00

Sulla base di questi dati è possibile notare come le prime 7 componenti principali spieghino circa il 90% della varianza totale. Per questo motivo si sceglie di proiettare il dataset dallo spazio originale nel nuovo sottospazio trovato con la PCA. Questo nuovo spazio ha dimensione pari a 7, ovvero il numero di componenti principali che si è scelto di utilizzare.

2.5 Scelta delle features

In questa sezione viene spiegata e giustificata la scelta di ogni feature.

2.5.1 Variabili scartate

Come si può notare dalla Table 2.1 non tutti le features vengono utilizzate. Le variabili scartate sono le seguenti:

- **id**: ID assegnato da spotify a ogni brano musicale. Questo campo non è rilevante per distinguere le due classi
- **name**: Titolo del brano musicale. Questa variabile non è utilizzata. Si potrebbero usare tecniche di NLP per tenere conto del sentimento del titolo brano e usare questa feature aggiuntiva.

- **release_data:** L'anno di rilascio del brano viene utilizzato solo per la fase iniziale per scartare i brani troppo vecchi. Questa covariata non viene utilizzata per il training in quanto si ritiene che non sia influente per distinguere le due classi (Figure 2.3). Inoltre si vogliono costruire modelli che siano indipendenti dalla data di rilascio.
- **year:** Campo uguale a "release_date". Nel dataset questa informazione è duplicata.
- **popularity:** Questo campo non viene utilizzato per i motivi discussi in section 2.4.1

2.5.2 Rappresentazione one hot encoding degli artisti

Dopo aver analizzato i wordcloud degli artisti emerge che questa informazione è utile per distinguere le due classi. Si procede quindi costruendo l'insieme degli artisti tra tutte le canzoni. Per ogni artista viene tenuto traccia del numero di brani musicali in cui l'artista compare.

La frequenza è necessaria perchè si vogliono tenere solo gli artisti che hanno una frequenza maggiore o uguale a 2. Questa operazione viene effettuata per non contare gli artisti completamente sconosciuti, in quanto si ritengono non influenti ai fine della classificazione. Inoltre a causa dell'undersampling è possibile avere diverse canzoni cantate da artisti "sconosciuti" e non avendo scelto una particolare strategia per l'undersampling adottiamo a questo punto l'utilizzo di un threshold.

Si procede quindi costruendo una matrice dove le colonne rappresentano ogni artista nell'insieme degli artisti, le righe invece gli individui nel dataset. L'entrata i, j della matrice può contenere il valore 1 o 0. Nel caso in cui il j -esimo artista ha cantato nella i -esima canzone allora l'entrata della matrice ha il valore 1, 0 altrimenti.

La matrice costruita costituisce la rappresentazione degli artisti nelle canzoni.

2.5.3 Variabili categoriche

Tutte le variabili categoriche vengono utilizzate ai fini della classificazione in quanto dalla descrizione della Table 2.1 risultano essere importanti nella distinzione delle classi.

Variabile key

La variabile key è di tipo factor i cui valori sono numeri da 1 a 11. Questa covariata indica la scala musicale utilizzata nella canzone. Dal momento che esiste una relazione d'ordine totale tra le note musicali, infatti si parla di scala musicale, per questa variabile viene effettuata l'operazione di cast a intero. Come secondo passo i valori vengono scalati in modo tale da essere compresi tra 0 e 1.

Variabile mode

La variabile mode è binaria, viene utilizzata impostando a 1 se vero 0 altrimenti.

Variabile explicit

Per la variabile explicit si procede analogamente a come fatto per la covariata mode.

2.5.4 Coordinate PCA

Le variabili numeriche non vengono direttamente usate per il training, piuttosto si utilizza il nuovo spazio ottenuto dalla PCA. Questo nuovo spazio ha dimensione inferiore rispetto al numero di variabili numeriche iniziale, ed è stato costruito proiettando gli individui del dataset di partenza nel nuovo spazio che ha come basi le 7 componenti principali.

2.5.5 Riassunto feature finali utilizzate

FEATURE	# COVARIATE
One-hot-encoding artisti	655
Componenti principali	7
Variabili categoriche	3
Label	1
	666

Table 2.3: Tabella riassuntiva features utilizzate.

Chapter 3

Campagna sperimentale

3.1 Approccio

3.1.1 10-folds cross validation

Per la fase di training dei modelli viene utilizzata la tecnica k-folds cross validation. Si procede quindi creando una partizione del dataset iniziale, dove ogni sottoinsieme, ovvero un fold, ha circa lo stesso numero di istanze ed è più o meno bilanciato tra la classe positiva e negativa.

Per gli esperimenti effettuati, è stato scelto $k = 10$, ovvero il dataset viene diviso casualmente in 10 parti, utilizzando ad ogni iterazione una porzione di dati diversa per il training-set e test-set. Avendo scelto questo valore di k , la proporzione tra i due insiemi sarà sempre 90% per il training-set e 10% per il test-set.

Per ogni modello si vogliono testare diversi iperparametri (subsection 3.1.2), pertanto ad ogni iterazione della 10-folds, fissato un fold per il training \hat{F} , si esegue un'altra 10-fold cross validation più interna su \hat{F} (90% del dataset totale), così da poter scegliere l'iperparametro migliore.

Procedendo in questo modo l'ottimizzazione degli iperparametri viene effettuata sfruttando solo il training set e il test set non viene utilizzato né per il training né per la scelta degli iperparametri ottimali.

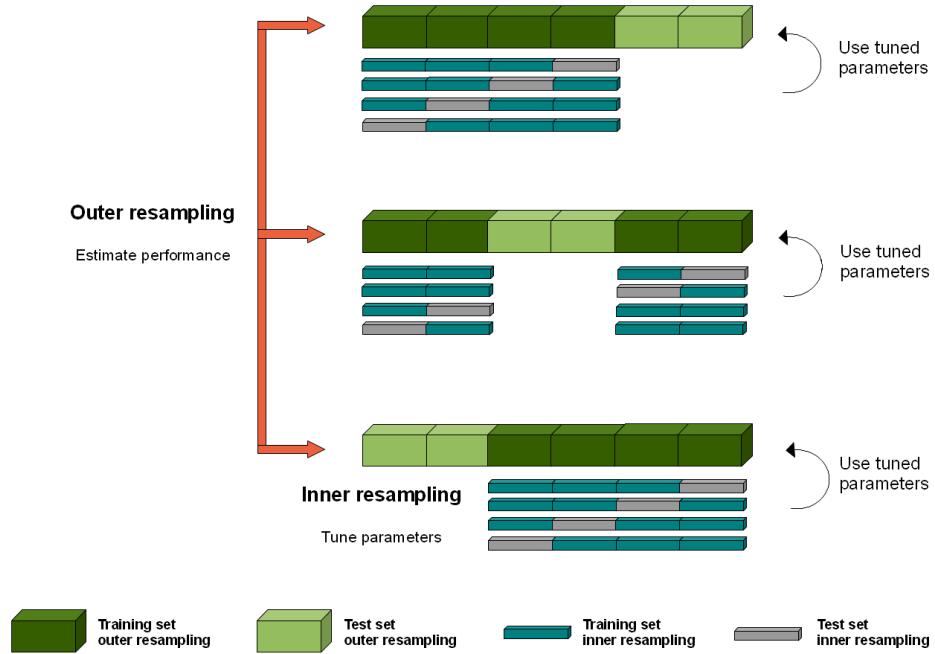


Figure 3.1: Nested 3-folds cross validation.

Una nota importante è che i 10 folds vengono generati in modo casuale, tuttavia per ogni modello di cui si vuole fare il training, questa generazione casuale viene effettuata a per mezzo dello stesso seed. Questo garantisce di avere folds generati casualmente ma in modo identico per ogni modello, in questo modo si ottengono dei risultati statisticamente validi.

3.1.2 Model selection

Come abbiamo sopra discusso, per ogni modello viene effettuato il tuning degli iperparametri usando la tecnica 10-folds cross validation.

Grid search

Per determinare gli iperparametri ottimali, vengono scelti un insieme di parametri da utilizzare per la fase di training dei modelli. Per ogni modello si considerano tutte le combinazioni dei valori scelti per i parametri.

SVM - RBF kernel		SVM - Linear kernel	Decision tree
<i>C</i>	<i>sigma</i>	<i>C</i>	<i>cp</i>
0.001	0.00001	0.000001	0.01580381
0.01	0.0001	0.00001	0.03487738
0.1	0.001	0.0001	0.17983651
1	0.01	0.001	
10	0.1	0.01	
		0.1	
		1	
		10	

Table 3.1: Tabella iperparametri utilizzati per grid search.

I valori in grassetto nella tabella sono gli iperparametri ottimali. Per quanto riguarda il parametro complexity parameter di decision tree, l'insieme dei valori non è stato scelto manualmente.

3.2 Misure di performance

Le misure di performance sono state ottenute combinando i risultati ottenuti dalla 10-fold cross validation per ogni fold sia per la classe positiva che per la classe negativa facendo la macro average delle misure cercate. Le misure considerate sono:

Accuracy L'accuracy è definita come il numero di veri positivi e veri negativi rapportati al numero di esempi totali.

Precision La precision è definita come il numero di veri positivi rapportati al numero totale di predizioni positive.

Recall La misura di recall è definita come il numero di veri positivi rapportati al numero di veri positivi e falsi negativi.

F-measure La F-Measure è definita come la media armonica di precision e sensitivity.

3.3 Support Vector Machine

Il primo algoritmo scelto è support vector machine. Nello specifico sono stati utilizzati due diversi kernel per poi confrontarne le prestazioni. I kernel utilizzati sono:

- Lineare
- Radial basis function

3.4 Decision Tree

3.4.1 Scelta del modello

Come secondo algoritmo di apprendimento è stato scelto un albero di decisione (decision tree/classification tree). Questo perché anche con un dataset relativamente ampio permette il training in un tempo ragionevole rispetto alla potenza computazionale in nostro possesso. Inoltre un albero di decisione è facilmente interpretabile, infatti permette di analizzare su quale base vengono prese le decisioni.

3.4.2 Decision Tree Plot

Il decision tree ottenuto è molto semplice, questo è dovuto al valore ottimale del complexity parameter. Si noti come utilizzando questo algoritmo la feature più significativa, ovvero dove decision tree effettua lo split, risulta essere la componente principale ottenuta con la tecnica di principal component analysis.

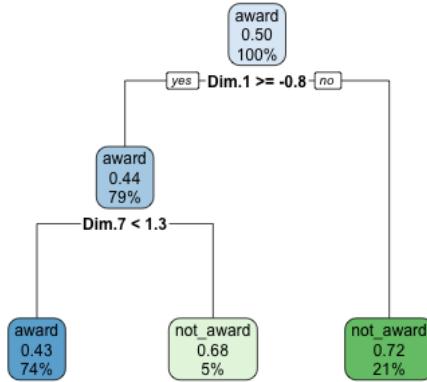


Figure 3.2: Decision tree plot.

3.4.3 Complexity Parameter

Il complexity parameter influisce direttamente sulla complessità dell'albero. In questo senso per complessità si fa riferimento alla VC-dimension, ciò si riflette direttamente sulla struttura dell'albero. Nel caso preso in esame vengono considerati differenti valori per il complexity parameter (elencati in Table ??).

Nell'immagine seguente vengono mostrate le differenti performance dell'albero, utilizzando la metrica ROC, al variare del complexity parameter.

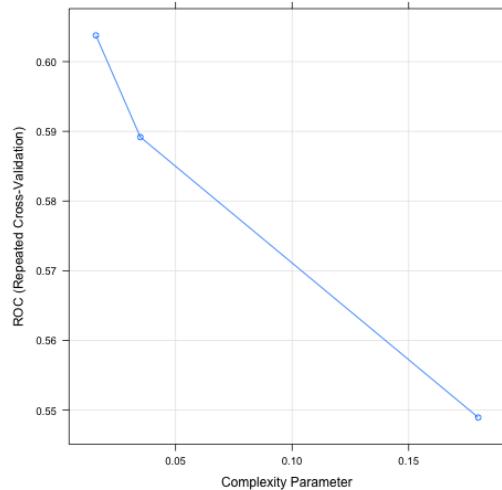


Figure 3.3: Complexity parameter plot.

3.4.4 Risultati esperimenti

Performance classe positiva vs negativa

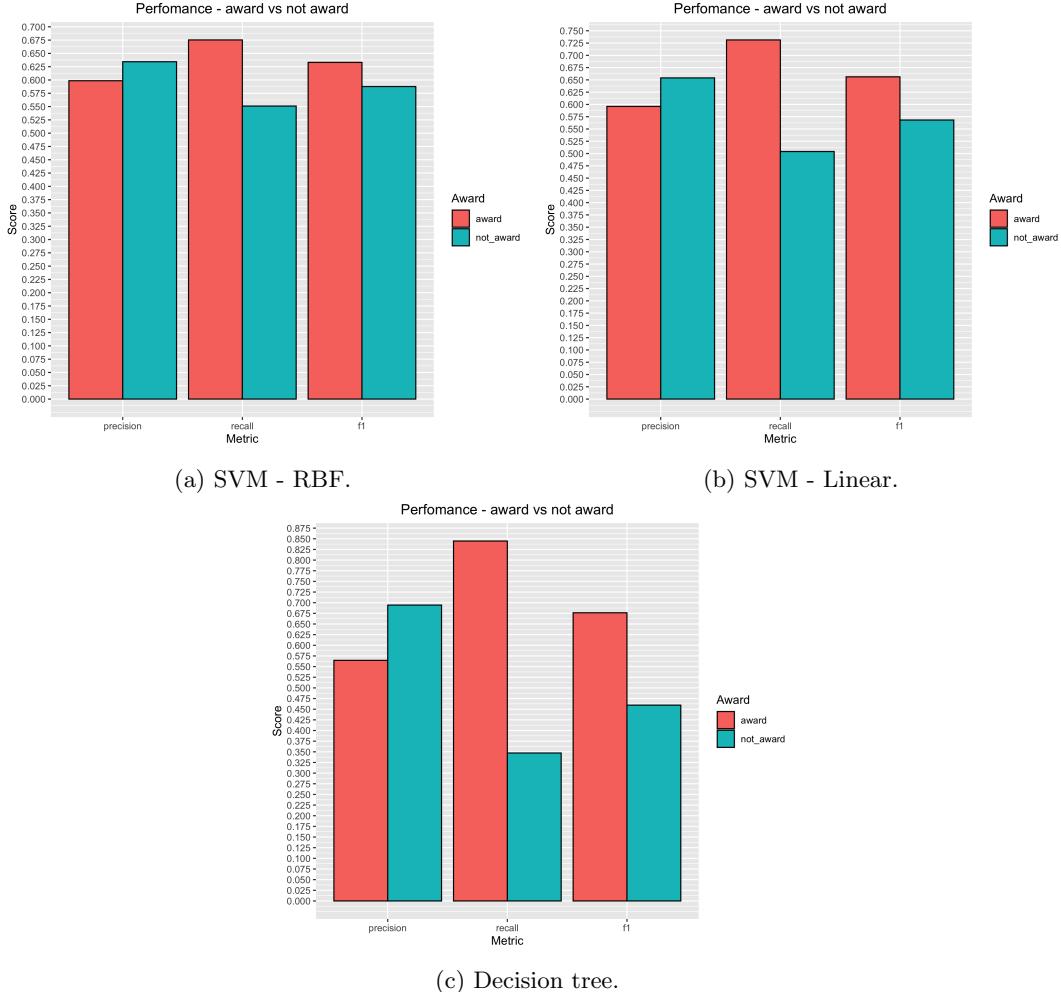


Figure 3.4: Performance dei modelli distinguendo tra classe positiva e negativa.

Dal grafico emerge che i modelli SVM sono piuttosto simili tra loro. Per quanto riguarda decision tree è interessante analizzare la metrica recall. Infatti si ottengono performance molto alte per quanto riguarda la classe positiva, ma piuttosto basse per la recall della classe negativa. Questo vuol dire che il modello tenderà a classificare la maggior parte delle istanze come positive, ottenendo quindi un valore di recall alto per la classe positiva. In questo modo si avranno molti falsi positivi, tuttavia i falsi negativi saranno pochi. Questo comportamento potrebbe essere utile nel caso in cui fossimo disposti a classificare brani che non sono di successo come di successo, a patto di non "perderne" nessuno.

Matrici di confusione

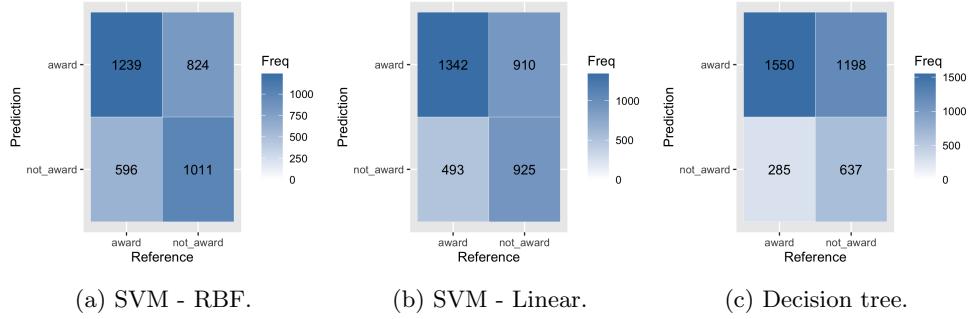


Figure 3.5: Matrici di confusione dei modelli.

Performance macro average

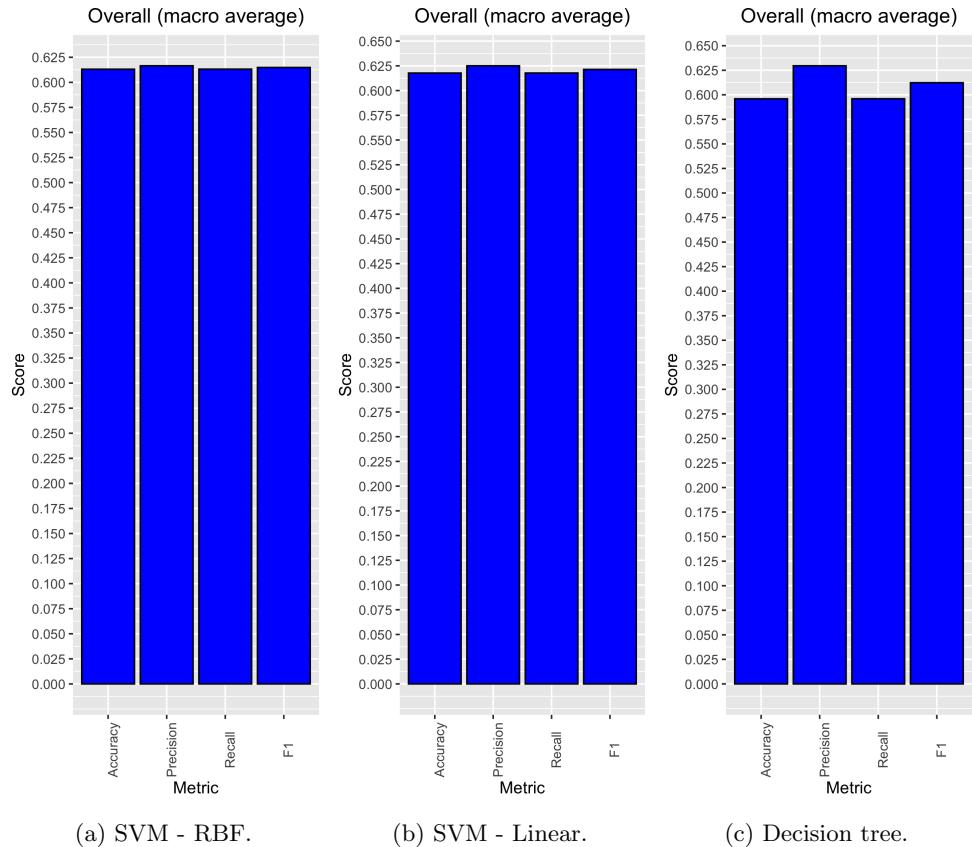


Figure 3.6: Performance dei modelli (macro average).

Come si può notare dai grafici, le performance sono piuttosto simili tra i tre modelli.

Curve ROC e AUC

Viene ora analizzata la curva ROC per la classe positiva (award) e la relativa area sotto la curva.

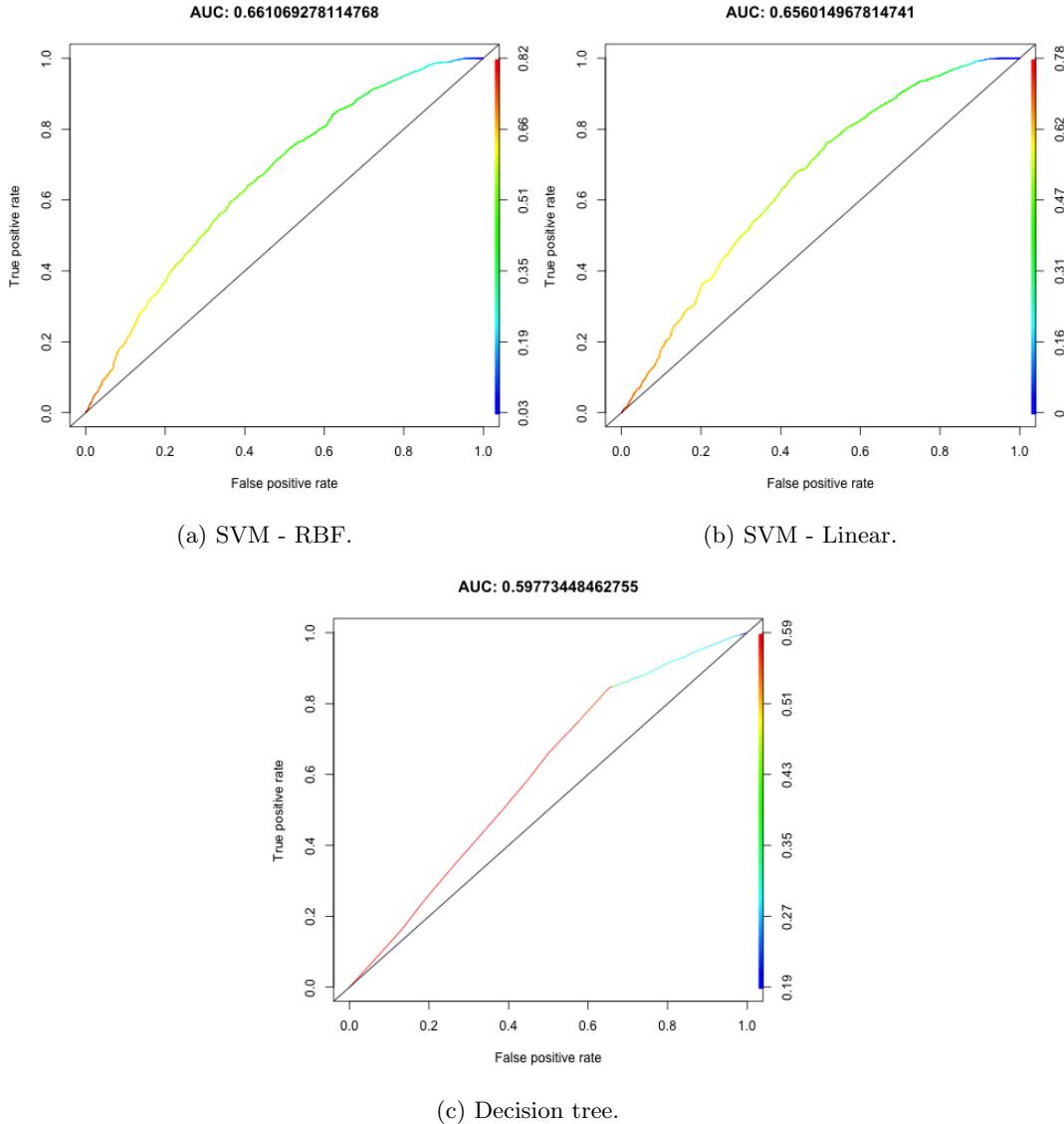


Figure 3.7: Curve ROC e AUC dei modelli.

Dalle curve ROC e la misura di AUC è possibile rendersi conto che i classificatori non sono molto efficaci, un valore di AUC di 0.5 significa che il nostro classificatore ha le stesse capacità predittive di un classificatore che classifica in maniera casuale. Il classificatore migliore risulta essere la support vector machine con kernel radiale mentre il peggiore risulta essere il decision tree ottenuto tramite rpart.

Un altro aspetto interessante è il fatto che la curva ROC mostra il rapporto dei falsi positivi e il rapporto dei veri positivi al variare del cutoff scelto. Questo permette di scegliere un cutoff in modo tale da bilanciare i falsi positivi e veri negativi secondo le esigenze di dominio.

Viene qui mostrata l'accuracy dei modelli al variare del cutoff:

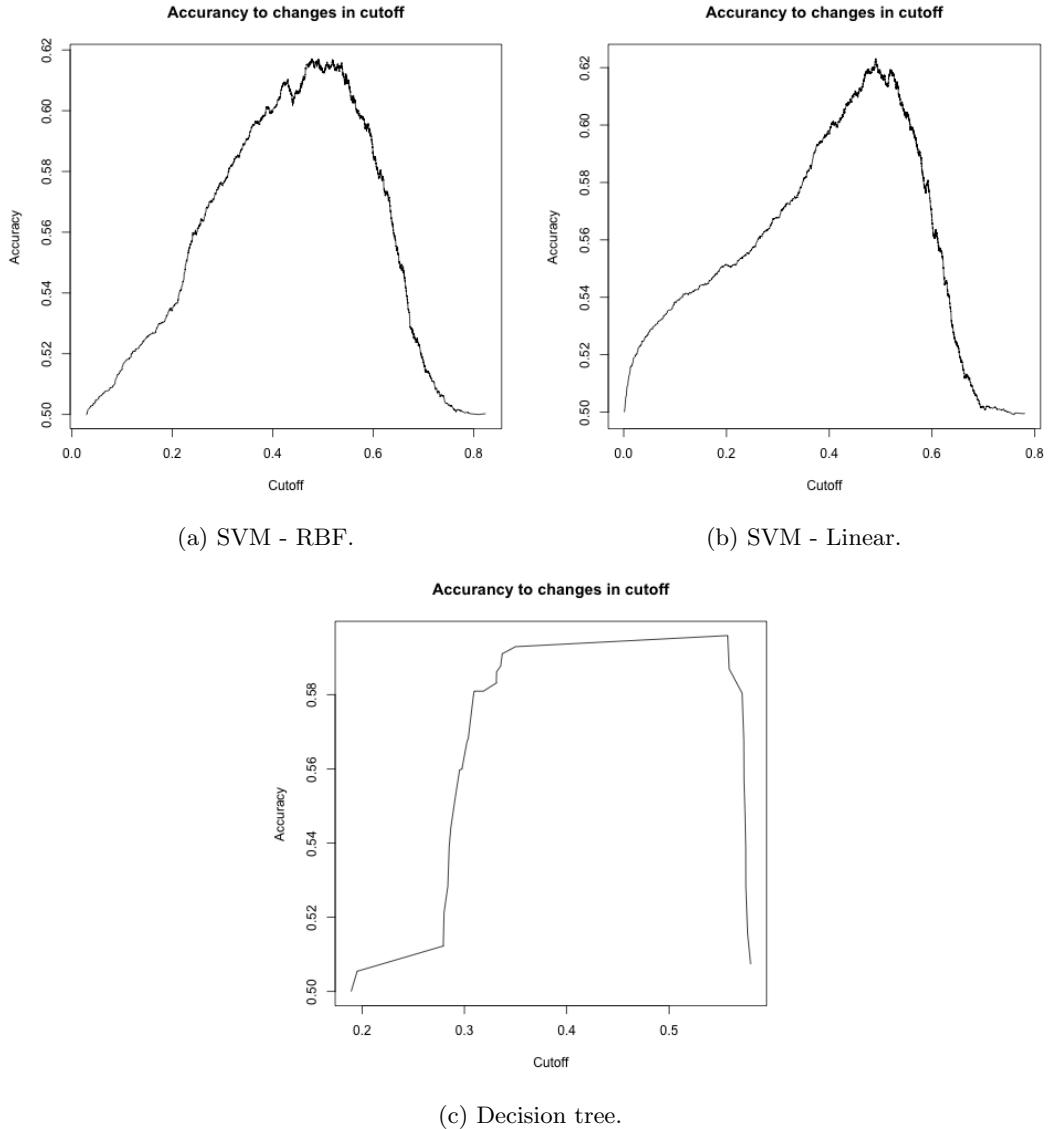


Figure 3.8: Accuracy dei modelli al variare del cutoff.

Tabelle riassuntive performance

	Precision+	Recall+	F1+	Precision-	Recall-	F1-
SVM - RBF	0.59860	0.67530	0.63313	0.63433	0.55098	0.58766
SVM - Linear	0.59600	0.73135	0.65625	0.65406	0.50413	0.56872
Decision tree	0.56467	0.84469	0.67633	0.69437	0.34724	0.45957

Table 3.2: Confronto performance dei modelli distinguendo tra classe positiva e negativa.

	Accuracy	Precision	Recall	F1	AUC
SVM - RBF	0.61309	0.61640	0.61139	0.61800	0.66106
SVM - Linear	0.61771	0.62503	0.61774	0.62136	0.65601
Decision tree	0.59592	0.62921	0.59571	0.61285	0.59734

Table 3.3: Confronto performance dei modelli (macro average).

3.5 Modelli a confronto

Per confrontare i modelli viene utilizzata la tecnica 10-folds cross validation con 3 ripetizioni. Questo vuol dire ripetere la 10-folds cross validation per 3 volte generando folds differenti. Con questa tecnica si ottengono stime di performance più accurate, a discapito dei tempi di training.

Un aspetto importante è che i folds vengono generati in modo identico per il training di ogni modello. In questo modo è possibile mettere a confronto le performance ottenute da ogni modelli in maniera statisticamente valida.

3.5.1 Confronto secondo metrica ROC

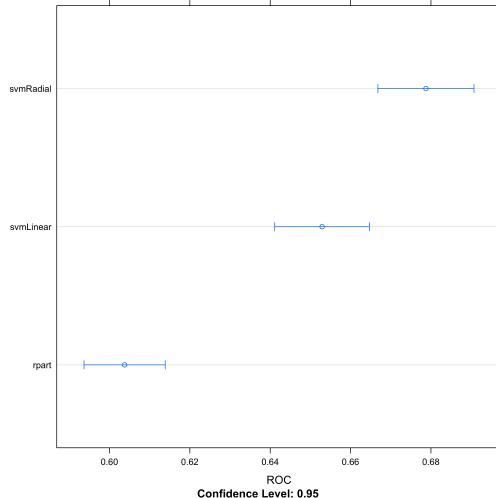


Figure 3.9: Intervalli di confidenza metrica ROC.

Dall'immagine sopra possiamo notare come fissato un intervallo di confidenza uguale a 0.95, l'intersezione tra i valori è vuota per ogni coppia di modelli. Possiamo quindi concludere con una confidenza al 95% che, basandosi sulla metrica ROC, il modello SVM con kernel radial basis function ottiene le performance migliori. Questo si paga in termini di complessità del modello e tempo per il training.

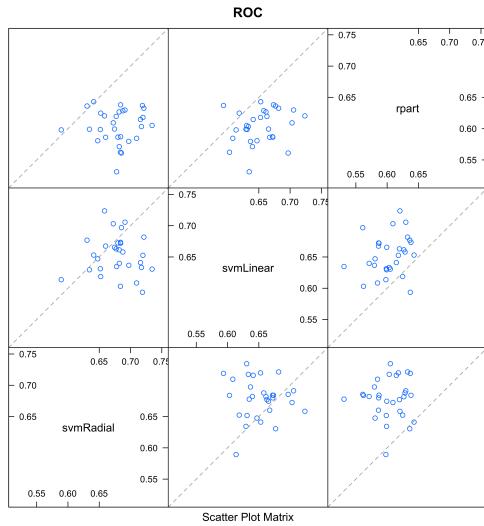


Figure 3.10: Scatter plot per metrica ROC.

Come spiegato precedentemente, il confronto viene effettuato eseguendo per 3 volte una 10-folds cross validation. Quindi alla fine si ottengono 30 stime di performance per ogni modello, ognuna delle quali è il risultato del training di uno specifico fold. In Figure 3.9 vengono mostrate le stime di performance utilizzando ognuno dei 30 folds. In modo coerente a quanto visto in Figure 3.10 notiamo che SVM con kernel rbf è decisamente migliore di decision tree, infatti praticamente per ogni fold ottiene performance migliori. Anche per quanto riguarda il confronto tra SVM con kernel rbf e lineare, il modello con kernel rbf ottiene performance migliori sulla maggior parte dei folds. Nel caso in cui avessimo avuto due modelli con i 30 folds distribuiti sulla bisetrice di un quadrante allora i modelli sarebbero stati identici.

3.5.2 Confronto Timings

Di seguito vengono mostrati i tempi di training per i modelli (espressi in secondi). La colonna Everything si riferisce al training con la tecnica 10-fold cross validation mentre final model indica il training del modello utilizzando tutto il dataset. Per la fase di training vengono sfruttati più cores in modo tale da ridurre i tempi di training.

	Everything	Final model
SVM - RBF kernel	1286.484	43.524
SVM - Linear kernel	666.136	34.730
Decision tree	16.049	0.884

Table 3.4: Tempi per il training dei modelli.

Il modello support vector machine risulta più lento per quanto riguarda i tempi di training, soprattutto se viene utilizzato il kernel radial basis function. Decision tree è quello più veloce.

Da questa tabella emerge chiaramente come le performance migliori si pagano in termini di complessità del modello e tempi di training.

Chapter 4

Conclusioni

Le performance migliori basandosi su metrica ROC si ottengono con SVM utilizzando kernel rbf. Secondo questa metrica i modelli SVM sono effettivamente migliori rispetto a decision tree. Anche se SVM con kernel rbf ha performance migliori rispetto a SVM con kernel lineare, di fatto le performance sono molto simili. Questo piccolo miglioramento si paga in termini di complessità del modello e tempo di training.

Nel complesso le performance dei modelli sono piuttosto basse. Riuscire a distinguere se un brano diventerà di successo oppure no è un'operazione non banale anche per gli umani. Anche per questo motivo le performance dei modelli sono relativamente basse. Tuttavia ci sono anche altri fattori che possono influire negativamente sulle performance.

Il dataset potrebbe essere ampliato per esempio aggiungendo il genere musicale di ogni brano. Potrebbe infatti esistere una correlazione statistica tra generi più popolari e la vittoria di un premio da parte di una canzone.

La probabilitá di successo di un singolo sono influenzate anche da fattori non presenti nel nostro dataset. Per esempio se una canzone viene utilizzata in un film o pubblicitá televisiva questo accrescerà la sua popolarità e probabilmente anche il suo numero di ascolti.

Inoltre sarebbe interessante utilizzare tecniche di NLP per estrarre informazioni riguardo la lyrics di un brano e valutare se un testo riguardante un particolare 'tema' (per esempio una canzone d'amore) la porta ad avere più successo rispetto ad altre.