Dopo aver preso visione dei dataset set su cui si andrà a lavorare si è deciso di iniziare con l’implementazione della soluzione.

Per quello che riguarda il punto:

**1) Explorative Data Analysis(EDA).**

Il punto **1.1** richiedeva di specificare il numero di artisti e di utenti presenti nella rete, per fare questo si sono contate le righe del dataset “artist.csv”, in modo da avere il numero di artisti presenti nella rete. Invece, per saper il numero di utenti presenti nella rete sono state calcolate le righe presenti nel file “user\_artist.csv” alla colonna (UserID) non considerando i valori ripetuti.

Il numero di artisti è: 17632 e il numero di utenti è: 1892

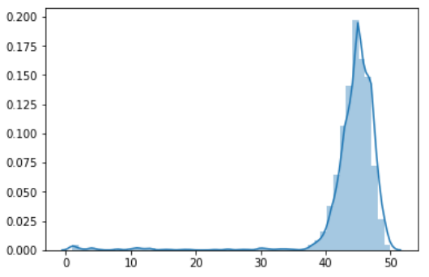
Il punto **1.2** richiedeva di trovare le coppie univoche di amici presenti nella rete, per fare questo si è preso in considerazione il file “user\_friend.csv” andando a creare un doppio ciclo for per verificare se l’amicizia risulta condivisa tra entrambi gli utenti.

Il numero di amici bi-direzionali è: 25434

Il punto **1.3** richiedeva di trovare qual è la media di amici per ogni utente, per fare questo ci si è basati su un array di appoggio andando a inserire in ogni cella il numero di amici di ogni utente e tento un array di appoggio per verificare se si era già preso in considerazione l’utente. Una volta riempito l’array contente il numero di amici si è fatta di tutti gli elementi presenti nell’array in modo da avere la media generale di amici di ogni utente.

La media di relazioni di amicizia per ogni utente è:13.44

Il punto **1.4** richiedeva di trovare l’associazione (utente – numero di artisti ascoltati), per fare questo ci si è basati sul file “user\_artist.csv”, andando a contare il numero di volte che compariva un’associazione di un determinato utente con degli artisti. Una volta calcolato questo numero veniva inserita la coppia utente – cantanti ascoltati all’interno di una matrice.

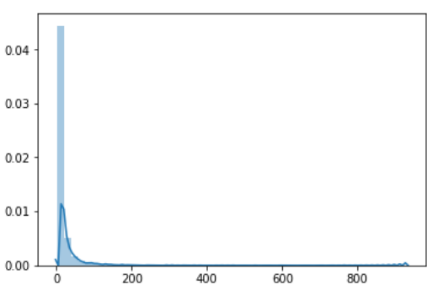
Il punto **1.5** richiedeva di andare a plottare un grafico contente la distruzione del numero di artisti ascoltati per ogni utente e di calcolare la media e la moda della distribuzione per fare questo si è utilizzato il vettore calcolato nel punto **1.4** e per calcolare la media e la moda si è eseguito il calcolo sullo stesso vettore.

Per plottare il grafico si è utilizzato seaborn attraverso la funzione displot.

La media della distribuzione è 44 artisti ascoltati per ogni utente e la moda è 45.

Fig 1: Distribuzione relativa al punto **1.5**

Si ottiene un grafico come in figura (Fig. 1).



Il punto **1.6** richiedeva di plottare la distribuzione di quanti tag ha ricevuto ogni artista e in seguito di calcolare la media e la moda della distribuzione. Per calcolare la distribuzione di questi valori è stato utilizzato il file “user\_taggedartists\_timestamps.csv” andando a contare per ogni artista qual è il numero di tag che gli sono stati assegnati. Dopo aver plottato la distribuzione anche in questo caso attraverso seaborn è stata calcolata la media e la moda della distribuzione.

Fig 2: Distribuzione relativa al punto **1.6**

La media della distribuzione è circa 15 e la moda è 1

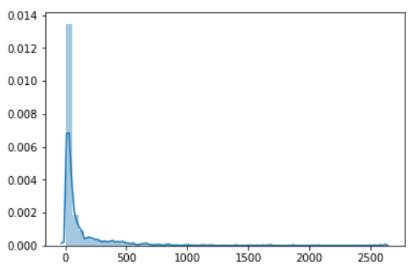
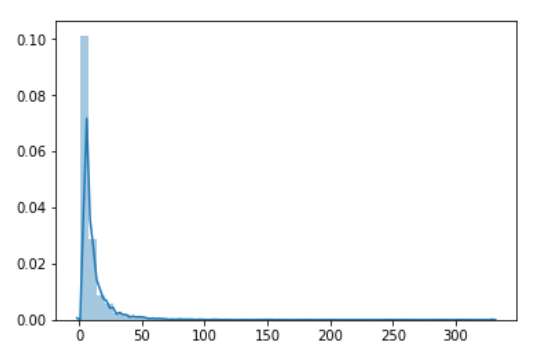
Il punto **1.7** richiedeva il calcolo e di plottare la distribuzione di quanti tag sono stati messi da ogni utente e di calcolare la moda e la media. Anche per questo punto si è utilizzato il file “user\_taggedartists\_timestamps.csv” ma andando a lavorare sui tag assegnati da ogni utente.

Fig 3: Distribuzione relativa al punto **1.7**

Si ottiene una media di circa 98 tag assegnati da ogni utente e una moda di 3 tag.

Il punto **1.8** richiedeva il calcolo e di plottare la distribuzione di quanti tag univoci sono stati assegnati a ogni artista e di calcolare la media e la moda della distribuzione anche in questo caso è stato utilizzato il file “user\_taggedartists\_timestamps.csv” andando a prendere tutti i tag assegnati ad ogni artista e in seguito rimuovere i duplicati presenti all’interno dell’array e contare quanti sono gli elementi univoci.

La media di tag univoci assegnati ad ogni artista è circa 9 e la moda è 1.

Fig 4: Distribuzione relativa al punto **1.8**

**2) Clustering tags**

Clusterizzare significa trovare gruppi di oggetti in modo tale che gli oggetti in un gruppo siano simili (o correlati) tra loro e diversi dagli (o non correlati) agli oggetti in altri gruppi. I diversi tipi di clustering sono: K-Means, Hierarchical e DBSCAN.

**K-Means:**

È un approccio di clustering partizionale che identifica il numero k di centroidi (in genere la media dei punti nel cluster), quindi assegna ogni punto al cluster più vicino.

I centroidi iniziali vengono scelti a caso.

La "vicinanza" può essere misurata in base alla distanza euclidea, alla somiglianza del coseno, alla correlazione o ad altre misure.

Per valutare le performance del K-Means si utilizza comunemente il valore (SSE), per ogni punto l’errore risulta essere la distanza dal centroide più vicino, per avere SSE va quadrata la somma di tutti gli errori per tutti i punti.

Un modo semplice per ridurre SSE risulta essere aumentare K, il numero di cluster.

**Bisecting K-Means:** variante di K-Means che può produrre un partizionale o

un cluster gerarchico e può risolvere il problema iniziale dei centroidi.

K-Means presenta le seguenti problematiche:

- Cluster di diverse dimensioni, densità o forme non globulari (soluzione: usa molti cluster. Trova gruppi di cluster e uniscili insieme)

- I dati contengono valori outlier

**DBSCAN**

DBSCAN è un algoritmo basato sulla densità, la densità è identificata dal numero di punti presenti all’interno di un raggio specificato (Epsilon). Si definisce dunque un punto come punto centrale (core point) nel caso in cui ha più di un numero specificato di punti (punti min.) all’interno di epsilon, questi sono i punti che risiedono all’interno di un cluster. Invece, un punto di confine (border point) è identificato come un punto che ha meno di punti min all’interno di epsilon ma si trova nelle vicinanze di un punto centrale. Infine, un punto di rumore (noise point) è qualsiasi punto che non è un punto centrale o di confine.

DBSCAN ha delle limitazioni quando si è in presenza di dati a densità variante e con un alta dimensionalità

**Hierarchical Clustering**

Produce una serie di cluster nidificati organizzati come un albero gerarchico (dendogramma).

Esistono due tipi principali di cluster gerarchici:

- Agglomerativo: inizia con i punti come singoli cluster e ad ogni passaggio, unisci la coppia di cluster più vicina fino a quando rimane solo un cluster

- Divisivo: inizia con un cluster all-inclusive e ad ogni passaggio, dividi un cluster fino a quando ogni cluster contiene un punto.

Gli algoritmi gerarchici tradizionali usano una matrice di somiglianza o distanza. I punti di forza del Clustering Gerarchico è che non è necessario assumere un numero particolare di cluster.

Limitazioni del Hierarchical Clustering:

- Una volta presa la decisione di combinare due cluster, non può essere annullata

- Diversi schemi hanno problemi con: sensibilità al rumore e valori anomali, difficoltà a gestire cluster di dimensioni diverse e forme convesse e rottura di cluster di grandi dimensioni

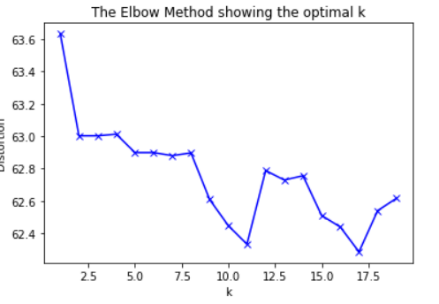
1. **Cluster tags according to the association to artists​. ​Justify your choices and comments on the results. Consider at least two different clustering algorithms based on similarity between tags, and compare the clustering.**

Per la soluzione di questo esercizio sono stati utilizzati come algoritmi di clustering K-Means e Hierarchical Clustering nella versione agglomerativa.

Prima di applicare gli algoritmi specificati è stato però necessario affrontare il problema degli outlier, per la rimozione di essi dal dataset si è deciso di rimuovere i tag che sono stati utilizzati meno di 5 volte in modo da avere un migliore risultato in fase di applicazione degli algoritmi di clustering. In seguito a questa procedura otteniamo 2074 categorie univoche di tag e 174486 tag.

Una volta rimossi quelli che noi consideriamo “outlier” è stato possibile costruire la matrice che ha sulle righe i tag e sulle colonne gli artisti e il valore rappresenta il numero di volte che un artista ha ricevuto quel tag, la matrice ottenuta viene normalizzata.

Andando a eseguire un prodotto elemento per elementi si può ottenere la matrice di similarità in modo da trovare la “similarità” tra i vari tag. La matrice di similarità viene normalizzata in modo da garantire dei migliori risultati in fase di clusterizzazione.

Per l’algoritmo del K-Means si utilizza il metodo nominato “Elbow” per trovare quale risulta essere il miglior valore di K da fornire all’algoritmo. Non risulta essere chiaro il grafico risultate da questa analisi ma si opta per un numero di cluster uguale a 11. Utilizzando questo valore abbiamo una buona distribuzione del numero di tag all’interno dei cluester.

**Fig 5:** Grafico generato dal “Elbow Method”

Al modello agglomerativo sono stati passati come numero di cluster 11 e tipo di affinità e quindi di similarità “cosine” i valori che questo parametro può assumere sono:

* *Euclidean Distance:* consiste nella distanza tra due punti. Richiede la standardizzazione se la scala dei  
  dati `e differente.
* *Manhattan Distance:* `e la distanza in linea d’aria
* *Cosine Similarity:* misura la somiglianza di due vettori proiettati in uno spazio multidimensionale, in base all’angolo tra loro [0, pi]. La cosine similarity assume un valore compreso nell’intervallo [*-*1*;* 1]

Anche in questo caso otteniamo una distribuzione dei tag all’interno dei cluster abbastanza buona.

1. **Associate to each cluster a meta-tag, and consider using it as tag for the music social network.**

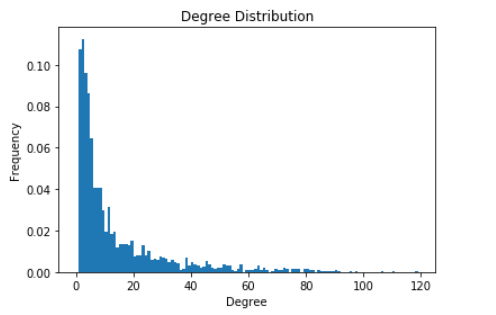
Per lo sviluppo di questo punto si è partiti dal dataset contente gli artisti taggati e si è associato il numero identificativo del cluster di appartenenza, in una nuova colonna chiamata MetaTag.

Andando a prendere l’artista più ascoltato da un’utente e metatag associato all’artista è stato possibile creare una matrice che ci l’associazione tra l’utente e il metatag alla quale appartiene.

1. **Group music artists based on the meta-tag and analyse the distribution of the artists with respect to the related meta-tags.**

**Task3: Social network analysis:**

1. **Compute the network properties: degree distribution, density, diameter.**



**Fig. 6**: Degree Distribution

Il set di dati utilizzato per calcolare la rete è costituito da relazioni di amicizia tra utenti facenti parte della stessa rete. Quindi i nodi rappresentano gli utenti nel set di dati e gli archi rappresentano le relazioni tra gli utenti.

- Nodi: 1892

- Bordi: 12717

- Densità: 0,0071

- Diametro: 9

- Degree Distribution : vedi Fig. 6

Il diametro(diameter) di un grafico è la lunghezza della SP più lunga tra qualsiasi coppia di nodi nel grafico. Mentre la densità sarebbe definita come il rapporto tra il numero di spigoli e il numero di spigoli possibili. Il valore massimo è 1 dove il grafico è completo.

1. **Compute the centrality measures, select the one that you consider more representative and justify your choice.**

**Centrality**

Le misure di centralità vengono utilizzate per classificare i nodi delle reti in base alla loro importanza. I diversi tipi sono:

- **Degree centrality:** classifica i nodi con connessioni più / migliori.

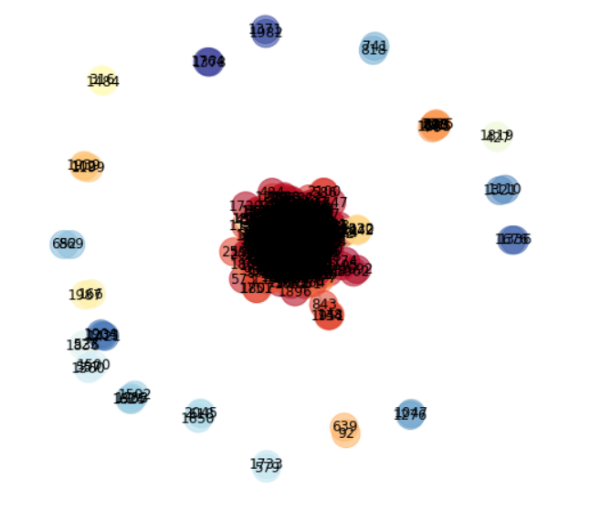
- **Closeness centrality** : è il reciproco della lunghezza dei loro percorsi più brevi.

- **Betweenness centrality** : quantifica il numero di volte in cui un nodo funge da ponte lungo il percorso più breve tra altri due nodi.

- **Group centrality** : misura l'importanza di un gruppo all'interno della rete

In questa rete la centralità che si ritiene essere più rappresentativa risulta essere la degree centrality perché ci dà l’idea di chi è l’utente della rete con più amici e quindi l’utente più “influente” della stessa. In questo caso il nodo con degree centrality maggiore è il nodo 1543.

Inoltre, la betweenness centrality può essere un indicatore di quelle persone che fanno da ponte tra diversi generi musicali.

1. **Extract:  
   ○ The communities based on social network**

**Fig.7:** Communities presenti all'interno del social network

Le comunità, o cluster, sono gruppi di nodi organizzati insieme all'interno di una rete o di un grafico. In questa rete le comunità identificate sono 32(vedi Fig. 7).

**○ The communities based on similarity (e.g., preferred artist(s), music type)**

1. **Compute the global clustering coefficient of:**

**○ the global network  
○ the communities  
○ (OPTIONAL) compare them and discuss the result**

Global clustering coefficient è basato su triple di nodi. Una tripla consiste in tre nodi connessi da due o tre collegamenti. Ogni tripla è incentrata su un nodo. Un triangolo consiste in tre triple chiuse incentrate sui tre stessi nodi che le compongono. Il coefficiente di clustering globale è, dunque, il numero di triple chiuse (o 3 volte il numero di triangoli) fratto il numero totale di triple (somma di quelle aperte e chiuse). Il global clustering coefficient rappresenta il grado in cui i nodi di un grafo tendono ad essere connessi tra loro.

Il valore di global clustering è 0.13 nella rete di amici.

All’interno delle communities il valore di global clustering coefficient risulta essere un valore più alto e questo è dovuto ad una maggiore interconnessione tra i nodi all’interno di una community.

1. **Compute the local clustering coefficients, and extract the average, within:  
   ○ the global network  
   ○ the communities  
   ○ (OPTIONAL) compare them and discuss the result**

Il coefficiente di clustering locale di un [nodo](https://it.wikipedia.org/wiki/Vertice_(teoria_dei_grafi)) in un grafo è una misura di quanto i suoi vicini tendano a formare un grafo completo.

Nella rete il valore di clustering locale(local clusering) per la rete degli amici è circa 0.18.

All’interno delle communities il valore di local clusering risulta essere maggiore perché si ha una maggiore interconnessione tra i nodi

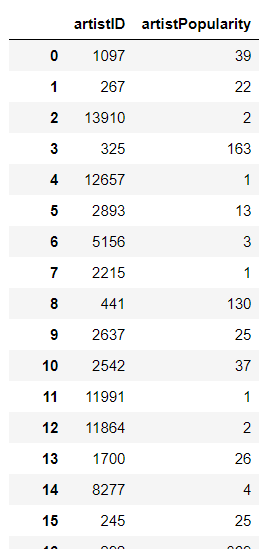
1. **Compute the similarity between users according to:  
   ○ Their common friends  
   ○ The artist(s) listened  
   ○ Music type  
   ○ (OPTIONAL) The artist(s) preferred by their friends  
   ○ (OPTIONAL) The type of music listened by their friends**

Il coefficiente di somiglianza Jaccard confronta i membri per due serie per vedere quali membri sono condivisi e quali sono distinti. È una misura di somiglianza per le due serie di dati, con un intervallo dallo 0% al 100%. Maggiore è la percentuale, più simili sono le due popolazioni. Sebbene sia facile da interpretare, è estremamente sensibile alle dimensioni di campioni piccoli e può dare risultati errati, specialmente con campioni o set di dati molto piccoli con osservazioni mancanti.

Questo indicatore è stato utilizzato nel calcolo della somiglianza tra utenti facendo riferimento ai loro amici in comune, i loro artisti ascoltati e il tipo di musica ascoltato.

**Task4: Binary classifier for inferring users's artists interest.  
Build a binary classifier which is able to infer, given a tuple <user-artist>, if the user would be interested in the artist or not. In order to build the classifier consider a set of features (at least 10 features are expected) which characterize a tuple user-artist. Some examples of features are listed below, given a tuple of user U and artist A.  
- Feature 1: popularity of artist A.**

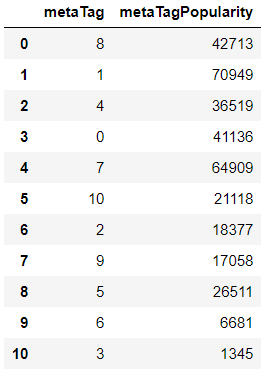
Per il calcolo della popolarità dell’artista A è stato calcolato il numero di persone che seguono quel determinato artista. Per fare ciò è stato utilizzato un doppio ciclo *for* nel quale per ogni artista si conta quanti sono i suoi “follower”.



**Fig. 8:** Tabella contente il numero di "follower" per ogni artista

**- Feature 2: popularity of the N clusters C(i), which artist A belongs to.**

Per questo punto, dato il metaTag, si sommano tutte le popolarità di tutti gli artisti appratenti a quel metaTag e si ottiene una tabella come la seguente(Fig.9).

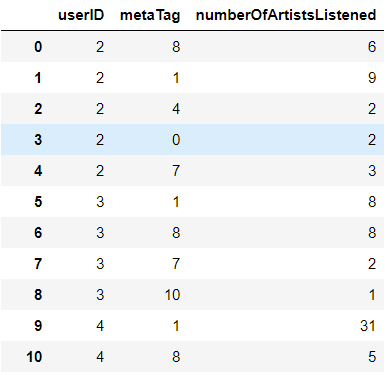


**Fig. 9:** Tabella con riferimento al metatag e alla popolarità dello stesso

Per ottenere questa associazione tra metaTag e la popolarità del metaTag si selezionano gli artisti del metaTag corrente. In seguito, si calcolano le popolarità degli artisti selezionati e le si sommano per trovare la popolarità del metaTag corrente.

**- Feature 3: number of artists listened to by user U, belonging to the same clusters C(i) of artist A.**

Per lo svolgimento di questo punto si sono estratti tutti gli artisti che ascolta l’utente corrente. In seguito, per ogni metaTag si contano quanti artisti contiene e si aggiunge una riga nella tabella rappresentata in Fig. 10.



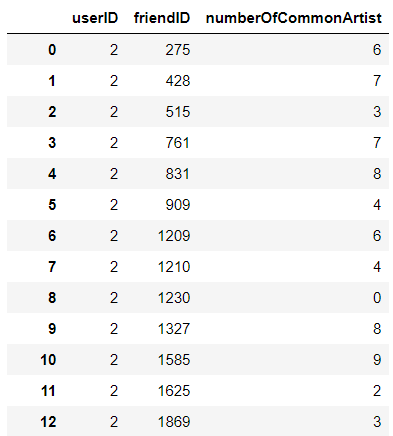
**Fig. 10:** Tabella con riferimento all’utente al metaTag e al numero di artisti ascoltati di quel metaTag

**- Feature 4: number of friends of user U, which listened to artist A.**

Per lo svolgimento di questo punto si estraggono gli amici dell’utente corrente. In seguito, viene estratta la lista degli artisti che ascolta l’utente. Questo procedimento viene svolto anche per ogni amico e quindi avrò una lista di artisti ascoltati per ogni amico.

Infine, si contano quanti amici sono in comune tra l’utente corrente e tutti i suoi amici.

Il risultato di questo procedimento sarà una tabella che avrà la forma seguente (Fig. 11).



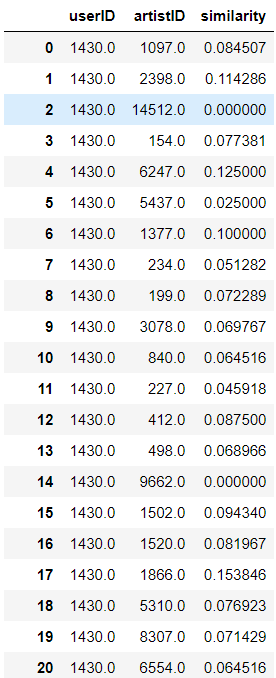
**Fig. 11:** Numero di artisti in comune tra due amici

**- Feature 5: boolean representing if artist A similarity to the user's most listened artists is  
lower than a certain distance threshold ​*d*​.**

Per lo svolgimento di questo punto sono stati ricavati gli artisti dell’utente corrente (ciclando su tutti gli utenti). Viene ricavato l’artista preferito dell’utente corrente ordinando gli artisti ascoltati dall’utente per peso e prendendo in seguito il primo nella lista.

Vengono poi presi i tag dell’artista preferito e per ogni artista dell’utente corrente calcolo la similarità degli artisti con l’artista preferito, prendendo i tag in comune e dividendoli con i tag totali. Così facendo si ha un ordine di similarità tra gli artisti.

Alla fine di questo processo avremo una tabella come quella presentata nella figura Fig. 12.

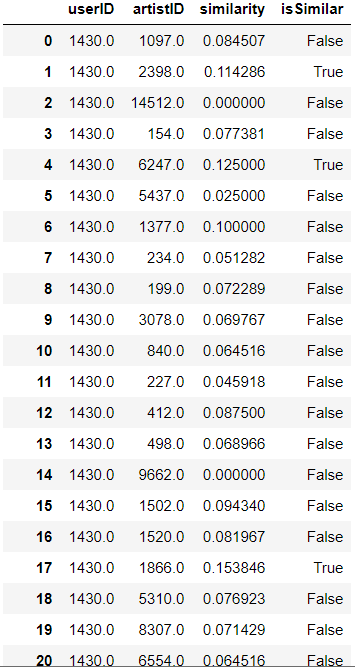


**Fig. 12:** Tabella contente la similarità degli artisti all'artista preferito dell’utente

Per ottenere un valore booleano come valore di similarità si è deciso di prendere come valore di soglia il valore medio. Se un valore di similarità sorpassa la media, l’artista viene identificato come simile. Invece, se l’artista ha un valore di similarità al di sotto della media, viene categorizzato come non simile all’artista preferito.

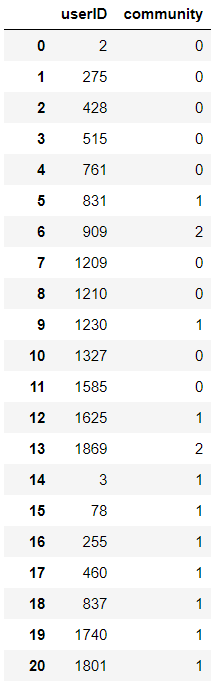
Quindi, avrò un dataframe contenente userID, ArtistID, similarity e isSimilar. La colonna isSimilar sarà la nostra feature da tenere in considerazione durante il train del nostro modello.

Il dataFrame risultante è il seguente. (Fig. 13)



**Fig. 13:** Tabella contente la similarità degli artisti all'artista preferito dell'utente(booleana)

**- Feature 6: the communities computed in Task 3.**



**Fig. 14:** Tabella contente le communities calcolate nel task 3

Nella tabella rappresentata in Fig. 14 abbiamo l’associazione tra ogni utente e la community a cui appartiene. Anche questa informazione verrà passata al nostro classificatore per avere una migliore classificazione.

Inoltre, è stata inserita come ultime feature userLikeArtist. Essa contiene informazioni che determinano se all’utente piace quell’artista. Per calcolare queste informazioni è stata calcolata la media dei pesi degli artisti. Se un utente ha un interesse sopra la media per un determinato artista viene settata a true.

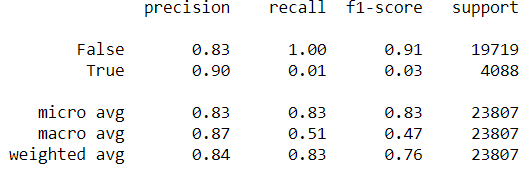
Una volta finita la parte di generazione delle varie features è stato necessario creare un dataframe contente tutte le varie features calcolate, in modo da passarle al classificatore.

**Classificatore SVC**

Support Vector Classifier (SVC) trova un iperpiano che massimizza il margine (bassa varianza) e minimizza la classificazione errata (bias elevato). SVC utilizza cross validation per trovare il valore migliore per il margine morbido.

Per la classificazione, in questo punto, è stato utilizzato il classificatore SVC in modo da avere una classificazione binaria delle istanze <user-artist>.

Per la divisione dei dati in input tra dati di train e dati di test è stato deciso di utilizzare la tecnica del Holdout e quindi di avere 1/3 delle istanze per il test e 2/3 delle istanze per il train.



**Fig. 15:** classificatore SVC e relative statistiche

Si ottiene un’accuratezza in fase di classificazione pari a circa il 67% delle istanze di test.

**Task5: (OPTIONAL) Scalable classifier**

**Similarly to Task 4, build a classifier which is able to infer, given a tuple ​*<user-artist>*​, if the user would be interested in the artist or not, but also consider that new users can enter the network.  
In this sense, the proposed solution should be able to minimize the training phase, that is: it  
should be avoided to have to restart the training phase from scratch.  
In order to build such a dynamic and scalable classifier consider only part of the original network (e.g ¾ of the original network) and then simulate to add the remaining users.  
Show the performances of the proposed classifier and compare it with the performances of the one produced in Task4 in the case of ⅚ of the original network in the beginning and the  
remaining ⅙ entering the network later.**

**Task6: Inferring users interest in artist.**

**Build a system which is able to recommend the weight of a couple ​*<user-artist>*​, which is the number of times a user is going to listen to this artist. The recommendation should be  
categorical, and the possible values are the following.  
● HIGH: weight greater than 1000 (high interest in the artist).  
● MEDIUM: weight between 1 and 1000 (medium interest in the artist).  
● LOW: weight equal to zero (no interest in the artist).  
In order to build the classification model, consider a set of features (at least 10 features are expected) which characterize a tuple user-artist. Some examples of features are listed in task 4.**

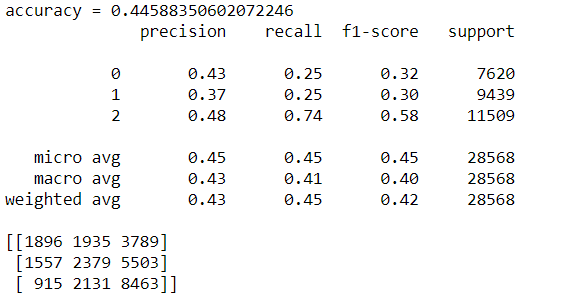
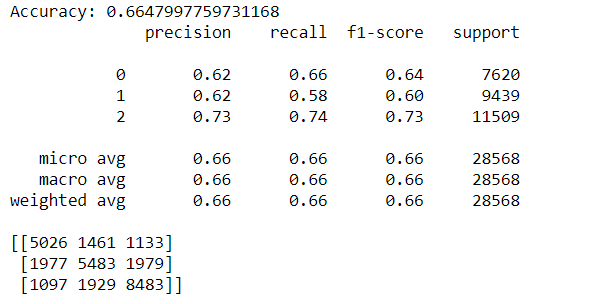
In questo compito, per la costruzione della variabile target, è stata categorizzata la colonna “weight” con i valori 0 (che indica il valore “LOW”, quindi un peso minore di 300), il valore 1 (che indica il valore “MEDIUM”, quindi un peso maggiore di 300 e minore di 1000) e il valore 2 (che indica il valore “HIGH”, quindi maggiore di 1000).

Lo scopo del nostro classificatore è quello di prevedere quale sarà l’interessamento di un utente verso un determinato artista.

**Logistic Regression:** rappresenta un modello di regressione nonlineare utilizzato quando la variabile dipendente è di tipo dicotomico. L'obiettivo del modello è di stabilire la probabilità con cui un'osservazione può generare uno o l'altro valore della variabile dipendente; può inoltre essere utilizzato per classificare le osservazioni, in base alla caratteristiche di queste, in due categorie.

**KNN:** è un algoritmo utilizzato nel riconoscimento di pattern per la classificazione di oggetti basandosi sulle caratteristiche degli oggetti vicini a quello considerato.

Per questo punto sono stati sviluppati due modelli di classificazione: un modello utilizzando la logistic regression mentre l’altro modello utilizzando il KNN.



**Fig. 17:** classificatore KNN e relative statistiche

**Fig. 16:** classificatore Logistic Regression e relative statistiche

Come si può notare utilizzando la Logistic Regression otteniamo un valore di accuratezza pari a circa il 45%. Mentre con KNN, con un numero di vicini uguale a 10, otteniamo un valore di accuratezza pari al 65%.

**File result.csv:**

Ho provato in diversi modi a generare il file result.csv ma non ho trovato una soluzione ottimale in quanto il mio modello prende in input delle features che non possono essere ricostruite dalla coppia user artist.