MUSIC SIMILARITY

FINAL PROJECT - ΜΟΥΣΙΚΗ ΠΛΗΡΟΦΟΡΙΚΗ

ΠΡΙΜΕΝΤΑ ΑΙΚΑΤΕΡΙΝΗ-ΜΑΡΙΑ
ΑΜ: 1115201900160

ΕΙΣΑΓΩΓΗ

Το παρόν project αφορά στην ενασχόληση με το αντικείμενο του music similarity (μουσική ομοιότητα). Το συγκεκριμένο πρόβλημα (task) ανήκει στο πεδίο του MIR (Music Information Retrieval), το οποίο ασχολείται γενικώς με την ανάλυση και την οργάνωση μουσικής-ηχητικής πληροφορίας. Αυτό γίνεται καταρχήν με τεχνικές επεξεργασίας σήματος (signal processing) ώστε να ληφθεί η κατάλληλη πληροφορία και στη συνέχεια χρησιμοποιούνται διάφορες τεχνικές και αλγόριθμοι για την οργάνωση, όπως για παράδειγμα αλγόριθμοι συσταδοποίησης και κατηγοριοποίησης (data mining), είτε άλλοι μέθοδοι μηχανικής μάθησης. Το music similarity έχει σκοπό να ερευνήσει τον βαθμό ομοιότητας μεταξύ 2 ή περισσότερων μουσικών κομματιών, του ίδιου ή διαφορετικού είδους, χρησιμοποιώντας διάφορα χαρακτηριστικά των κομματιών. Άλλα παρόμοια παραδείγματα που ανήκουν σε αυτό το πεδίο είναι η κατηγοριοποίηση με βάση το είδος μουσικής (genre classification) και σύστημα προτεινόμενων κομματιών (music recommendation).

ΕΦΑΡΜΟΓΗ

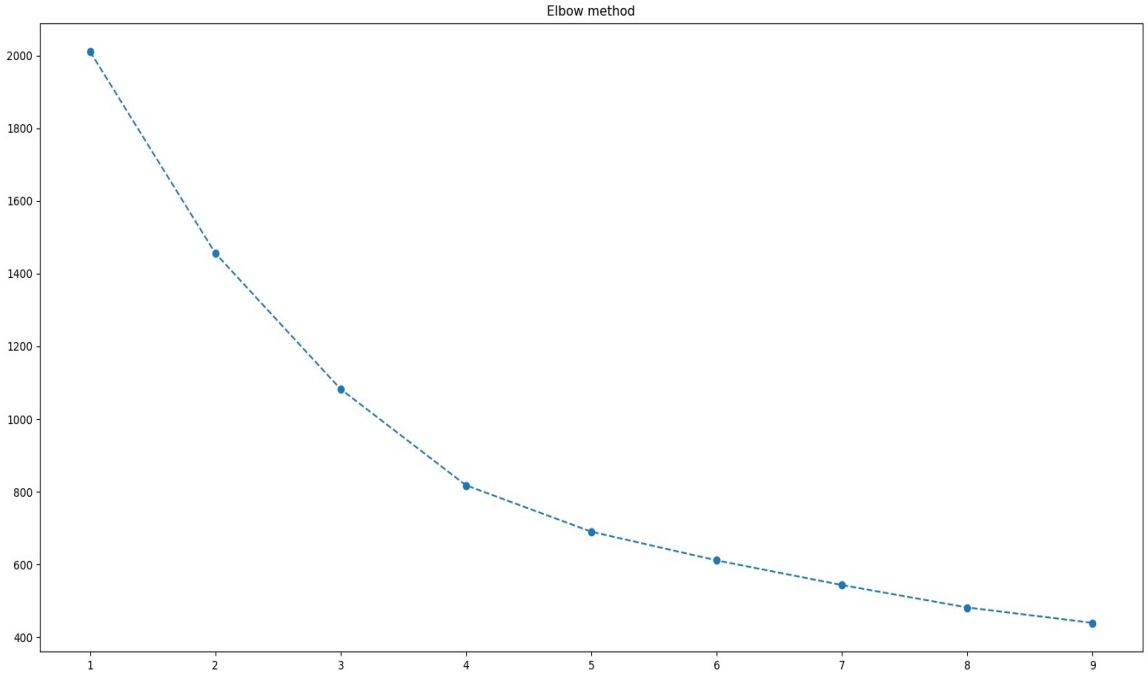
Το πρόβλημα music similarity που υλοποιείται εδώ είναι η ομοιότητα (ή ανομοιότητα) μεταξύ κομματιών του ίδιου είδους μουσικής. Στόχος της υλοποίησης είναι κυρίως η παρατήρηση αποτελεσμάτων και η σύγκρισή τους με την αντίληψη του ακροατή.

Συγκεκριμένα, επιλέχθηκε ένα μεγάλο dataset με περίπου 600 audio files της μορφής .flac τα οποία γενικώς ανήκουν στο είδος της jazz. Θεωρώντας ότι αυτό το είδος είναι αρκετά γενικό και αποτελείται από πολλά υποείδη, τα δεδομένα μας αποτελούν κομμάτια διαφορετικών υποειδών. Για παράδειγμα, υπάρχουν αρχεία του είδους ragtime, boogie-woogie, rockabilly, blues, pop (με jazz αρμονική βάση), κλασσική jazz και πιο ελεύθερη jazz. Από αυτά φυσικά υπάρχουν πολλά κομμάτια που ανήκουν στον ίδιο καλλιτέχνη, οπότε ένα αναμενόμενο αποτέλεσμα είναι να ταιριάζουν κομμάτια του ίδιου καλλιτέχνη και υποείδους.

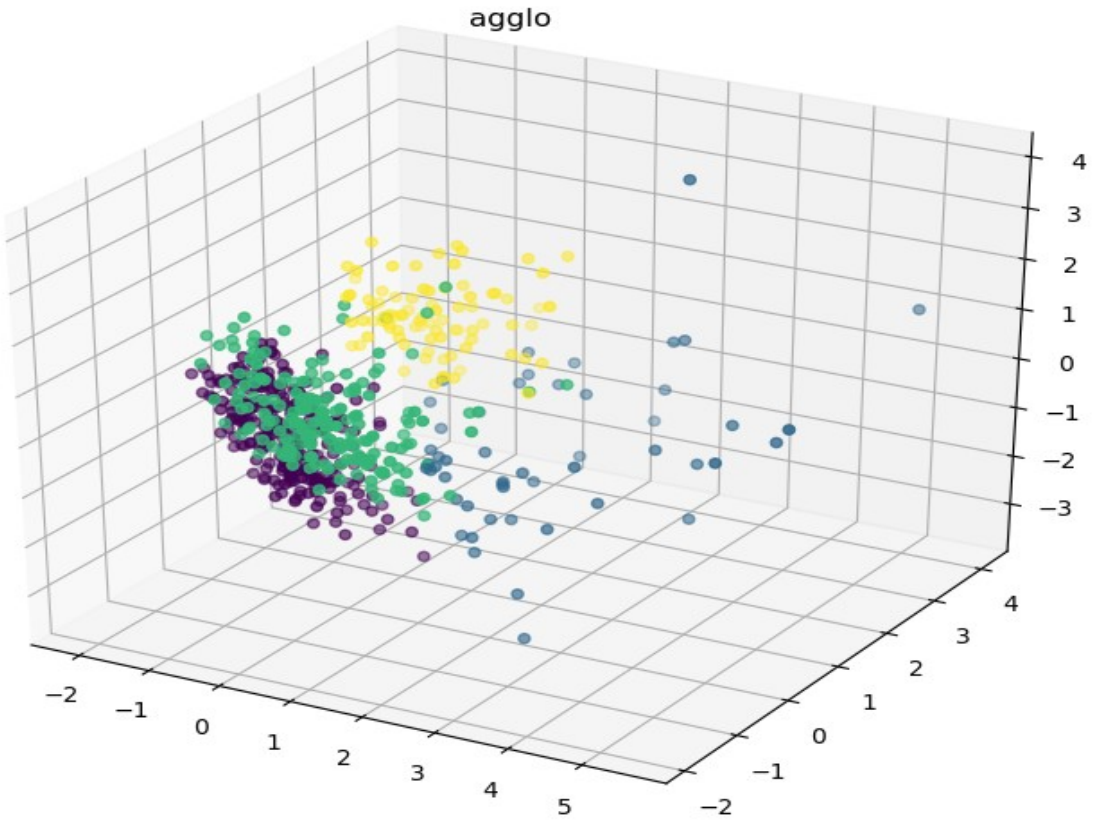
Στο αρχείο createDataframe.py βρίσκεται η υλοποίηση για την κατασκευή ενός .csv file ώστε να συγκροτηθούν όσο το δυνατόν περισσότερες πληροφορίες από κάθε κομμάτι. Το .csv που προκύπτει είναι το music_dataset.csv, μετά από αρκετή ώρα εκτέλεσης του script. Λόγω της μορφής των αρχείων (flac), είχαμε πολύ εύκολη πρόσβαση σε στοιχεία του κάθε κομματιού, όπως ο τίτλος, ο καλλιτέχνης, το άλμπουμ, η ημερομηνία κυκλοφορίας (η οποία αποδείχτηκε λαθος οπότε και αγνοήθηκε στη συνέχεια) και η διάρκεια. Σε πρώτη φάση λοιπόν, επλέχθηκε να προστεθούν 3

ακόμα χαρακτήριστικά, τα οποία έγιναν extract με τη χρήση έτοιμων βιβλιοθηκών της python. Αυτά είναι το tempo, η σκάλα (μείζονα/ελάσσονα) και το loudness. Δεν χρησιμοποιήθηκαν παραπάνω χαρακτηριστικά, καθώς προέκυψαν προβλήματα κατά την εκτέλεση τα οποία ήταν πολύ δύσκολο να εντοπιστούν δεδομένου του όγκου του dataset. Για αυτό τον λόγο μάλιστα χρησιμοποιήθηκε μία διαφορετικη προσέγγιση που θα αναλυθεί στη συνέχεια.

Το αρχείο clustering.py περιέχει όλη την υλοποίηση για την συσταδοποίηση των δεδομένων. Επομένως, πρώτη φάση της υλοποίησής μας είναι να χωρίσουμε τα δεδομένα σε ομάδες (clusters), ανάλογα με το πόσο ταιριάζουν σε 4 χαρακτηριστικά: tempo, loudness, duration, scale. Χρησιμοποιείται pca για την μείωση των διαστάσεων των δεδομένων και στη συνέχεια εφαρμόζεται το elbow method με χρήση του k-means για να βρούμε τον κατάλληλο αριθμό clusters. Από την εκτέλεση, παρατηρούμε ότι οι συστάδες θα είναι 4.

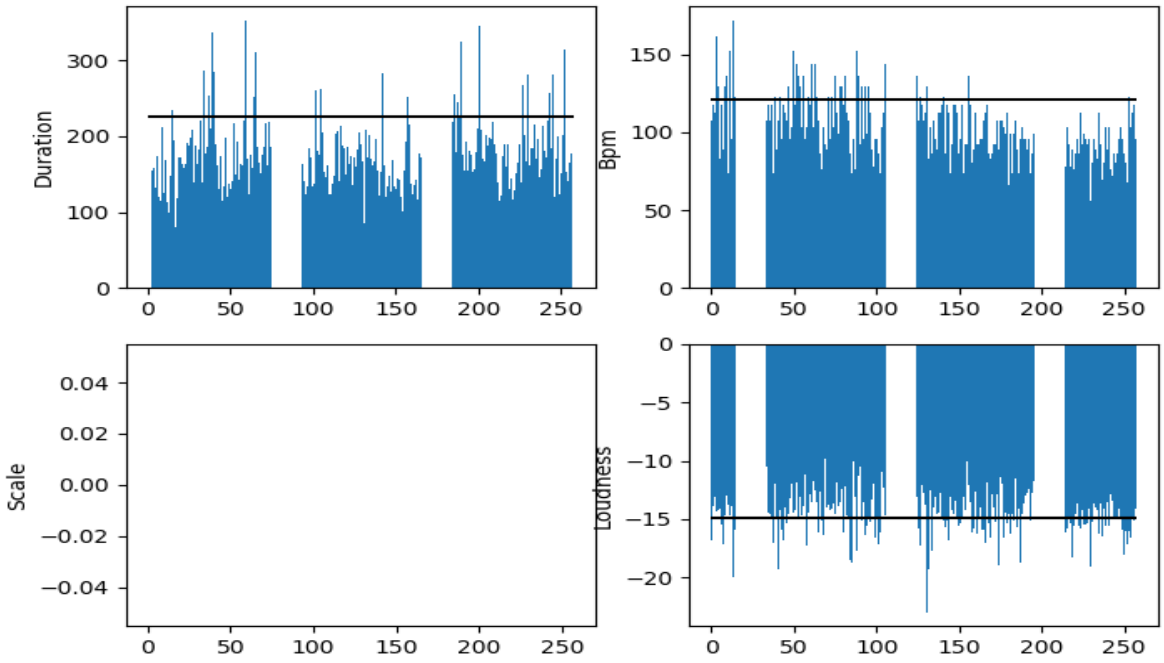


Για τη συσταδοποίηση, αποφασίστηκε η χρήση ιεραρχικού αλγορίθμου, καθώς είναι γενικά πιο αποδοτικός από τον k-means. Τα αποτελέσματα του agglomerative φαίνονται στο παρακάτω projection.

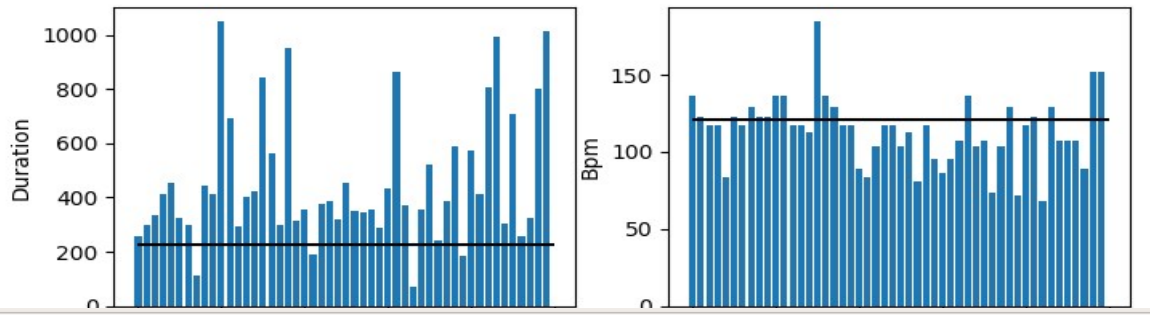


Για καθένα από τα 4 clusters, δημιουργούμε διαγράμματα για καθένα από τα 4 χαρακτηριστικά που λήφθηκαν υπ’όψιν. Τα clusters αποθηκεύονται σε .csv αρχεία για μετέπειτα χρήση τους.

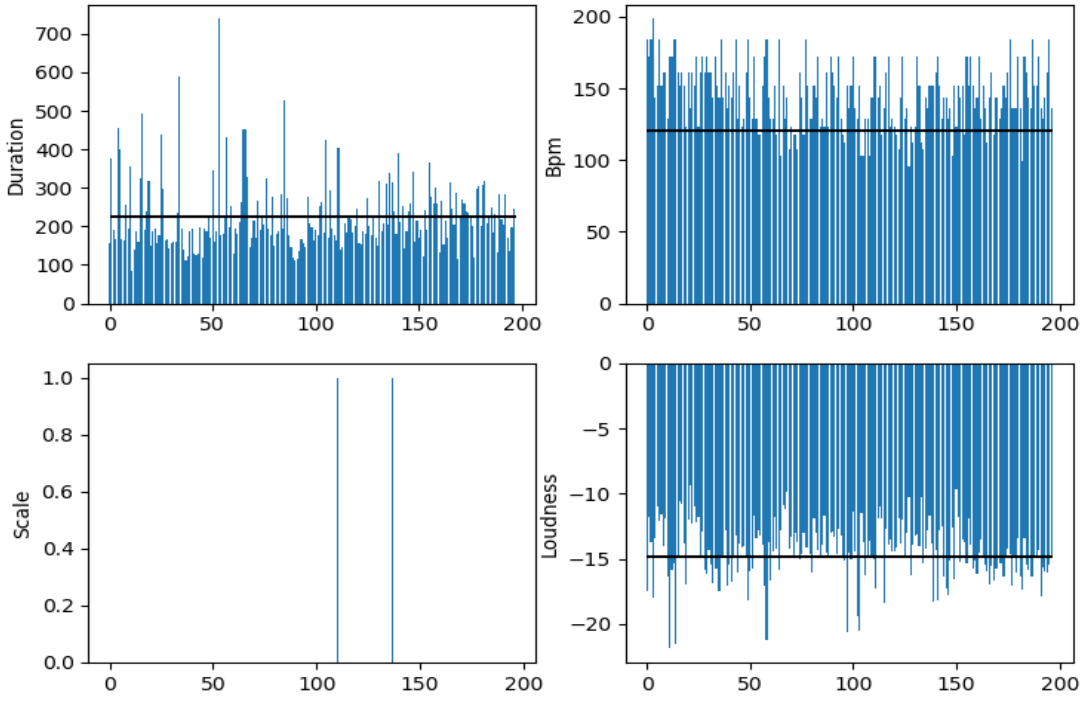
Cluster 0

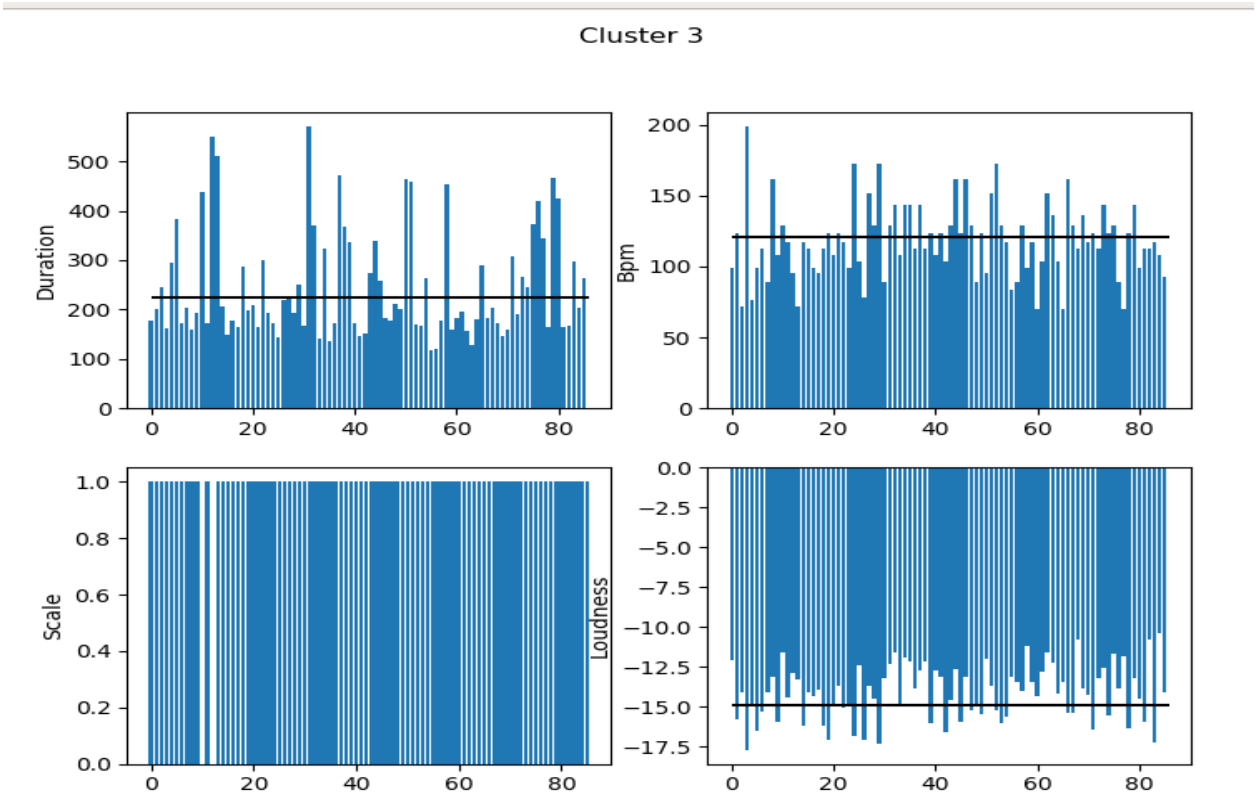


Cluster 1



Cluster 2

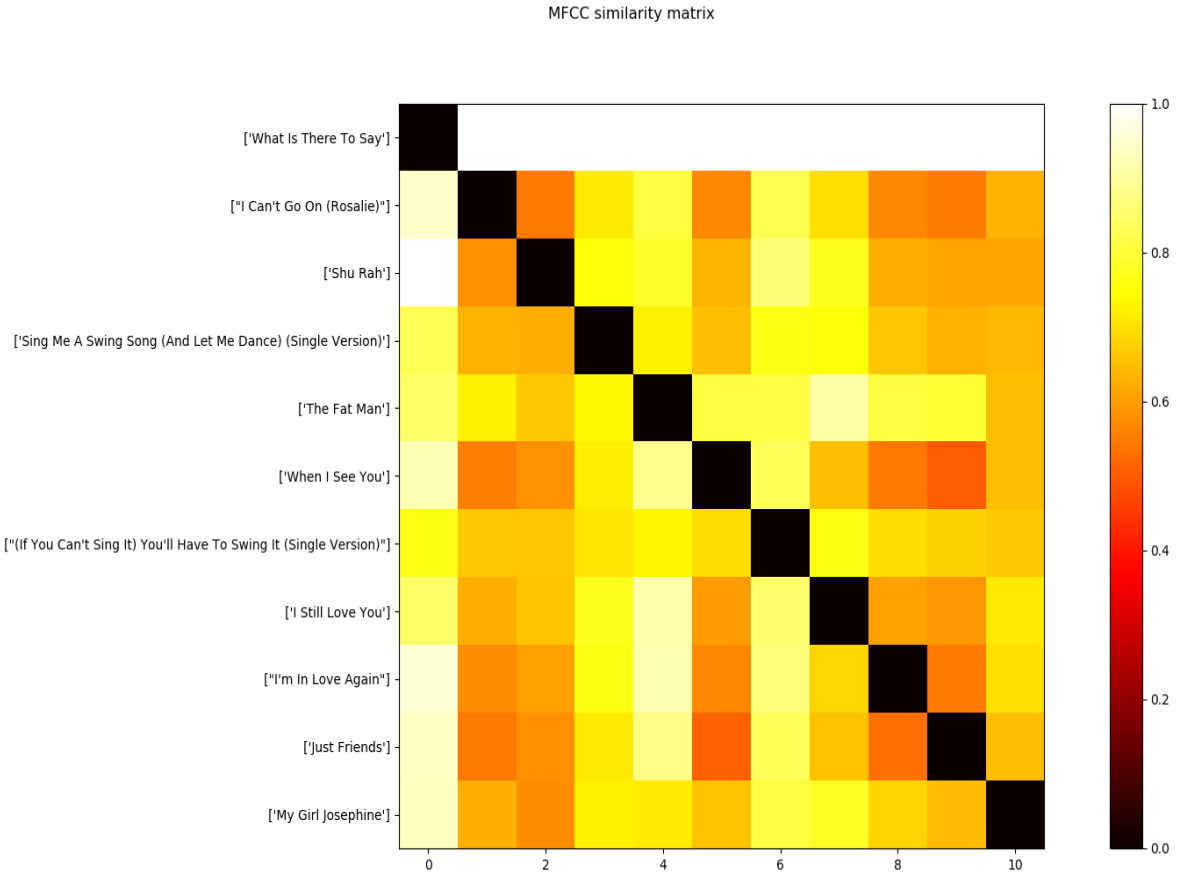


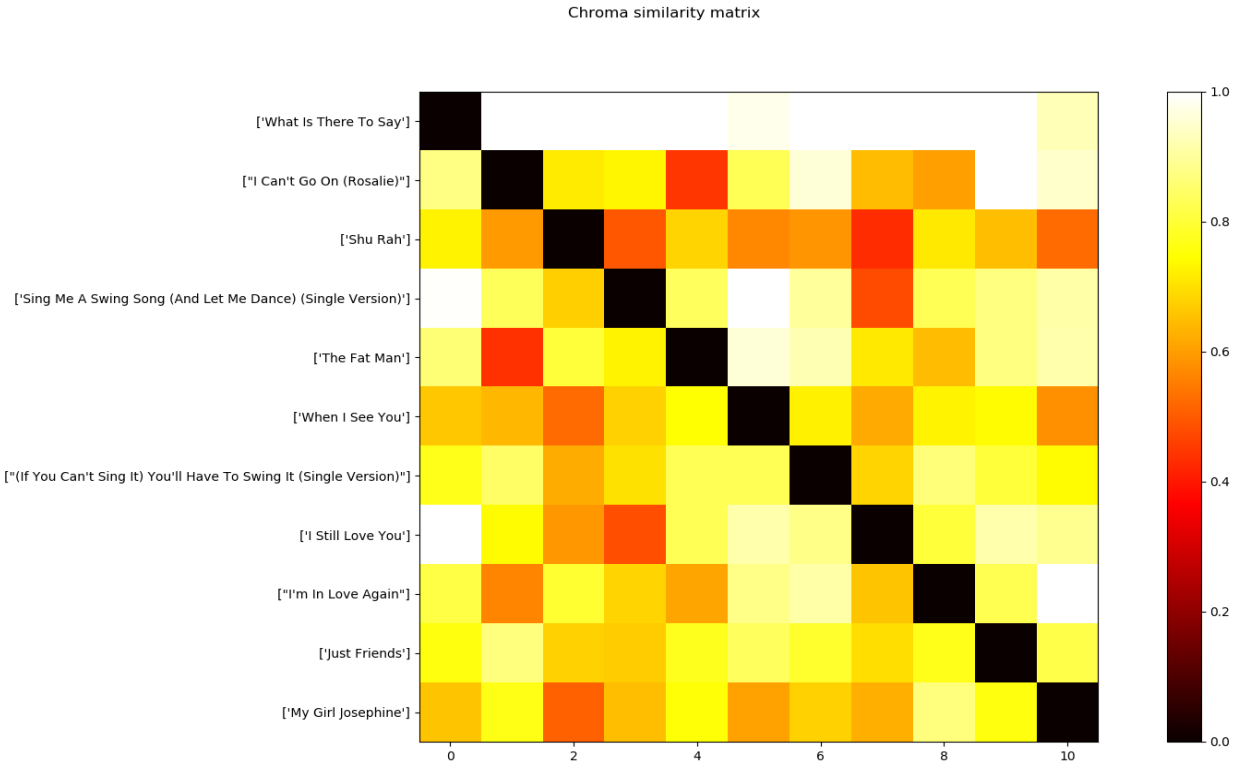


Παρατηρούμε γενικά ότι το cluster 0 έχει κομμάτια σε μείζονα κλίμακα (τιμή 0) αντίθετα με το 3 που έχει κυρίως κομμάτια σε ελάσσονα κλίμακα (τιμή1). Επιπλέον, το cluster 1 έχει πολύ λιγότερο πλήθος κομματιών με τα περισσότερα να είναι μεγάλης διάρκειας. Το cluster 2 έχει τα πιο γρήγορα σε ρυθμό κομμάτια, κρίνοντας από το bpm διάγραμμα.

Τέλος, το script μας παραθέτει για κάθε συστάδα τους πιο δημοφιλής καλλιτέχνες με βάση το πλήθος των κομματιών που περιέχονται σε αυτό. Το dataset βέβαια περιέχει καλλιτέχνες που έχουν πολλά περισσότερα κομμάτια από κάποιους άλλους. Επομένως, έχει περισσότερη σημασία να εντοπίσουμε το πλήθος των κομματιών κάθε καλλιτέχνη σε ένα cluster και πόσες φορές εμφανίζεται σε άλλα clusters.

Στο αρχείο similarity.py, περιέχεται μία μοναδική συνάρτηση, η οποία παίρνει για όρισμα τον αριθμό της συστάδας που επιθυμούμε να διερευνήσουμε περαιτέρω. Για λόγους απλότητας και ευκολίας, κρίθηκε κατάλληλο να χωρίσουμε πρώτα το dataset σε συστάδες και μετά να δούμε την ομοιότητα μεταξύ κομματιών του ίδιου cluster. Έτσι, για την ομοιόμορφη απεικόνιση των similarities, παίρνουμε 10 από τα κομμάτια ενός cluster (έστω του cluster 0) και κάνουμε extract το mfcc και το chromagram τους. Για καθε δυάδα από αυτά τα 10 κομμάτια, βρίσκουμε τη μέση ευκλίδεια απόσταση χωριστά για το mfcc και το chroma και την αποθηκεύουμε σε πίνακα. Παρακάτω βλέπουμε τον similarity matrix του mfcc και του chroma για 10 κομμάτια από το cluster 0.





Όσο πιο σκούρο το χρώμα, τόσο μικρότερη η απόσταση μεταξύ των χαρακτηριστικών διαφορετικών κομματιών. Η κύρια διαγώνιος είναι η πιο σκούρα, καθώς αντιπροσωπεύει την

ομοιότητα μεταξύ των ίδιων κομματιών (default distance = 1). Σε περαιτέρω παρατήρηση και ανάλυση των διαγραμμάτων, διαπιστώνουμε ότι το πρώτο κομμάτι διαφέρει αρκετά από όλα τα υπόλοιπα. Αυτό είναι λογικό καθώς είναι διαφορετικού καλλιτέχνη από τα υπόλοιπα κομάτια. Παρόλα αυτά, δεν μπορούμε να αγνοήσουμε ότι το διάγραμμα φέρει έντονα χρώματα, άρα και αρκετές ομοιότητες μεταξύ διαφορετικών κομματιών.

Προβήματα που εμφανίστηκαν κατά την εκτέλεση αυτού του script είναι το γεγονός ότι χρειάζεται πιθανώς καλή RAM στον υπολογιστή για να εκτελεστεί το προγραμμα και για τα άλλα clusters (για αυτό και δεν παρουσιάζονται άλλα διαγράμματα άλλων clusters). Παρά τις προσπάθειες για χρήση pca στο similarity και batches κατά τον υπολογισμό των αποστάσεων, δεν κατέστει εφικτή κάποια λύση.

ΑΝΑΦΟΡΕΣ

For clustering:

https://towardsdatascience.com/k-means-clustering-and-pca-to-categorize-music-by-similar-audio-features-df09c93e8b64

Python libraries:

https://librosa.org/doc/latest/index.html

https://pypi.org/project/audio-metadata/

https://essentia.upf.edu/reference/std_MusicExtractor.html

https://pypi.org/project/pyloudnorm/

https://scikit-learn.org/stable/modules/classes.html#

https://docs.scipy.org/doc/scipy/reference/spatial.html