Predicting Musical Genres Using Machine Learning: An Exploratory Study of the k-Nearest Neighbors Algorithm

Daniel Lopez Villa
University of California, San Diego
Email: dlopezvi@ucsd.edu

Jimmy Do
University of California, San Diego
Email: jdd010@ucsd.edu

Abstract
This report will explore in depth the implementation of the k-Nearest Neighbors in predicting musical genres. Using Spotify's Web API, datasets containing several musical attributes were obtained. Using multiple attributes such as danceability and valence, we were able to train our model using kNN and successfully predict the genre of a given set of songs with about 90% test accuracy. Although initial prediction results were very promising, we attempted to further refine our model by tweaking KNN parameters. This resulted in very marginal changes to overall accuracy of our model.

Introduction/Motivation
In the world of music, genre classification is very subjective and often counterproductive when describing the essence of a piece of art(music). Nonetheless the goal of this project is to categorize an arbitrary list of songs by genre to gain a greater appreciation and understanding for the distinction between artistic interpretation and analytical data. There are many ways to approach the problem of musical analysis, and because of the project's complexity and the subjective nature of music, great care must be taken in selecting parameters which contribute to a unique genre. The debate between analytical and artistic interpretation of a song is one which will continue for years to come, but it is our intention to investigate and simplify this disparity. Although our project scope is narrow, the results from this investigation have implications across many breadths especially in the music streaming industry where predictive algorithms aid in retaining listeners.

Problem Solving Approach
The first problem unanimous with any machine learning project is deciding what type of data set is appropriate in the context of our overarching goal. In the case of musical genre classification, the data set must meet the following criteria: the data set must be sufficiently large, the data set must be sufficiently diverse, and the data set must be sufficiently accurate. In addition to these three requirements, perhaps the most important decision influencing the choice of data is the method of analysis, or in other words how we would interpret a list of songs to provide an appropriate genre prediction. Initially we considered a strict audio analysis as a form of data extraction, where we would perform a Fourier Transform on the raw audio file to analyze the frequency spectrum of a song, but ultimately we decided against this approach due to complexity. Because we are still relatively new to the world of machine learning and Python programming in general, we thought it

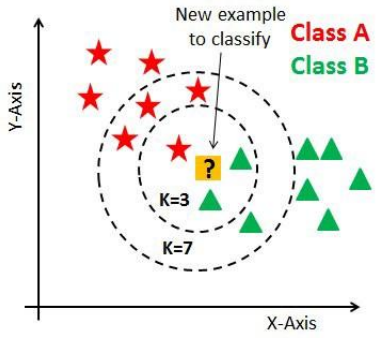
best to not over complicate our analysis and instead choose a data set which offered an intuitive library of musical attributes. As a more appealing alternative, we turned our eyes to the Spotify Web API which offered a library of characteristics such as Dancibility, Energy, Instrumentalness, and many others. An advantage of using the Spotify API was the ability to create a custom and easily accessible data set shareable via a simple web link. In this way, we were able to bypass the heavy lifting of creating an audio analyzer, and could now focus our attention entirely on the interpretation of the traits at hand.

With Spotify API data in hand, we were able to construct a streamlined data set consisting of four distinct Genres: Classical, 80's, Hip Hop, and Metal. By reducing the target classification to these four categories, implementation of a machine learning algorithm was much more feasible. In order to fetch the Spotify API, the Spotipy Python package was utilized, which was specifically created to interface with Spotify data scraping. By using Spotipy, our Python script was able to interpret a playlist link and return data files containing song title, artist name, audio features, and most importantly genre. It is important to note that Spotify's built in Genre tags are tied to the artist rather than the songs themselves, which does not necessarily affect the intention of our algorithm, but it was a detail which we felt was mildly interesting. The audio features which Spotify provides are as follows: Danceability, Energy, Key, Loudness, Mode, Speechiness, Acousticness, Instrumentalness, Liveness, Valence, Tempo, and Duration. In order to visualize which features would provide the most impact to our algorithm we decided to generate various scatter plots using MATLAB, as seen in figures 1 through 8 in the appendix. From the plots it was clear certain features were better than others, and our decision to exclude Key, Loudness, Mode, Liveness, Tempo, and Duration from the model was based on their lack of uniqueness. In addition to removing features due to lack of uniqueness and avoiding the curse of dimensionality, we reduced the dimensionality of our model by more than half.

Because musical genres are largely used to identify a group of similar sounding songs, the choice of the KNN algorithm as a mode of classification was more than obvious for our application. The KNN model utilizes a 'nearest-neighbor' approach to classification, and by analyzing the distribution and distance of data the algorithm is able to classify and predict the genre of a given playlist. From here, we opted to use MATLAB and its built in machine learning functions to train and test our data. Using the 'fitcknn' function in MATLAB, which fits a KNN classifier to our given set of features, we were able to use KNN to train our model with a default of 1 as our number of neighbors. This resulted in a test accuracy of 87.5%.

Before we went on to modify parameters of our model, we were curious upon how the selection of our number of neighbors (K-values) would affect our testing accuracy. Fundamentally, the KNN algorithm is dependent on the K-value. From figure 6, we can see how the algorithm changes as we increase K. For example, at K=3, the algorithm finds 3 nearest neighbors and based on the total number of classes, it chooses a class based on whichever class has the largest number. From preliminary research on choosing the number of neighbors for the KNN algorithm, we stumbled upon an article on Towards Data Science, which

Figure 6: Varying K-Values of KNN



discusses recommendations on choosing an optimal K-value. Generally, there is no

predetermined method in choosing a K-value; however, the article does recommend creating a plot of K-values versus the corresponding error rate. Doing this, will allow us to determine the K-value that gives the least amount of classification error. Applying this method, we obtained the following plot (Figure 9 in appendix) . As we can see, K=3 gives us the lowest misclassification error, hence we choose K=3 as our number of neighbors. With a K-value of 3, we obtained a test accuracy of 90%.

Additionally, we also wanted to explore how our model changes with different distance metrics. By default, the 'fitcknn' function uses Euclidean distance to calculate the distance between classes. However, we wanted to also test Chebychev and Minkowski distance metrics. [insert equations of minkowski, euclidean, cosine]. From figures 10 through 16 in the appendix, we can see that misclassification error vs. K-value varies only slightly but overall remains the same as in Euclidean distance. Correspondingly, accuracy of our models across Chebychev and Minkowski distance metrics also remain closely similar to that in Euclidean distance.

Summary/Conclusion

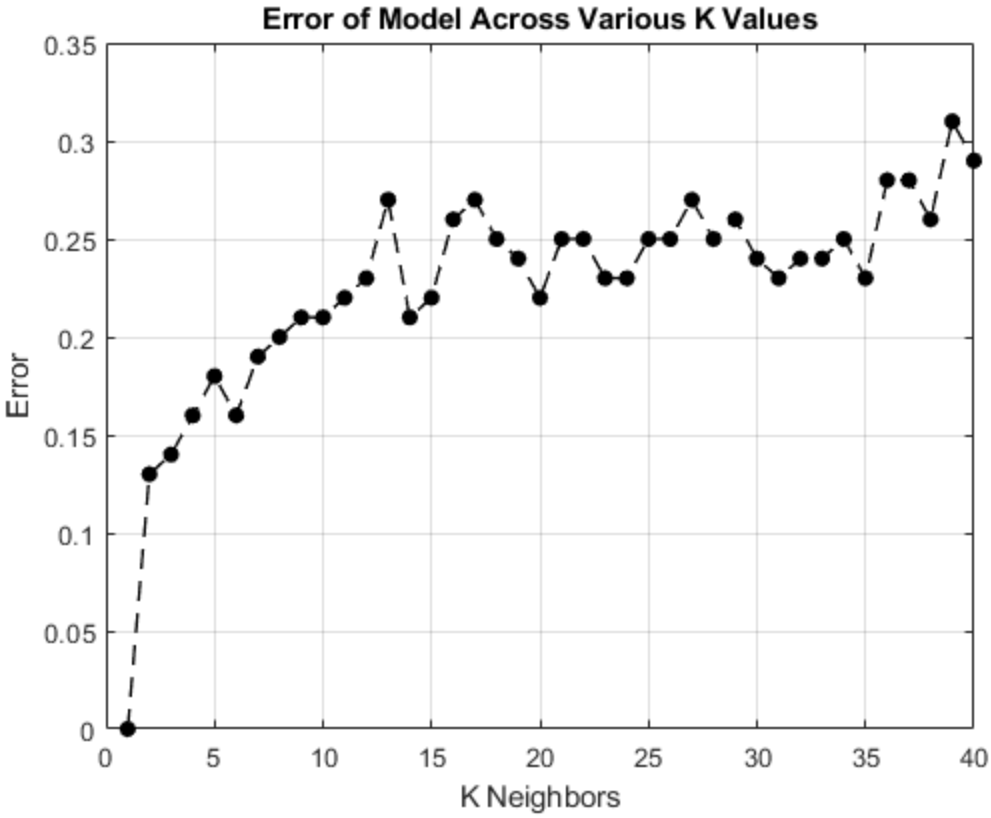
Despite our best efforts, admittedly there are many aspects of this project which can be improved. The first and most obvious point of improvement would be the data set, which would ideally be anywhere from 5-10 times larger in order to ensure a much more accurate model. The truth is a training size of 100 songs is sufficient for the proof of concept of our intentions, but it is not enough to have confidence in the robustness of our results, despite promising preliminary tests. For future reference, it would be worth investigating the utilization and limitation of open source API's, and had we known Spotify capped the bandwidth of data collection we would have spent more time on the Python collection script to guarantee an adequate amount of data. Another aspect which deserved more attention was the lack of Genres. Perhaps we limited ourselves too much to only classify four genres, as the applications of such a system is minimal. Knowing what we know now our future plan is to increase the scope of our project to be more ambitious with its performance.

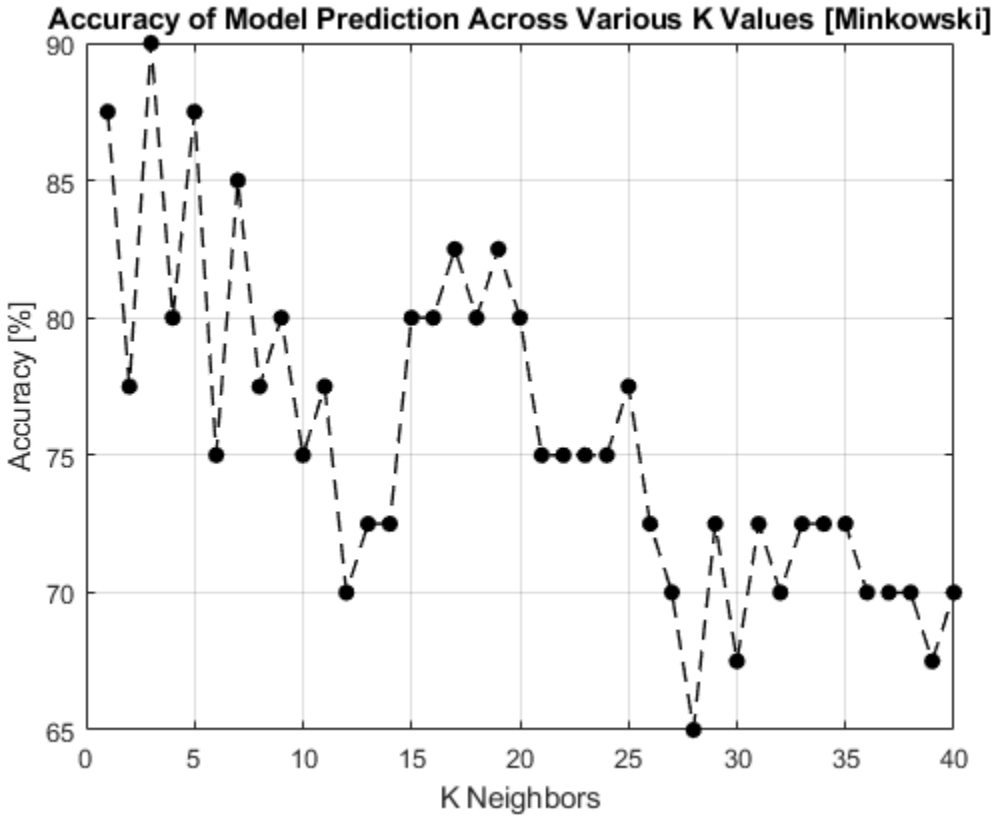
References

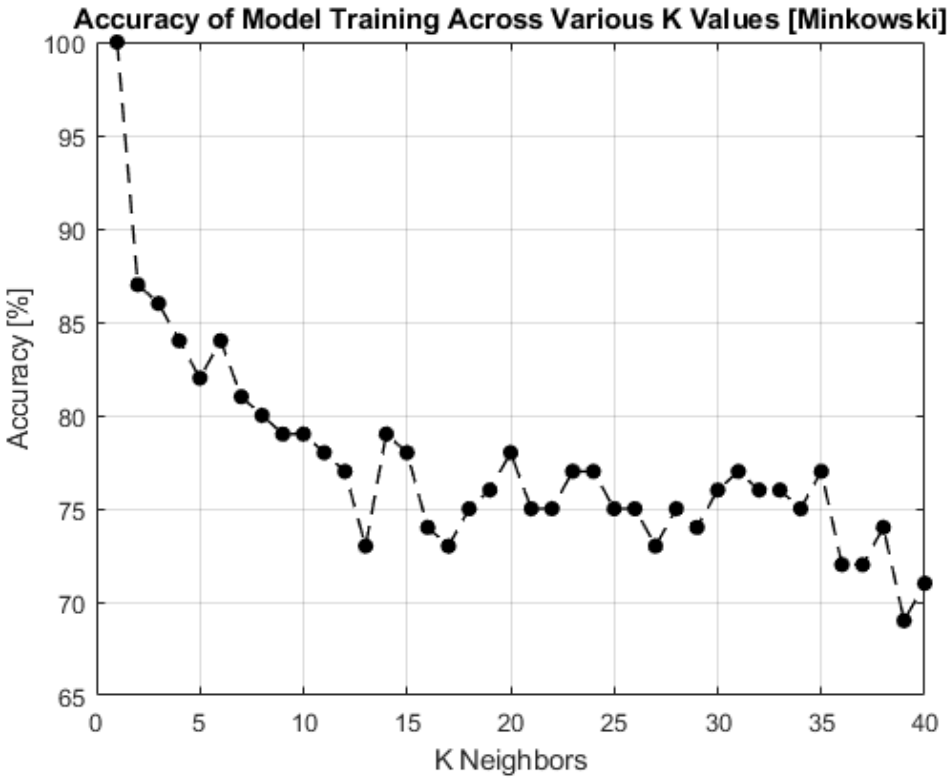
[1] Band, Amey. "How to Find the Optimal Value of K in KNN?" *Medium*, Towards Data Science, 10 Apr. 2021, towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb.
[2]

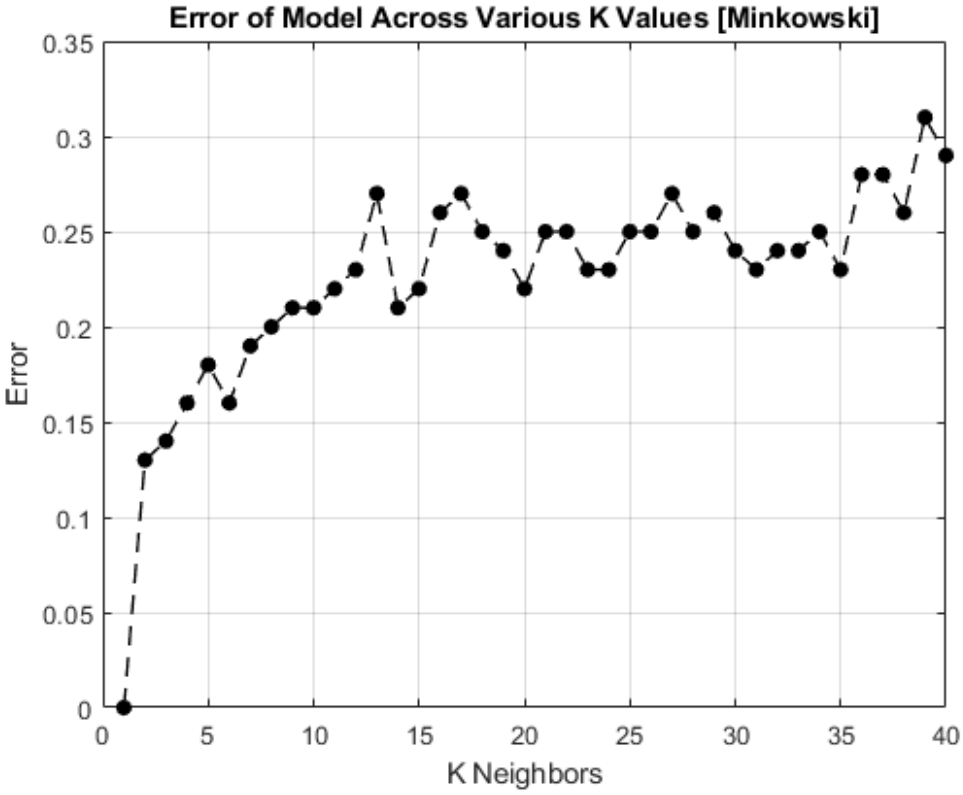
Appendix

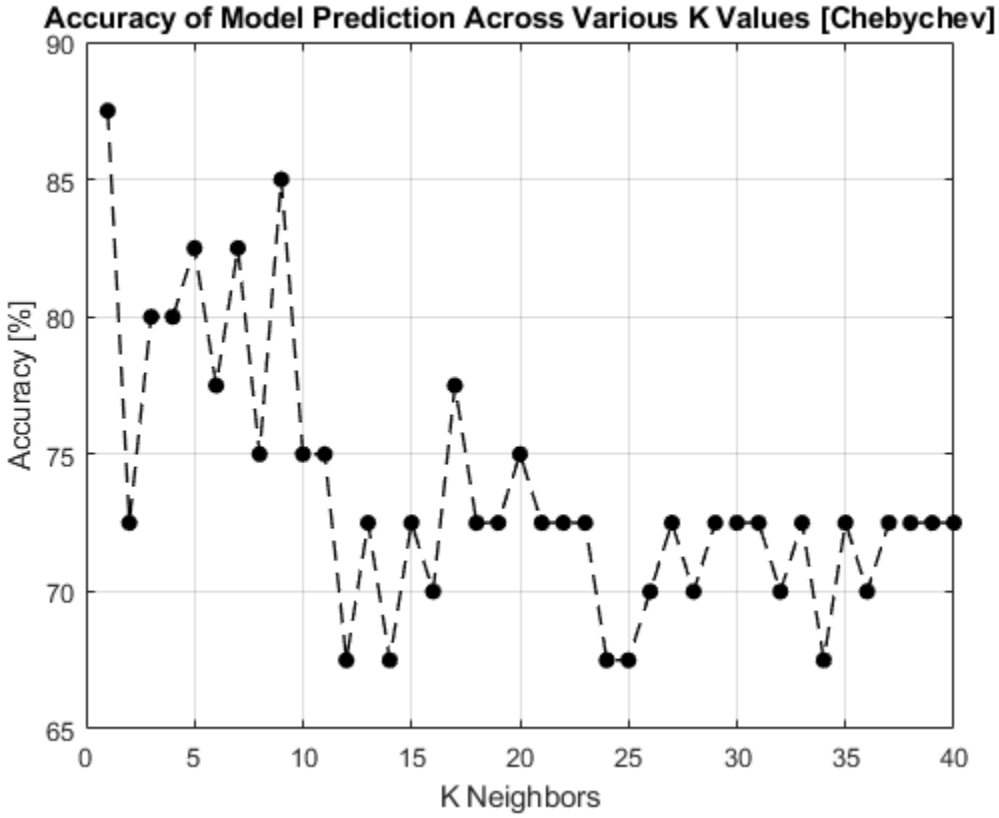

Accuracy of Model Training Across Various K Values
Accuracy [%]
K Neighbors
100
95
90
85
80
75
70
65
0
5
10
15
20
25
30
35
40

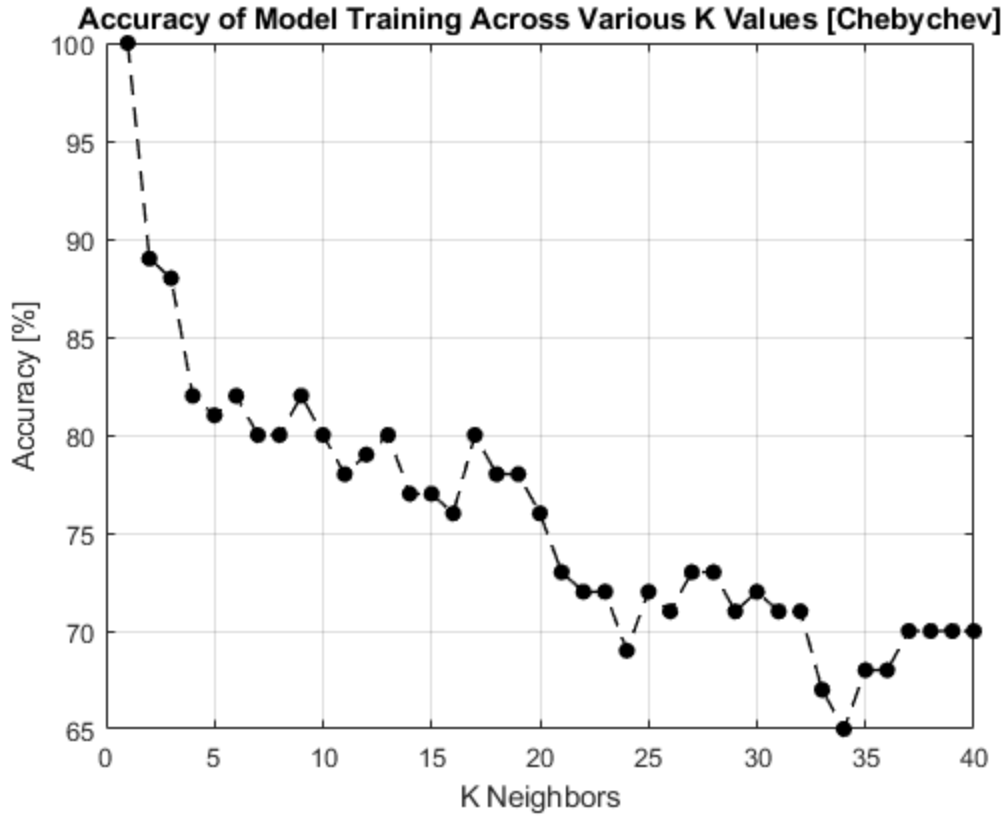

Error of Model Across Various K Values
Error
K Neighbors
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

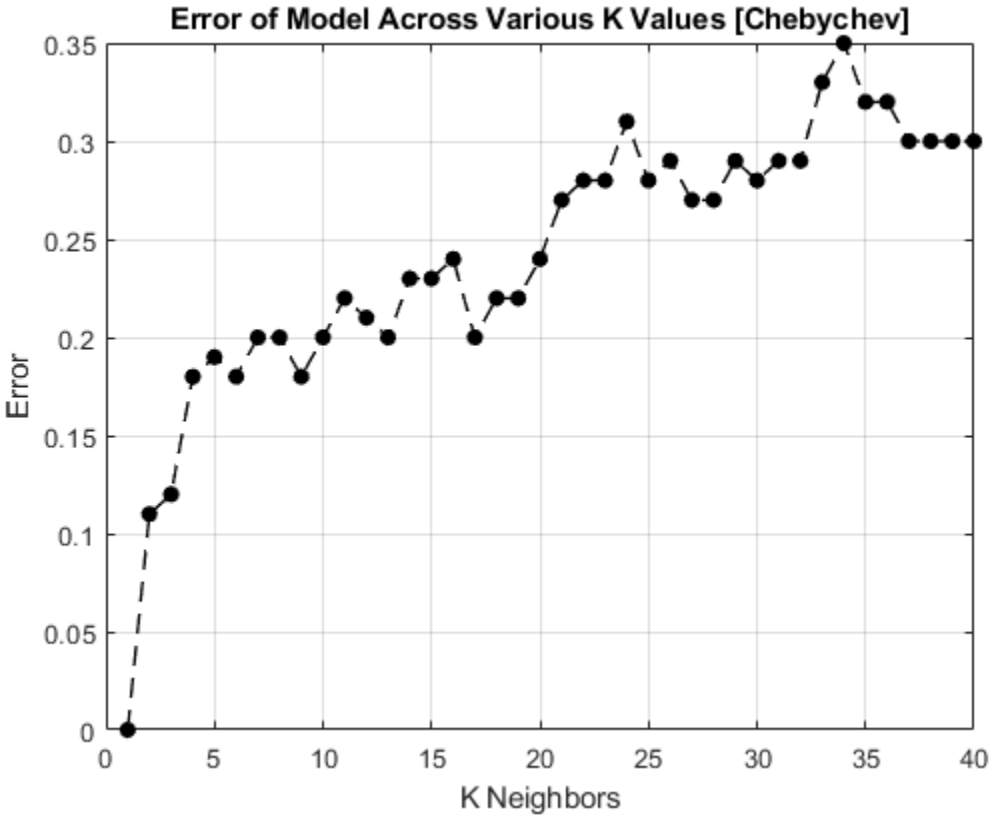

Accuracy of Model Prediction Across Various K Values [Minkowski]
Accuracy [%]
K Neighbors
90
85
80
75
70
65
0
5
10
15
20
25
30
35
40

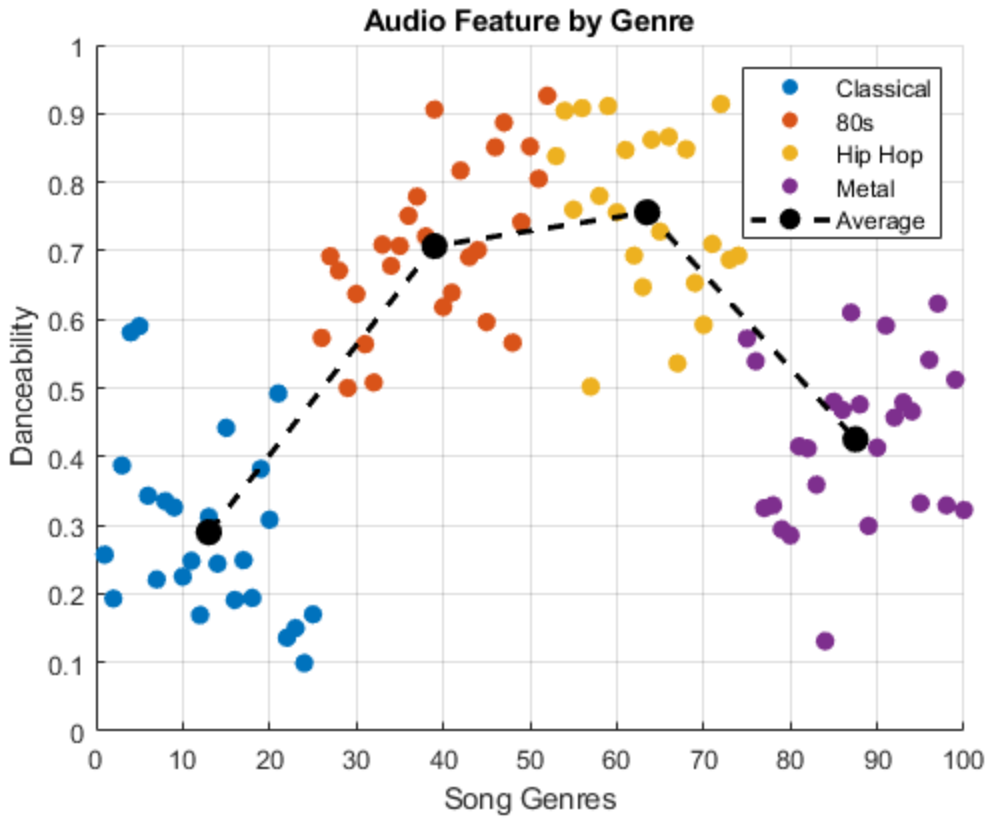

Accuracy of Model Training Across Various K Values [Minkowski]
Accuracy [%]
K Neighbors
100
95
90
85
80
75
70
65
0
5
10
15
20
25
30
35
40

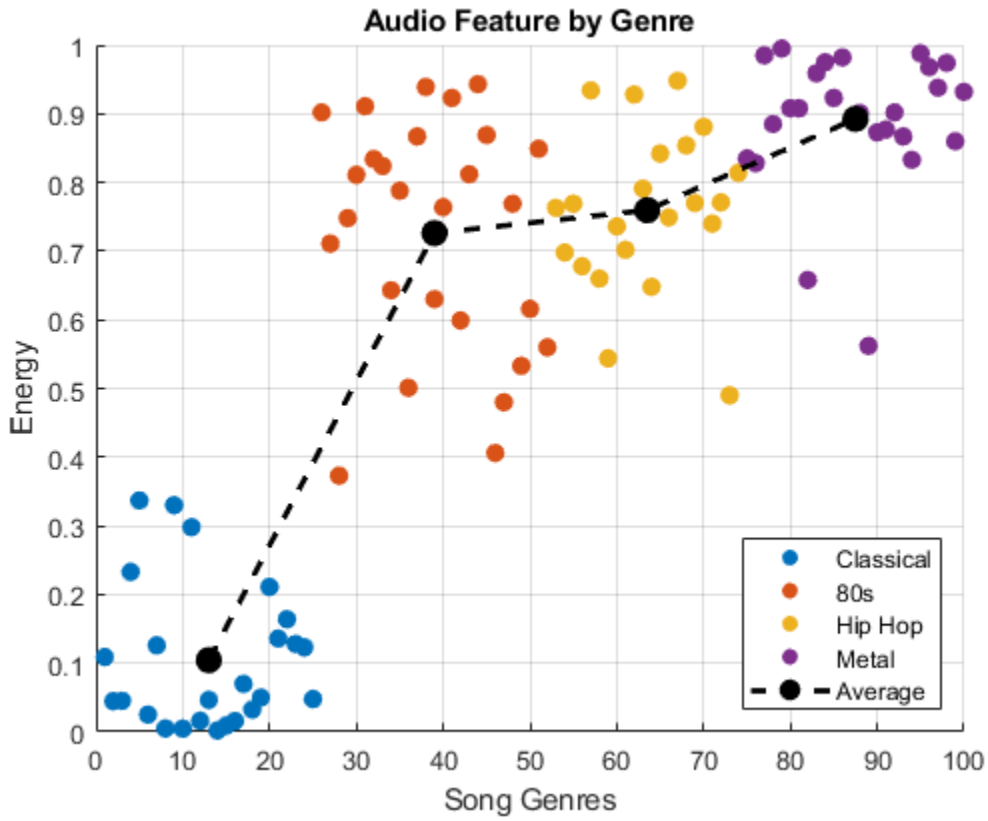

Error of Model Across Various K Values [Minkowski]
Error
K Neighbors
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

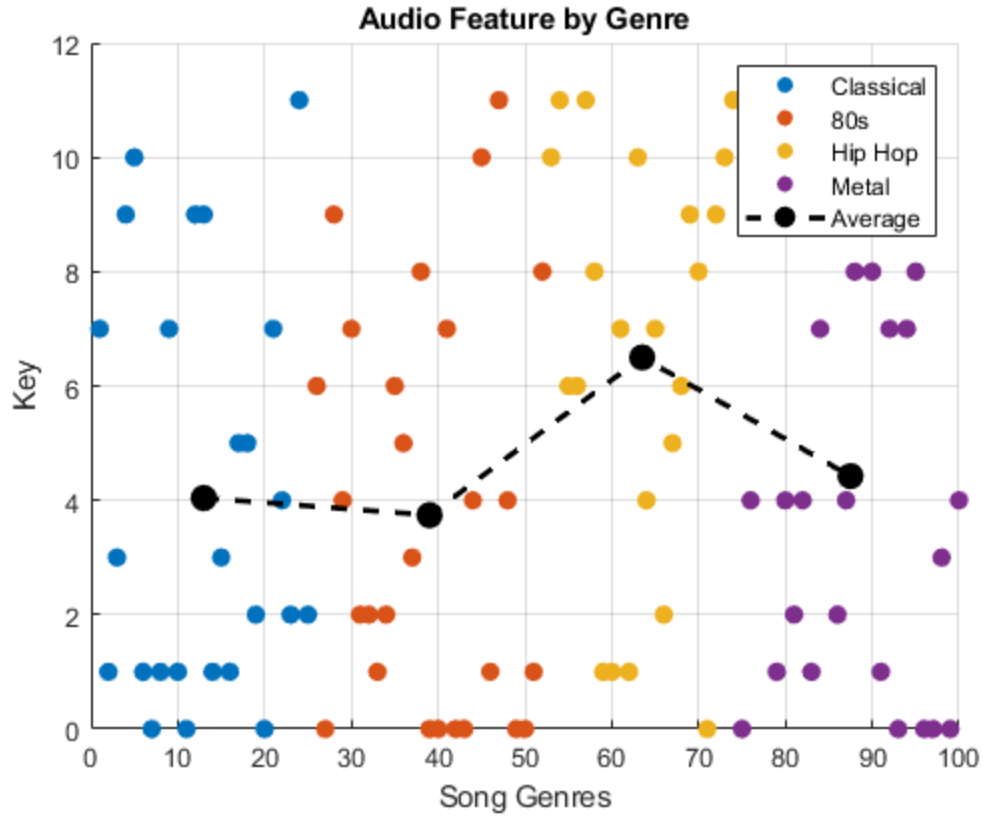

Accuracy of Model Prediction Across Various K Values [Chebychev]
Accuracy [%]
K Neighbors
90
85
80
75
70
65
0
5
10
15
20
25
30
35
40

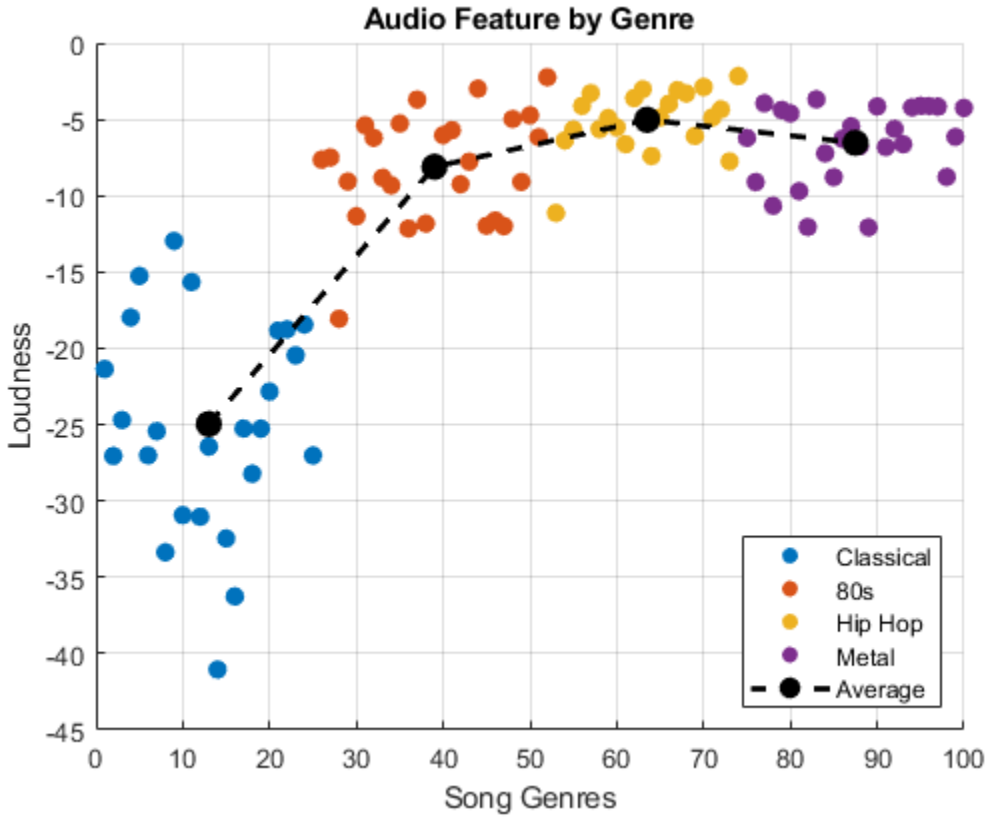

Accuracy of Model Training Across Various K Values [Chebychev]
Accuracy [%]
K Neighbors
100
95
90
85
80
75
70
65
0
5
10
15
20
25
30
35
40

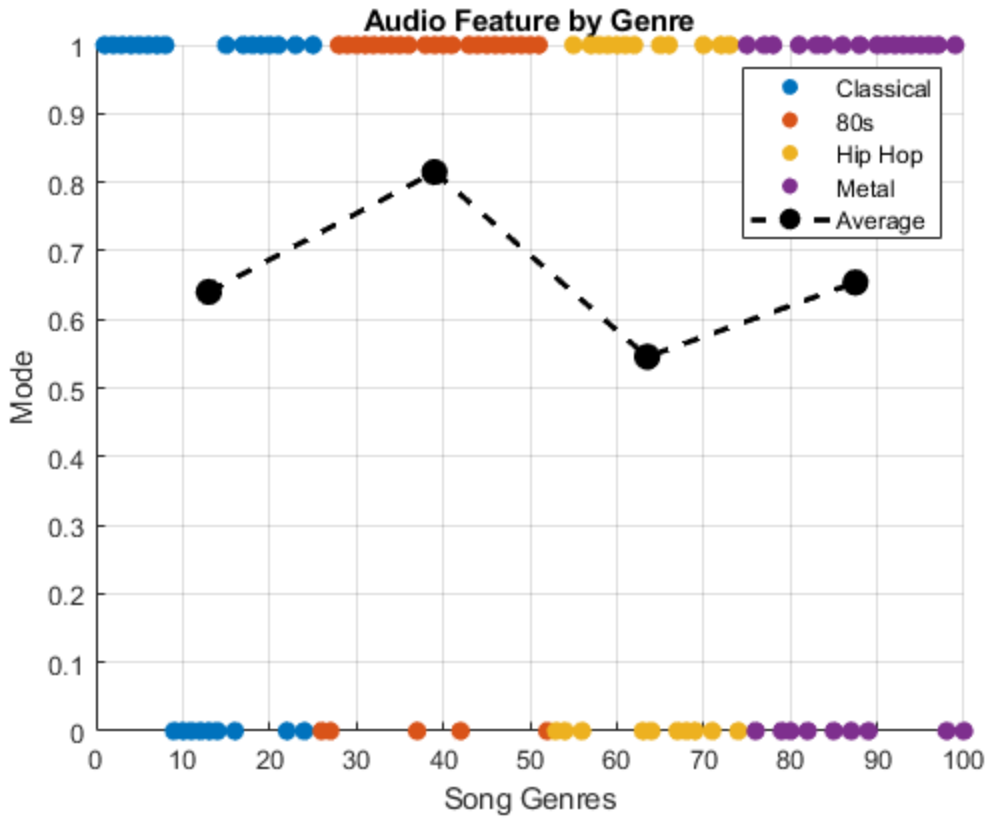

Error of Model Across Various K Values [Chebychev]
Error
K Neighbors
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40

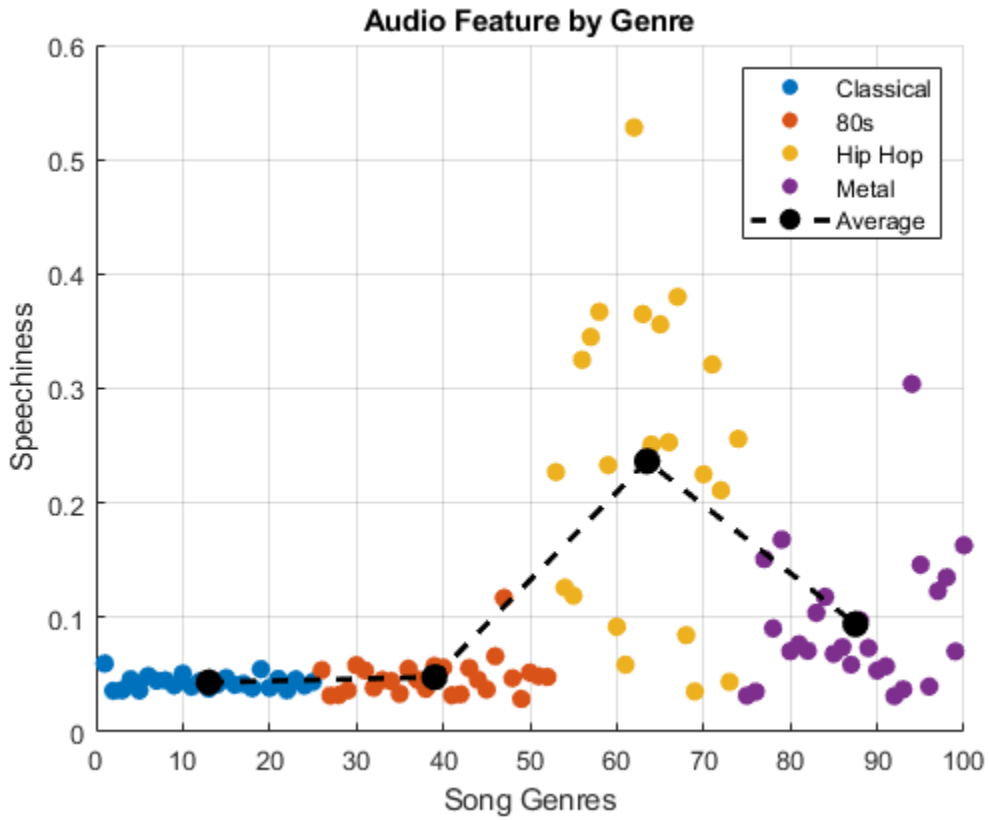

Audio Feature by Genre
Danceability
Song Genres
Classical
80s
Hip Hop
Metal
Average
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
60
70
80
90
100

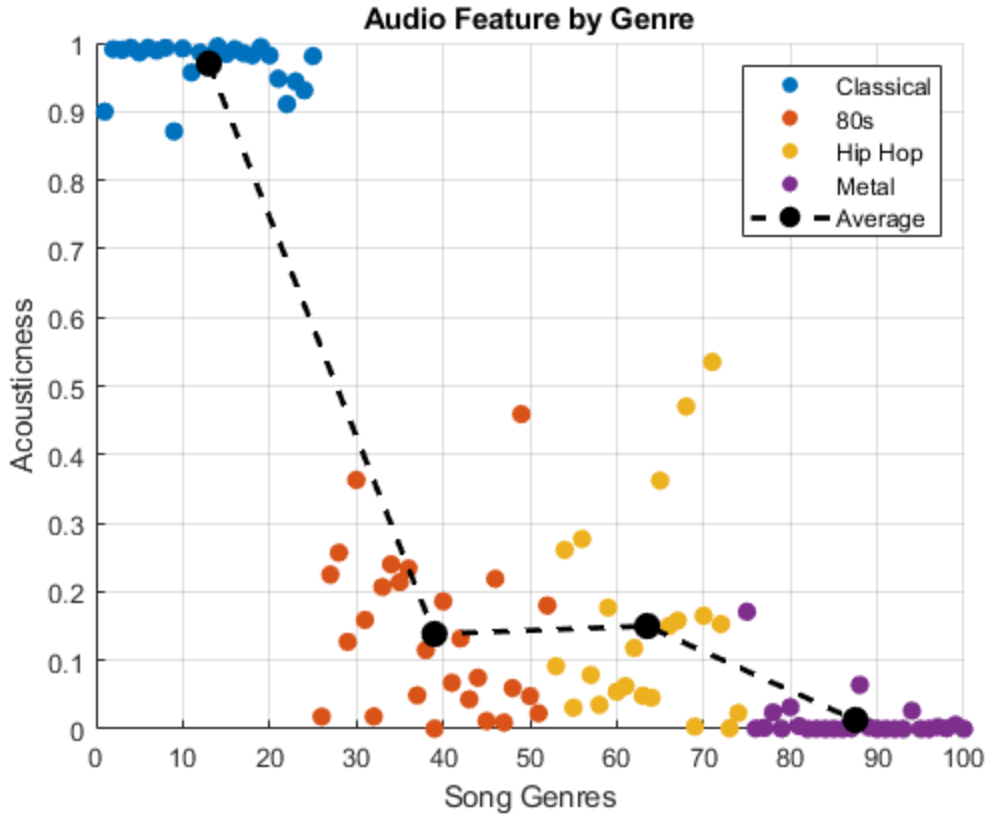

Audio Feature by Genre
Energy
Song Genres
Classical
80s
Hip Hop
Metal
Average
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
60
70
80
90
100

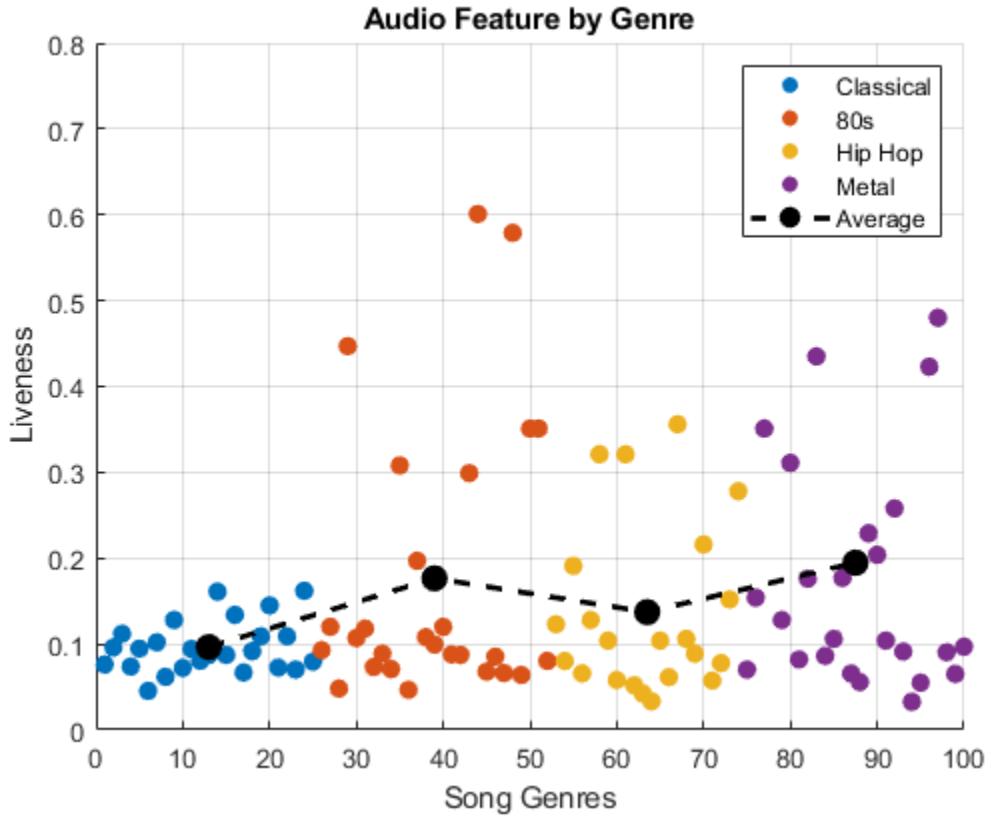

Audio Feature by Genre
Classical
80s
Hip Hop
Metal
Average
Key
Song Genres
0
2
4
6
8
10
12
0
10
20
30
40
50
60
70
80
90
100

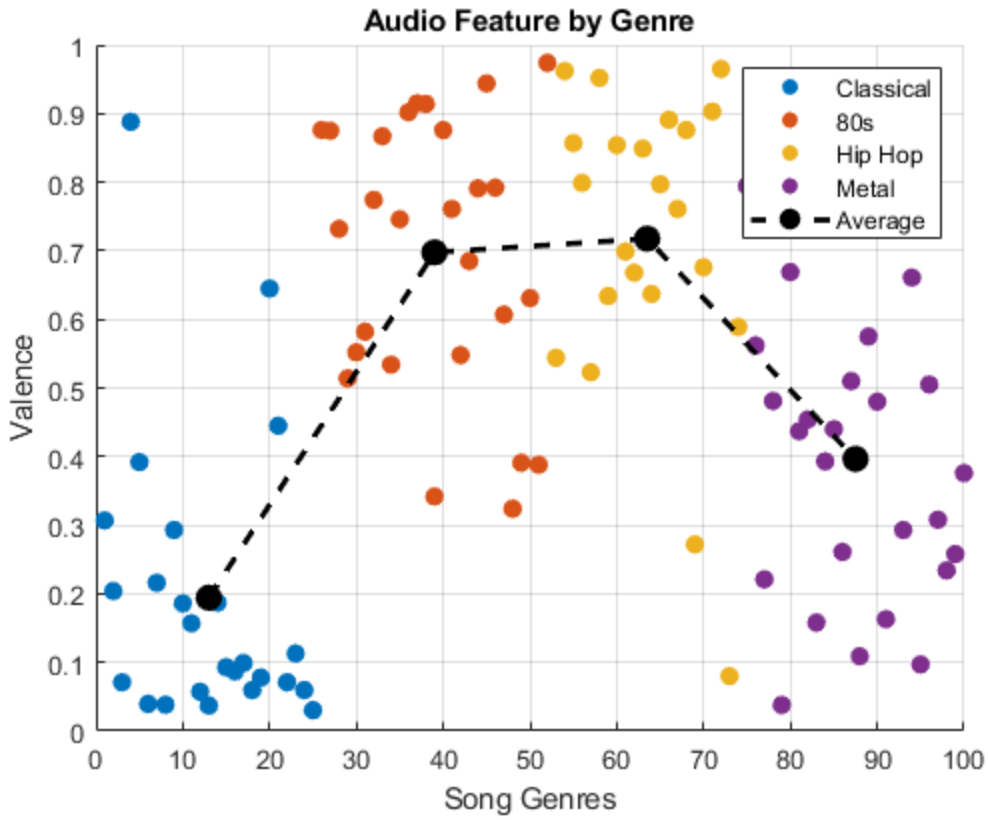

Audio Feature by Genre
Classical
80s
Hip Hop
Metal
Average
Loudness
Song Genres
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
0
10
20
30
40
50
60
70
80
90
100

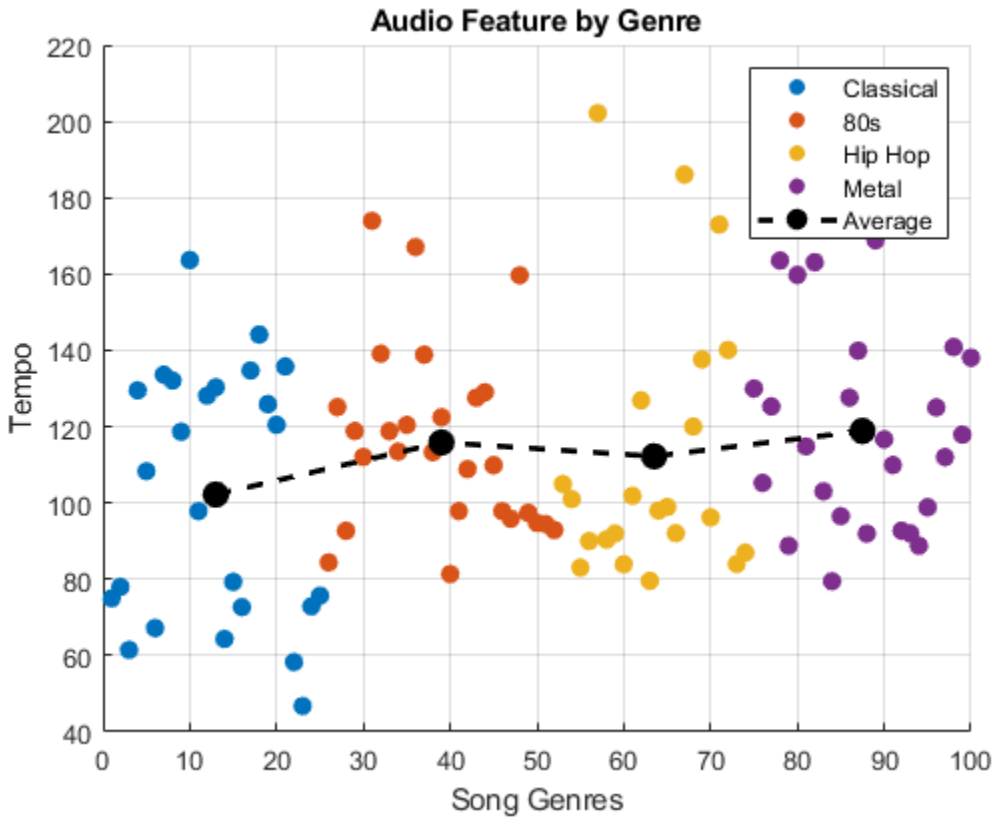

Audio Feature by Genre
Classical
80s
Hip Hop
Metal
Average
Mode
Song Genres
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
60
70
80
90
100


Audio Feature by Genre
Classical
80s
Hip Hop
Metal
Average
Speechiness
Song Genres
0
0.1
0.2
0.3
0.4
0.5
0.6
0
10
20
30
40
50
60
70
80
90
100


Audio Feature by Genre
Classical
80s
Hip Hop
Metal
Average
Acousticness
Song Genres
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
60
70
80
90
100


Audio Feature by Genre
Classical
80s
Hip Hop
Metal
Average
Liveness
Song Genres
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
10
20
30
40
50
60
70
80
90
100


Audio Feature by Genre
Valence
Song Genres
Classical
80s
Hip Hop
Metal
Average
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
60
70
80
90
100


Audio Feature by Genre
Tempo
Song Genres
Classical
80s
Hip Hop
Metal
Average
40
60
80
100
120
140
160
180
200
220
0
10
20
30
40
50
60
70
80
90
100

References

https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html
https://medium.com/@FinchMF/praise-questions-and-critique-spotify-api-38e984a4174b
https://towardsdatascience.com/classifying-song-genre-using-spotifys-built-in-features-vs-extracting-my-own-a4d5fe448948

https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761