Sina Astani

*Cal State Long Beach*

*CECS 551: Advanced AI*

**Music Genre Classification:**

**A Comparative Analysis of Machine Learning Techniques**

*Abstract*—Music genres are a way to categorize similar pieces of music. They are significant to modern music recommender systems and have been an active topic of discussion in music information retrieval. In this paper, I perform a detailed study of this topic and present an analysis of various methods used in the task of automatic music genre classification. I will attempt to compare the performance of three common machine learning algorithms: SVM, Gaussian Naive Bayes, and kNearest Neighbors as applied to the task of classifying the genre of various samples of audio. Python code was prepared to sort songs into five different genres and then preprocess them prior to feature extraction. Following this step, each of the classifiers was trained. Then, their performance was evaluated using the standard statistical methodology of cross validation.

# **Introduction**

With the advent of digital audio formats and portable music players many older means for music consumption have died, giving rise to streaming music services. Applications such as Pandora, Spotify, and Soundcloud have become very popular. These and similar song recommender systems have turned music classification into an interesting and useful topic of study. In recent years, the amount of data available related to individual songs has increased tremendously, promoting further

research into music genre classification and related topics in music information retrieval.

# **II. Related Work**

The compilation of the GTZAN dataset, a selection of 1000 audio tracks over 10 different genres, and G. Tzanetakis’ seminal paper titled “Musical Genre Classification of Audio Signals” in 2002 lead to a significant amount of research of this task. The GTZAN dataset was compiled by Tzanetakis from various CD, radio, and microphone recordings compromising of the Blues, Classical, Country, Disco, Hip Hop, Jazz, Metal, Popular, Reggae, and Rock genres. It has been used in numerous studies involving music analysis and information retrieval. Prior to his research and the availability of this dataset, genre categorization was done manually.

Tzanetakis’ paper, as well as his subsequent papers, such as “Factors in Automatic Musical Genre Classification of Audio Signals” [8] have explored the usefulness of MFCCs, or mels frequency cepstral coefficients as features for music classification. While initially MFCCs were predominately used for speech recognition, he proposed that they could also be used as features for music genre classification. Following his suggestion, MFCCs have been used alone or coupled with other features for various problems in music information retrieval.

While the GTZAN dataset was initially important, the availability of the Million Song Dataset, a collection of metadata and audio analysis features, such as loudness, timbre, and pitch for a million popular music songs made large scale studies possible [7]. However, in the past few years research into this topic his plateaued.

I have decided to reconsider this topic from a practical perspective and see how my results stand next to decades old research. In the results section, I will compare my results to those of several related studies.

# **Approach**

My main goal of this project was to implement three different classification algorithms: kNN, SVM, and Naive Bayes and compare their performance when used for the task of music genre classification. My approach to solving this problem can generalized as follows:

*Dataset generation*: Create a dataset spanning five different genres

*Feature extraction*: Extract the features for each member of the dataset

*Classification*: Build models using the dataset and each classification algorithm

*Validation*: Using cross-validation, confirm the results

The first step was to find or create a dataset appropriate for the problem. Although the GTZAN dataset suited my needs, I found it dated, as it did not contain newer, more contemporary genres and songs. The Million Song Dataset was appealing, but it only consisted of textual data for each song and did not include any audio samples. Due to the limitations of both datasets, I compiled a set of six hundred different audio tracks from my personal collection, all of which were 320kbps lossy compressed mp3 files. I organized the files into separate directories according to their genre, tagging each song with the appropriate label.

Following my compilation of the dataset, I performed some initial preprocessing of audio and converted each mp3 file to a thirty second mono 44.1 kHz wav file. Given that the duration of most songs is roughly three minutes, I kept the 30 seconds of audio after the one and a half minute mark of each song. Each audio signal was then normalized, limiting the min/max decibel values by ensuring that the amplitude of any segment was not below -32 dB or higher than -18 dB; this was performed to avoid any clipping which would later occur when flattening the signal to a mono signal.

Then I performed feature extraction on each wav file by extracting the mfccs using the Python scikit.talkbox library. I combined the features for songs of the same genre into one csv file. Following this process, I had five different files (one for hip hop, one for rock, one for electronic, one for jazz, and one for country) each representing all the features for each class.

Finally, I used all of these feature files to train and test each of my three classifiers.

# Dataset

Most songs in the dataset were picked from my personal collection. In order to have a high amount of variance in my sample set, the songs I selected for each genre were from different artists. Since I have limited knowledge of and exposure to country music, I choose popular country songs from Spotify playlists and added them to my dataset. Using a Python script I developed, I obtained the artist names, song titles, and album names from the ID3 tags (metadata containers used conjunctively with mp3 files) of each song. Most of the mp3 files I used did not have the proper genre tag or did not have a genre tag at all, so I tagged each song using the label I felt was most appropriate.

With my research, I wanted to observe how supervised learning methods would work with genres that are popular today, so I picked the following five genres: Hip Hop, Jazz, Rock, Electronic, and Country. Although I noticed that most research on music genre classification had used genres which are more distinctive, I felt my choices would more accurately represent the contemporary listener. For this reason, I omitted genres such as Classical music. I did not include the Pop genre since I speculated that many Pop songs could also be considered Hip Hop or Electronic, and thus could be problematic when training my models.

Except for the Jazz genre, a majority of the songs in the dataset are from the past decade. The Rock genre contains music which would be considered “Classic Rock” today, such as “Feel Flows” from the Beach Boys (1971) and “Time” from Pink Floyd (1973). However, the Hip Hop/Rap and Electronic genres consist of songs from the last ten years such as “ELEMENT” from Kendrick Lamar (2017) and “XMAS\_EVET10 (Thanaton3 Mix)” from Aphex Twin (2014). A full list of all the songs as well as the features for each song in the dataset can be found on GitHub: <https://github.com/sastani/GenreClassifier>.

# Feature extraction

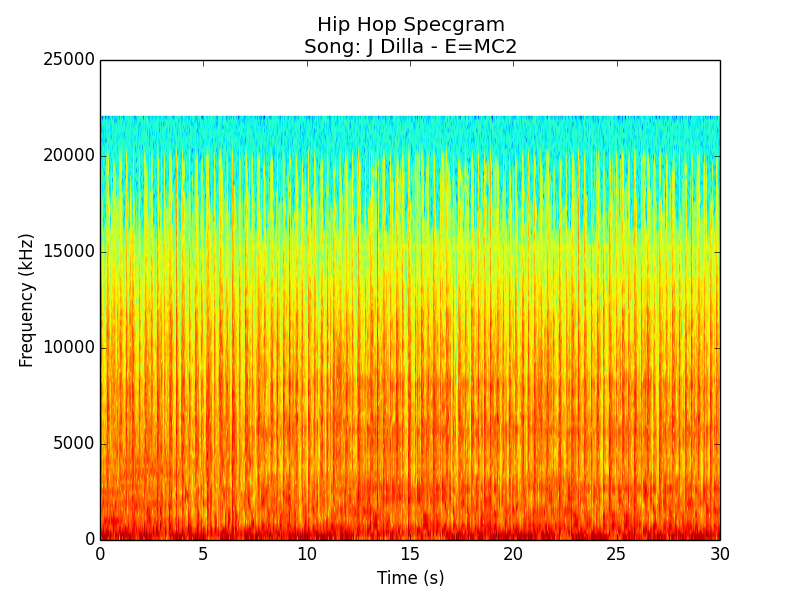
As mentioned previously, I used melfrequency cepstrum coefficients as features for each of my learners. Theyareused in describing the shape of short segments of audio called frames. Each frame contains amplitude or loudness information at that particular point in time within the song. Mfccs represent the timbre, or musical “texture” or type of sound for a frame. Timbre can be described more accurately as the “perceived sound quality that distinguishes different types of musical instruments or voices.”

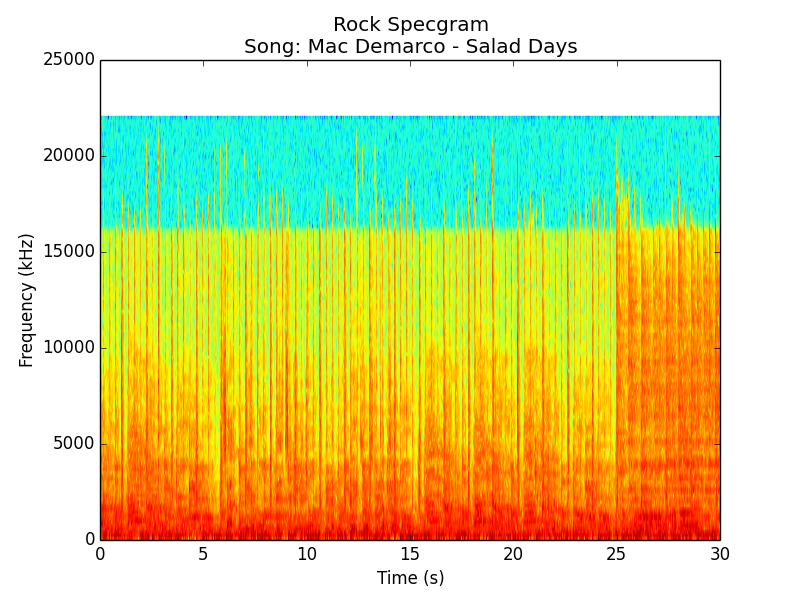
The process of calculating MFCCs for a given waveform can be described as follows, after splitting signal into short frames [5]:

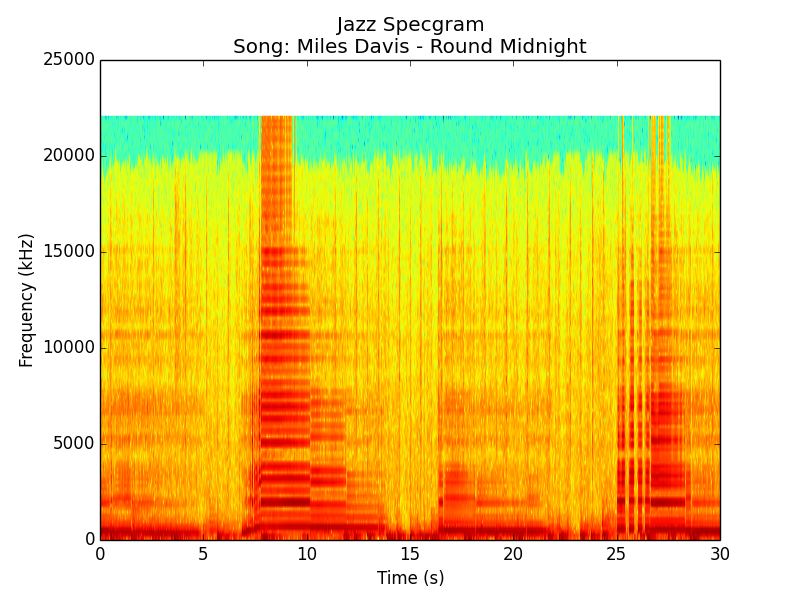
1. Take the short time Fourier transform (STFT) of a frame
2. Smooth frequencies and map the spectrum above onto mel scale
3. Take logs of powers at each mel frequency
4. Take the DCT, or discrete cosine transform, of the mel log powers
5. Keep the remaining specified coefficients

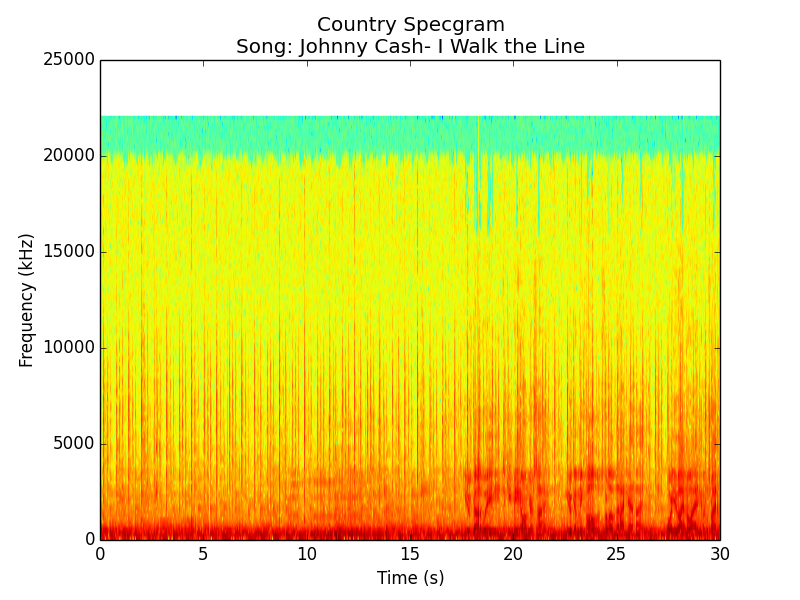
Spectrograms were plotted for five different songs, one from each genre. A spectrogram represents the spectrum of frequencies of a signal and show how these frequencies vary over time. They are produced by taking the short time Fourier transform (STFT) of a signal [3].

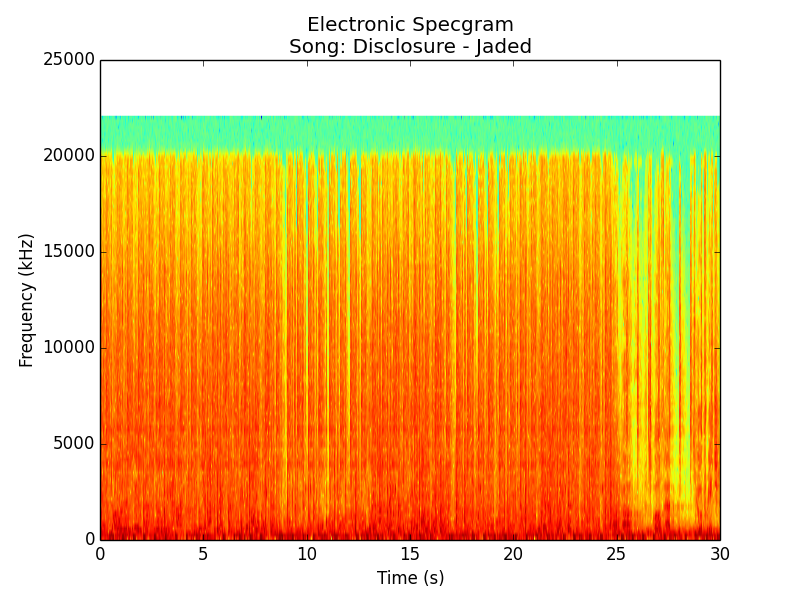
The following represent graphs produced after the first step of MFCC calculation.











After performing a STFT, the frequencies are converted from the Hertz scale to the mels scale, followed by a few mathematical transformations [3]. I will not go into further detail regarding the remaining procedures, given that it is quite dense and a topic of discussion on its own. However, when the process is complete, thirteen MFCCs are produced for each frame where the first MFCC is a constant that can be discarded [6]. I kept 13 mfccs for each frame of a 30 second sample of each song. Then I converted each 2D array (an array consisting of arrays for each frame of the 30 second sample) for each song into a 1D array, which represented all MFCCs for the entire 30 seconds.

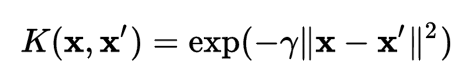
# Classification

Once I constructed a feature set for all my songs, I started building my classifiers. I used Scikit-learn, a common open source Python machine learning library to implement my classifiers. Scikit-learn has implementations of many different machine learning classification and regression algorithms such as support vector machines, random forests, k-means, as well as others. For this project I focused on three algorithms: kNearest Neighbors, Naïve Bayes, and Support Vector Machines. Specifically, I focused on SVM because I extensively studied this algorithm in this course. I hypothesized that SVMs would perform most accurately for this task, given that SVMs typically work very well with high dimensional data, or data with a large amount of attributes or features.

#### Support Vector Machines

Support vector machines work by constructing decision surfaces given a certain dataset, where each member, or point, of the dataset is represented as a vector. First, hyperplanes are determined based on the position of the support vectors. The goal is then to find a decision surface which maximizes the margin between supporting hyperplanes. These decision surfaces are linear in the case of linear SVMs, but they can be nonlinear by using the kernel trick which uses a kernel function to map the input space into a higher dimensional feature space where the inputs can be separated. This aspect of SVMs, as well as their resistance to overfitting, makes them attractive for many problems where separating the data is a complex problem.

In this project, I utilized a Support Vector Machine with a radial basis kernel. The function for an RBF kernel is as follows:



In this formula, represent feature vectors and represents a kernel coefficient. The value multiplied by gamma is the squared Euclidean distance between feature vectors. For my models, I used a which was equal to .

Since each input consisted of an array of 13 MFCCs for each sample of audio, and there were 8,268 frames for every audio file, the number of features for each input was equal to 107,484.

#### KNearest Neighbors

Unlike SVM, the KNearest Neighbors algorithm works in the feature space and is non-parametric. kNN is a simple algorithm which uses the k nearest training examples or data points to find the label of the current data point. Typically, k is chosen to be an odd value to avoid cases where there could be equal training examples of two or more classes. For my classifier, I chose a value of five for k. Distance dependent weights can be used in this method such that closer training examples are weighted more heavily [9]. However for my specific classifier I used uniform weighting, so each training example in the neighborhood of the current point was weighted equally.

#### Gaussian Naïve Bayes

Naïve Bayes is a probabilistic classifier built on the idea of conditional probability. Essentially, this is the probability that some event will happen given that some other event has happened. It is an important part of Bayes rule, which uses conditional probabilities to determine the likelihood of some event. In our case this event is some class or genre. In a Naive Bayes model each input serves as evidence for some outcome or event, and influences the probability of some input being labeled as such. The class that has the highest probability given our training examples determines our label.

# **Results**

Performance varied between the classification techniques. Each classifier was evaluated on its ability to classify a test sample in one of the five classes.

Interestingly enough, Support Vector Machines

were most accurate overall, while Naïve Bayes performed almost as well, trailing just behind in accuracy. The kNearest Neighbors algorithm performed quite poorly in comparison, and its performance seemed to vary widely depending on the training and testing sets that were provided. Ultimately kNN did not seem well suited for this task, given its simplicity and that songs of different, yet closely related genres can have a similar set of features.

TABLE 1

Average Accuracies of Classifiers

|  |
| --- |
| Model 15 Seconds 30 Seconds |
| KNN 39.37% 39.33%  GNB 49.59% 49.00%  SVM 54.06% 53.67% |

Average results using both 15 and 30 second audio samples

Table 1 shows classification results for all three of our classifiers. Results were calculated using 10 fold stratified cross validation, such that 9 out of 10 partitions of the entire dataset are used for training and 1 partition is left for testing for each of the 10 rounds of testing. For each iteration, stratified partitions were created so that both training and test sets had equal numbers of members from each class.

Classifiers were tested using both 15 and 30 second audio samples. There seemed to be no significant difference between using 15 seconds versus 30 seconds of audio and thus twice the amount of frames. The additional information provided to each classifier served to provide no additional benefit, as fifteen seconds proved to be enough time to distinguish classes. This could be due to the fact that spectral features do not vary significantly beyond fifteen seconds. Given that many songs repeat the same motifs during such short segments, this is highly likely.

Tables 2, 3, and 4 show detailed performances for each of the classifiers in the form of confusion matrices. The confusion matrices from ten separate runs were combined into one confusion matrix for each classifier. The horizontal axis is the actual label and the vertical axis is the predicted label. The bottom row represents the percent correct.

TABLE 2

KNearest Neighbors

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Country | Electronic | Hip Hop | Jazz | Rock |
| Country | 14 | 3 | 2 | 5 | 7 |
| Electronic | 20 | 79 | 56 | 37 | 14 |
| Hip Hop | 0 | 0 | 0 | 0 | 0 |
| Jazz | 25 | 14 | 7 | 58 | 14 |
| Rock | 61 | 24 | 55 | 20 | 85 |
| Accuracy | .1167 | .6583 | 0 | .4833 | .7083 |

TABLE 3

Gaussian Naive Bayes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Country | Electronic | Hip Hop | Jazz | Rock |
| Country | 51 | 15 | 13 | 19 | 27 |
| Electronic | 6 | 37 | 21 | 15 | 9 |
| Hip Hop | 4 | 29 | 68 | 11 | 7 |
| Jazz | 15 | 23 | 5 | 66 | 5 |
| Rock | 44 | 16 | 13 | 9 | 72 |
| Accuracy | .425 | .3083 | .5667 | .55 | .6 |

TABLE 4

RBF Support Vector Machine

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Country | Electronic | Hip Hop | Jazz | Rock |
| Country | 51 | 15 | 13 | 19 | 27 |
| Electronic | 6 | 37 | 21 | 15 | 9 |
| Hip Hop | 4 | 29 | 68 | 11 | 7 |
| Jazz | 15 | 23 | 5 | 66 | 5 |
| Rock | 44 | 16 | 13 | 9 | 72 |
| Accuracy | .55 | .4333 | .6083 | .5167 | .575 |

As can be seen in table 1, kNearest Neighbors failed at accurately determining the label of certain genres.

While all three classifiers had some difficulty differentiating between hip hop and electronic genres, kNN simply could not label hip hop songs at all, while the other classifiers confused electronic songs as hip hop. kNN confused a majority of hip hop songs as electronic or rock.

On the other hand, kNearest Neighbors performed well on electronic and rock genres.

It performed the worst of all three classifiers on country, where it exhibited a 11.67% success rate accurately predicting 14 out of 120 correct.

Both Naive Bayes and SVM classifiers performed well with respect to Hip Hop and Rock; Naïve Bayes models accurately predicted Hip Hop songs 56.67% of the time and Support Vector Machine 60.83% of time.

# **Discussion**

Mels frequency cepstral coefficients seem to be appropriate features for music genre classification. It is likely, however, that our results would improve if we included another type of feature for each song in our dataset. Adding features such as loudness and tempo, as measured in beats per minute, could have possibly improved results [1]. Lyrical content would not have helped the results as significantly; while adding lyrical content to timbre features has been shown improve performance for Hidden Markov Models slightly, I do not believe that it would have shown much use for genres which are mostly instrumental[1].

While it was expected that classifiers would have difficulty differentiating between Electronic and Hip Hop songs, it was not expected that Electronic songs would be as problematic as they were for Naive Bayes and Support Vector Machine learners. I expected that SVM’s ability to transform features via kernels would allow it to distinguish between inputs that cannot typically be separated linearly.

Considering the accuracies of all models with either Electronic or Hip Hop genres and the inability for any one algorithm to provide reliable accuracy for both, it can be concluded that these genres are too similar for most classification methods. I suspect that the results could be improved with a neural network. Neural nets have been shown to work well for a complex problem such as this one [3], outperforming SVM, as well as kNN.

Although the accuracy numbers I calculated were lower than those found in Tzanetakis’ seminal paper which used a Gaussian based classifier to accurately predict 90% of Hip Hop [2] and another using DAG (Directed Acyclic Graph) SVMs which accurately predicted 86.67% of four different genres [4], my results are not far off from most research. However, it is difficult to directly compare my results with previous research given my decision to use my own dataset. Additionally, while most research has shown that SVMs can be applied to this task, most new studies have focused on Neural Networks.

During my study of music genre classification, I noticed that most research used an averaged feature vector for each sample as input. Instead of using the entire set of MFCCs for a given sample, the MFCCs for all frames were converted into a mean vector. Originally, I had averaged my features as well during feature extraction. By following this methodology I found poor results, often seeing accuracies between 30-35% for any given classifier, with an RBF SVM performing the best. When I had realized that I was shrinking the amount of possible features in the learning process, essentially reducing each song to one frame, I decided to use the entire set of MFCCs for each sample. By using all MFCCs I noticed a vast improvement in performance. I propose that all possible features should be used unless computational time is important.

Additionally, an initial source of error was my omission of a crucial step immediately before training each model. I had initially forgotten to normalize my data by putting at all on one scale, leading to overfitting. I scaled all of my six hundred feature vectors so that they had zero mean and unit variance.

My research could be extended, and my future work involves including additional types of features as well as exploring other machine learning techniques. While Naive Bayes and Support Vector Machine learners can be applied to this problem reasonably well, it has lead me to consider how much performance could be improved by using a neural network with multiple layers.

References

1. Liang, D., Gu, H. and O’Connor, B. “Music Genre Classification with the Million Song Dataset”. School of Music, Machine Learning Department. Carnegie Mellon University, 3 Dec. 2011.
2. Tzanetakis, G, Essel, G, Cook, P. “Automatic Musical Genre Classification of Audio Signals”. Computer Science Department. University of Princeton, 2001.
3. T. Li, M. Ogihara, and Q. Li, “A comparative study on content-based music genre classiﬁcation,” Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2003, pp. 282–289
4. Haggblade, M., Hong Y. and Kao K. “Music Genre Classification”. Department of Computer Science. Stanford University, 2011.
5. “Mel frequency cepstrum”. In *Wikipedia*. Retrieved May 15, 2017, from  
   <https://en.wikipedia.org/wiki/Mel-frequency_cepstrum>.
6. “Mel Frequency Cepstral Coefficient (MFCC) tutorial”. Retrieved May 16, 2017 from <http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/>.
7. Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.
8. Tzanetakis, G and Li, T. “Factors in automatic musical genre classification of audio signals”. In Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA) New Paltz, NY (USA), October 2003.
9. “Music Genre Classification”. GRITs Lab. Georgia Tech. Retrieved from: <http://gritslab.gatech.edu/Pickem/wp-content/uploads/2012/12/cs6601_project_2_paper.pdf>