Music Genre Classification using Audio Feature Extraction

VINCENT JAKE K. CALAG AND KRISTOFFER DANN D. CABERTO, Ateneo de Davao University

1 INTRODUCTION

1.1 Background of the Study

Studies conducted about audio signals have become a field to which attracts many researchers for the improvement of an accurate and automatic system for the creation of different sound algorithms in extraction of complex audio signals. Techniques used in this kind of field such as Music Information Retrieval has become an active research topic which also has the need of advancement of the system.

Music has been around for so many decades and has a great impact in our society. For many years music has evolved in how it is played or constructed by different kinds of musical instruments which vary on what genre the music is. Technologies nowadays have allowed users to apply, manipulate or construct music. With these technologies processing of sound is made easy and can now let the users explore and analyze sound content of audio files. For this research the proponents decided to explore the audio signal processing field, if it is possible to automatically recognize what kind of music genre based on the content, particularly the audio features of a music file.

1.2 Research Objectives

General Objective:

• To develop a desktop application implementing the Audio signal processing with the use of different modules compatible with audio feature extraction in Python.

Specific Objectives:

• To implement the use of MFCC Features and the extraction of features from the music file for the classification of the different kinds of genre gathered from the sound.

• To research and experiment the use of different libraries/framework for the application used in audio signal processing in Python.

• To learn and explain how audio feature extraction can be used in the identification of the genre in a music file.

1.3 Scope and Limitations

The study covers the identification of genre in music using the basics of audio feature extraction and modules used for audio signal processing in Python. A system which uses only Mel Frequency Cepstral Coefficients (MFCCs) as our only feature used for representing each music genre, and by using the Logistic Regression as our classifier for identifying these genres.

The study seeks to classify the music genre of songs. The structure of a song will become more complicated as time goes by, thus giving it a sub-genre. This study focuses only in classifying the main genre on which the song belongs. And the genre that will be tested in this study will only be limited to the four basic genres, namely, Classical, Jazz, Pop, and Rock.

2 REVIEW OF RELATED LITERATURE AND TECHNOLOGY

2.1 Music Genre Classification

There are many previous researches about the field of audio signal processing. Music genre classification in audio signals revolves around the extraction of features of a music file. There have been manual classifications of music done in the past and therefore techniques for automatic genre classification would be valuable and would be a good addition in the development of audio information retrieval for music (Tzanetakis, G., & Cook, P. 2002). It is obvious that genre classification in music aims to determine the genre by studying the unique features of audio signals.

Many music files are accessible from different source like the Web, which needs a method for an organized search from these databases. Since music is a multifaceted, multi-dimensional medium, it requires unique abstraction, representation, and processing techniques which are different from other retrieval tasks (Talupur, M., Nath, S., & Yan, H. 2001).

2.2 Feature Extraction

2.2.1. *Timbre Texture Features*

Table 1. Timbral texture features

Feature	Description
MFCC	Representation of the spectral characteristics based on Mel-frequency scaling [12]
Spectral centroid	The centroid of amplitude spectrum
Spectral rolloff	The frequency bin below which 85% of the spectral distribution is concentrated.
Spectral flux	The squared difference of successive amplitude spectrum.
Zero crossings	The number of time domain zero crossings of the music signal.
Low-energy	The percentage of analysis windows that have energy less than the average energy across the texture window.

Figure 2.2.1.: Chang, K. K., Jang, J. S. R., & Iliopoulos, C. S. (2010, August)

2.2.2. *Musical Surface Features*

From Tzanetakis, G & Cook, P (2002) Musical *genre classification of audio signals, "*musical surface*"* is used to denote the characteristics of music related to texture, timbre and instrumentation. 9-dimensional feature vector is used in their system. The following are: mean-Centroid, mean-Rolloff, mean-Flux, mean-ZeroCrossings, std-Centroid, std-Rolloff, std-Flux, std-ZeroCrossings, & LowEnergy. These means and standard features are calculated in a "texture" and an "analysis" window which is based on the Short Time Fourier Transform (STFT) that is calculated using the Fast Fourier Transform (FFT) algorithm (Tzanetakis, G., & Cook, P. 2002).

2.2.3 *Mel-frequency Cepstral Coefficients (MFCC)*

In most projects that are related to audio signal processing never fails to mention the Mel-frequency Cepstral Coefficients (MFCC) features. These features are mostly used for speech, musical, and classical genres in audio signals.

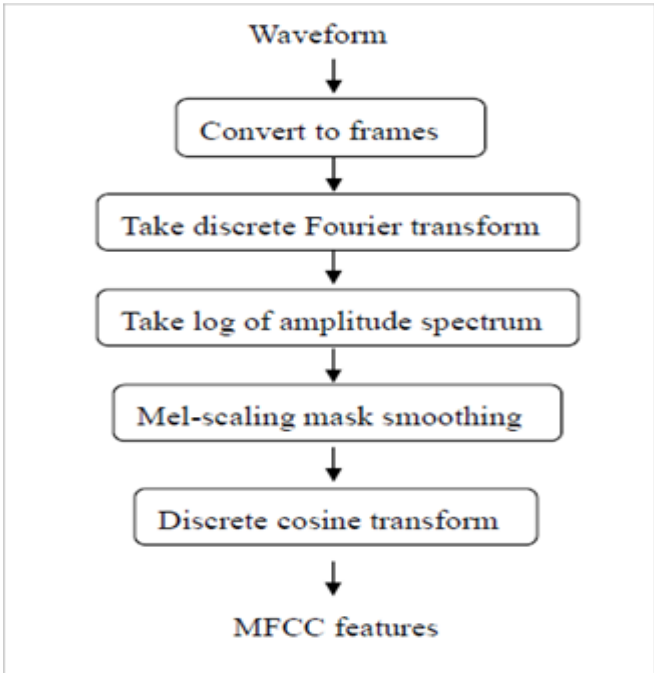

Figure 2.2.2: The flow chart of the calculation of MFCCs.
Chu, M.(2002) Automatic Music Genre Classification.

c. *Rhythm Features*

Rhythm Features describes the periodicity of an audio signal. The calculation of these features is based on the Wavelet Transform (WT). It provides high time and low frequency resolution for all frequencies (Tzanetakis, G., & Cook, P. 2002).

2.3 Classification Methods

2.3.1 Support Vector Machines

Support Vector Machines (SVM) is a type of machine learning systems that analyze data and recognize data. It is a popular approach used for the classification, data mining, pattern recognition, and regression analysis in many research areas.

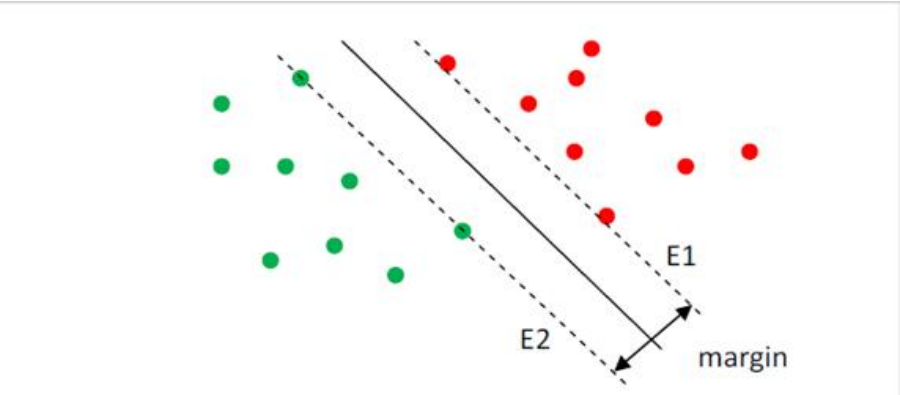

Figure 2.3.1: SVM classification hyperplane to maximize the margin between two support hyper-planes**.**
Chu, M. (2002) Automatic Music Genre Classification

2.3.2 Artificial Neural Networks

ANNs are computational models that are capable of machine learning and pattern recognition. In Talupur, M., Nath, S., & Yan, H. (2001), they used a structure of the neural network. [1] Four output units, one for each genre. [2] One hidden layer, having 30 hidden units. [3] Input layer, having 128 units. From their experiment they found out that the learning phase becomes low if the number of units in the hidden layer is increased above 30.

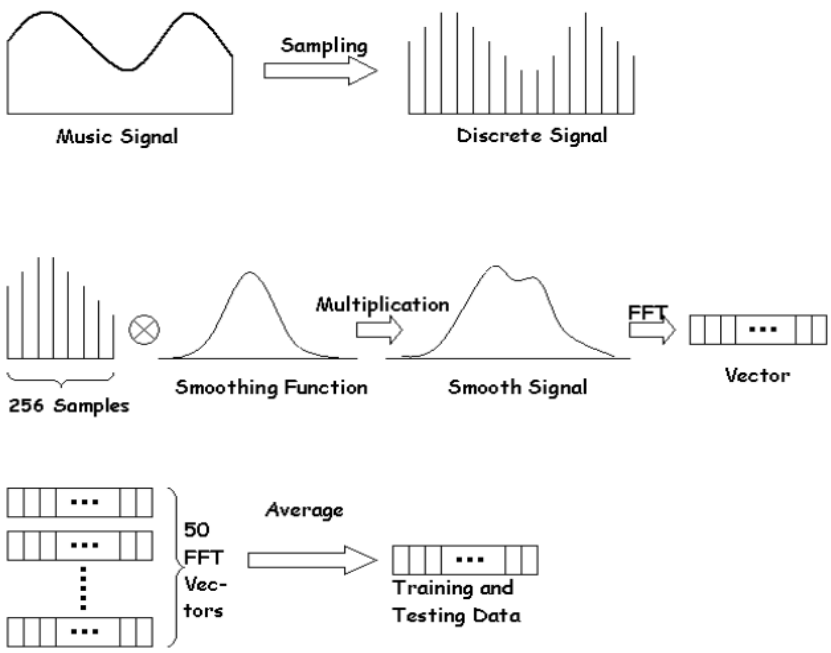

Figure 2.3.2: Pre-processing of audio clip to get input data
Talupur, M., Nath, S., & Yan, H. (2001). Classification of music genre.

2.3.3 *Naïve Bayesian*

In Iqbal, M., Khandelwal, A., Singh, K. (2012) *Automatic Music Classification,* they implemented the Naïve Bayesian as the classifier used in MATLAB and implemented on Android. The classifier assumes that the distribution of features in each genre is normal with some mean and variance which follows the Central Limit Theorem. The Naïve Bayesian also assumes that the features used are independent of each, or simply oversimplification. The classifier showed satisfactory results but it was the worst among the three classifiers that was implemented.

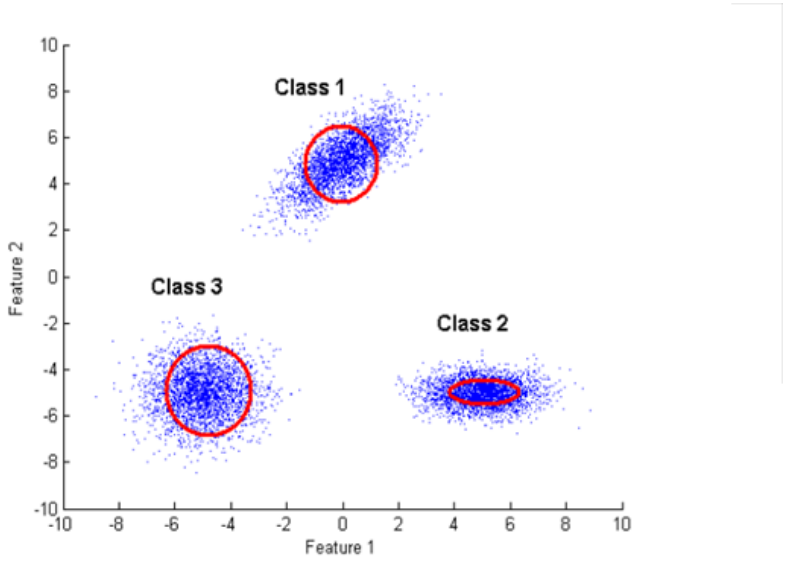

Figure 2.3.3: Distribution of classes determined by a naive Bayesian Classifier.
(Iqbal, M., Khandelwal, A., Singh, K. (2012) Automatic Music Classification)

2.3.4 *K-nearest Neighbour (KNN)*

The KNN is a non-parametric learning algorithm; this algorithm makes no assumption on the distribution of data which makes it useful. It also does not use training data points to do generalization, that it is very minimal and makes the training phase fast. The result of the newly inputted data is classified based on the majority of K-nearest neighbor category. The distance between positions is commonly measured by the equation Minkowski metric (Mei-Lan Chu (2002)). An example operation is shown below.

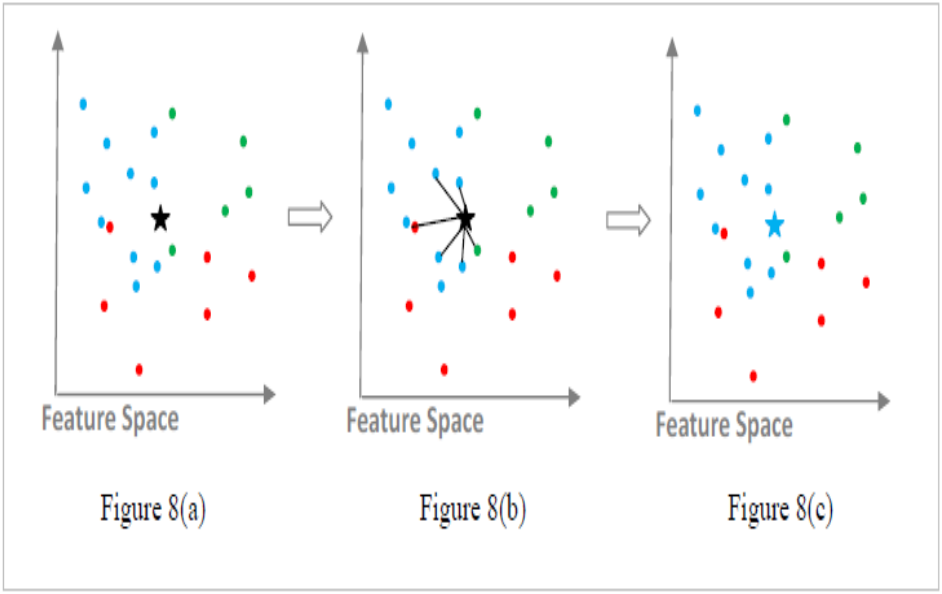

Figure 2.3.4: K-Nearest Neighbor.
Chu, M. (2002) Automatic Music Genre Classification.

2.3.5 *Compressive Sampling (CS)*

Compressive Sampling defined is a signal processing technique for efficiently acquiring and reconstructing a signal, by finding solutions to undetermined linear systems (Wikipedia 2014). This signal processing technique has recently been used as a research topic which attracts many researchers in different fields.

Chang, K. K., Jang, J. S. R., & Iliopoulos, C. S. used the Compressive Sampling (CS). They presented a CS-based classifier for music genre classification with two sets of features, including the short time and long-time features of audio music. Their proposed classifier generates a compact signature to achieve a significant reduction in the dimensionality of the audio music signals. The results showed that the time of computation of the CS-based classifier is only about 20% of SVM with an accuracy of 92.7% on the GTZAN dataset (Chang, K. K., Jang, J. S. R., & Iliopoulos, C. S. (2010, August)).

2.3.6 *Linear Vector Quantization (LVQ) Networks*

Learning Vector Quantization is a neural network based method to find a good set of reference vectors to be stored as nearest neighbors classifier's reference set. This method has high performance and it is often used for speech recognition. LVQ contains a Kohonen layer, a layer which learns and performs classification. It assigns equal number of Processing Elements (PE) for each class (Talupur, M., Nath, S., & Yan, H. 2001).

The basic LVQ trains and then uses Kohonen layer as follows:

1. In the training mode, the distance of a training vector to each PE is computed and the nearest PE is declared to be the winner.
2. If the winner PE is in the class of the training vector, it is moved toward the training vector.
3. If the wining PE is not in the class of the training vector, it is moved away from the training vector. This is referred to as repulsion.
4. During the training process, the PEs assigned to a class migrate to the region associated with their class.
5. In the classification mode, the distance of an input vector to each PE is computed and again the nearest PE is declared to be the winner. The input vector is then assigned to the class of that PE.

Figure 2.3.6.a: Talupur, M., Nath, S., & Yan, H. 2001,
Classification of Music Genre

2.3.7 *Evaluation of Feature Extractors and Psycho-acoustic Transformations for Music Genre Classification*

This section talks about the importance of psycho-acoustic transformations for effective audio feature calculation in identifying the genre of music. Musical genres are what the humans has created to characterize pieces of music, to be able to identify which musical style a single music belongs.

There are three (3) features that are being used in this study: the Rhythm Patterns feature and two (2) new representations that are being introduced: Statistical Spectrum Descriptors and Rhythm Histogram features. The two new features will have their performances be evaluated individually as well as in combination with the Rhythm Patterns feature. The following are the algorithm that they developed for extracting the Rhythmic Patterns:

Pre-processing 1 convert audio from au, wav or mp3 format to raw digital audio

Pre-processing 2 if audio contains multiple channels, average them to 1 channel

Pre-processing 3 take a 6 second excerpt from the audio, according to current processing position and considering lead-in, fade-out and step-width options

Step [S1] transform audio segment into spectrogram representation using Fast Fourier Transform (FFT) with hanning window function (23 ms windows) and 50 % overlap

Step [S2] apply Bark scale (Zwicker and Fastl, 1999) by grouping frequency bands into 24 critical bands

Step [S3] apply spreading function to for spectral masking effects (Schroder et al., 1979)

Step [S4] transform spectrum energy values on the critical bands into decibel scale [dB]

Step [S5] calculate loudness levels through incorporating equal-loudness contours [Phon]

Step [S6] compute specific loudness sensation per critical band [Sone]

Step [R1] apply Fast Fourier Transform (FFT) to the Sone representation. The result is a time-invariant representation of the 24 critical bands that captures reocurring patterns in the audio signal and thus is able to show rhythmic structure on each of the critical bands, i.e. amplitude modulation with respect to modulation frequencies. The transformation obtains amplitude modulation in the range from 0 to 43 Hz, however only the range from 0 through 10 Hz is considered in the Rhythm Patterns, as higher values are beyond what humans can perceive as rhythm.

Step [R2] weight modulation amplitudes according to fluctuation strength sensation. According to human hearing sensation amplitude modulations are perceived most intense at 4 Hz and decreasing towards 15 Hz.

Step [R3] apply a gradient filter to emphasize distinctive beats and perform Gaussian smoothing to increase similarity between two feature descriptors by diminishing un-noticeable variations. Post processing from all the Rhythm Patterns descriptors retrieved from the 6 second segments of a given piece of music, calculate the median as a descriptor for the whole piece of music

Figure 2.3.7: Lidy, T. & Rauber, A. 2001,
Evaluation of Feature Extractors and Psycho-acoustic Transformations for Music Genre Classification

Best results in Rhythm Patterns extraction were achieved with the GTZAN, ISMIRrhythm and ISMIRgenre audio collections in experiments O, C, and O respectively. Accuracy was 64.4, 82.8, and 75.0 %, respectively. The Statistical Spectrum Descriptor performed best when calculated after psycho-acoustic transformations, and taking the simple mean of the segments of a piece of audio. Accuracy was 72.7, 54.7, and 78.5 % in the GTZAN, ISMIRrhythm and ISMIRgenre data set, respectively, which exceeds the Rhythm Patterns feature set in 2 of the 3 collections. Rhythm Histogram Features achieved 44.1, 79.94, and 63.17 % accuracy, which rival the Rhythm Patterns features regarding the ISMIRrhythm data collection. Obviously a combination of feature sets offers itself for further improvement of classification performance. (Lidy & Rauber, 2005).

3 RESEARCH DESIGN AND METHODOLOGY

3.1 Conceptual Framework

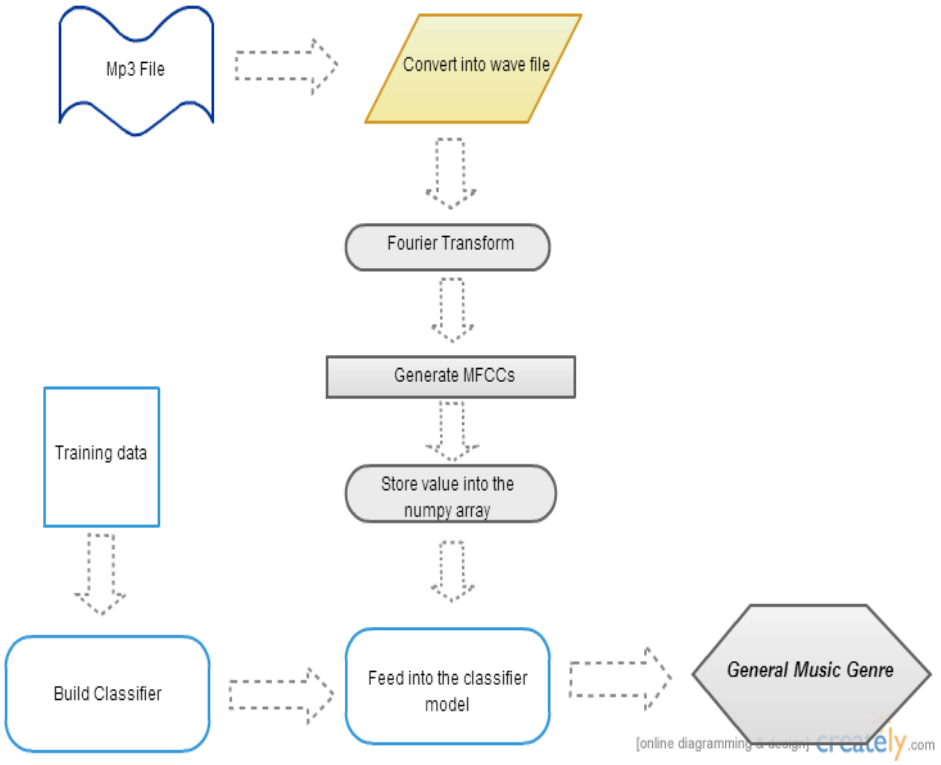

Figure 3.1: Framework for the study, *Music Genre Classification using Audio Feature Extraction*.

4 METHODOLOGY

4.1.1 Gathering and Manipulation of Music File

Music files are gathered from different sources mostly from the internet, format should be in .mp3 or .wav for the extraction of values. If the file is .mp3, the system should automatically convert it to wave file using a module in Python called Pydub. Using Pydub's Audio Segment function the wave file's channels will now be set into Mono and its sample rate into 22050.

4.1.2 Feature Extraction

The Fast Fourier Transform (FFT) is the algorithm used to compute and a way to understand an audio signal. It helps you see what the present components inside a complicated audio signal are by breaking it down into separate sine waves.

After FFT has been applied, calculation of Mel-frequency Cepstral Coefficient (MFCC) is then applied. These are features widely used to uniquely represent an audio file. These processes will be applied to the audio file after being converted to wav and being tweaked by the system for processing.

4.1.3 Conversion of dataset into a NumPy file

After the audio file has been processed, the data and features have been extracted and stored inside an '.npy' file, it will then undergo classification. For the classification the Logistic Regression will be used using different modules in Python. The genres that will be used in the classifier are Jazz, Classical, Pop and Rock.

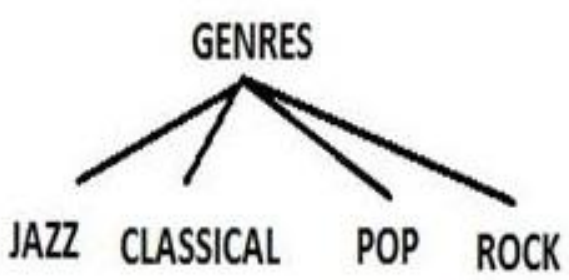

Figure 10: Genres to be classified for the study
Music Genre Classification using Audio Feature Extraction.

4.1.4 Testing the System

The system will be tested with set of audio files from different genres mentioned. The same audio file format will be used for the collection of songs that will be used for testing.

5 RESULTS AND DISCUSSIONS

5.1 Gathering of songs for each genre

For the study, the proponents have used at least 520 songs for the training with at least 130 songs per genre. We gathered the songs from different sources in the internet. We found datasets which is contains songs from each genre as our ground truth. The ISMIR 2004 dataset and the GTZAN dataset were used, also combined with our own songs from our playlists. Since the GTZAN dataset has a length of 30 seconds, and recorded at 22,050 Hz mono in WAV format, we converted the rest of the files from the ISMIR dataset and from our playlists which are .mp3 into wav with the same format as the GTZAN.

5.2 Feature Extraction

Mel-Frequency Cepstrum Coefficients

In the course of this study, the proponents decided to use MFCC as our only feature to be used in classifying music genres.

What are MFCCs? MFCCs are a way of representing the spectral information of a sound. These are widely used in speech recognition systems. MFCCs are derived from a type of cepstral representation of the audio clip. The difference between a normal cepstrum and a mel frequency cepstrum is that in the

MFC, the frequency bands are equally spaced on the mel scale which approximates the human auditory system's response more closely than the linearly –spaced frequency bands in a normal Cepstrum. With this it allows us to get a better representation of sound.

In the implementation that we used MFCCs are computed as follows:

- Pre-processing in time-domain (pre-emphasizing)
- Compute the spectrum amplitude by windowing with a Hamming window
- Filter the signal in the spectral domain with a triangular filter-bank, whose filters are approximately linearly spaced on the Mel scale, and have equal bandwidth in the Mel scale
- Compute the Discrete cosine transform of the log spectrum.

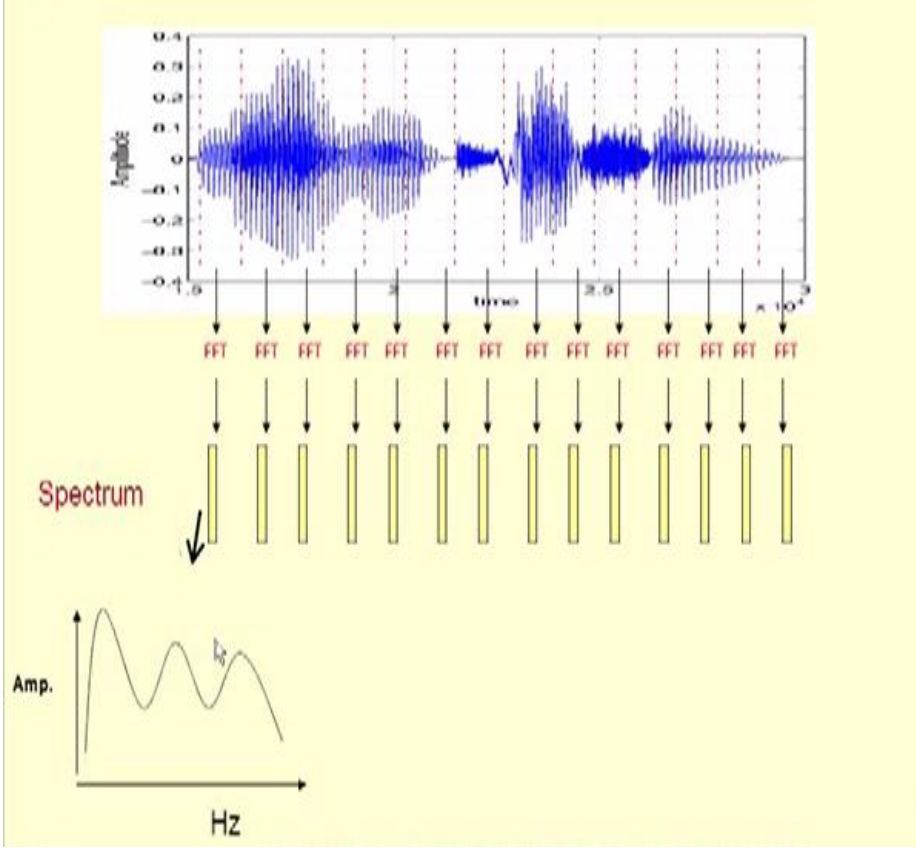

Figure 5.2.a: Visual representation of a signal represented as a sequence of spectral vectors [17]

In pre-processing the signal into time domain, first the signal is processed in short analysis windows, the signal is divided into successive overlapping frames with sizes 10-25 milliseconds. Each frame is applied by a window function. In each window it goes through FFT. Then the spectrum is derived after the FFT is being applied. Spectrum is the one that represents frequency components of a signal. The peaks indicated in the frequency range of the spectrum are referred to as formants, they denote dominant frequency components in the signal and these are what carry the identity of a sound.

A Cepstrum analysis is then applied to the spectrum. Basically what is done in this analysis is that it captures the spectral envelope of the signal. A Spectral envelope is the smooth curve connecting all formants (peaks).

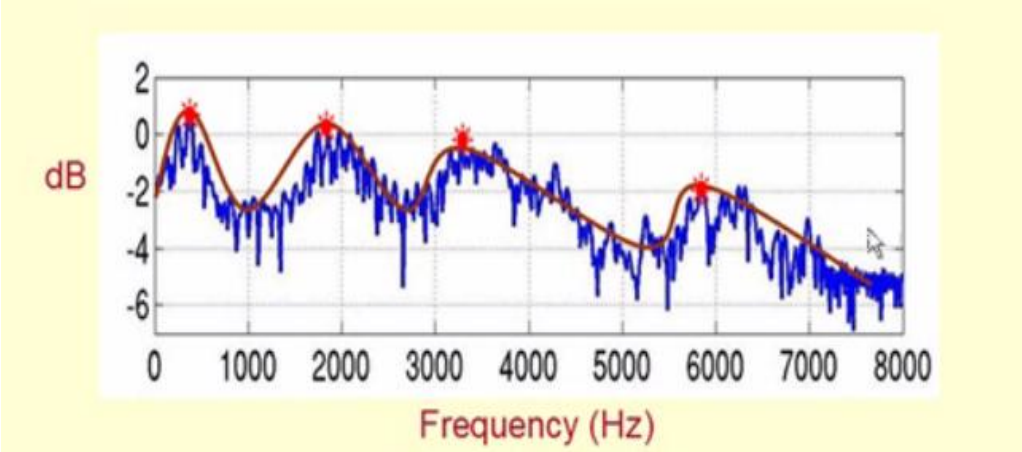

Figure 5.2.b: Spectral Analysis [17]

Next will be the Mel-frequency analysis, this analysis of a signal is based on the human ear perception experiments. It was observed that human ear acts as filter which means it only concentrates on only certain points of the frequency components. This is done so that the extraction of features will behave like the human ear, as we know the human ability to distinguish easily between genres even with minimal music exposure. The spectrum is passed through Mel-filters so that we obtain a Mel-spectrum.

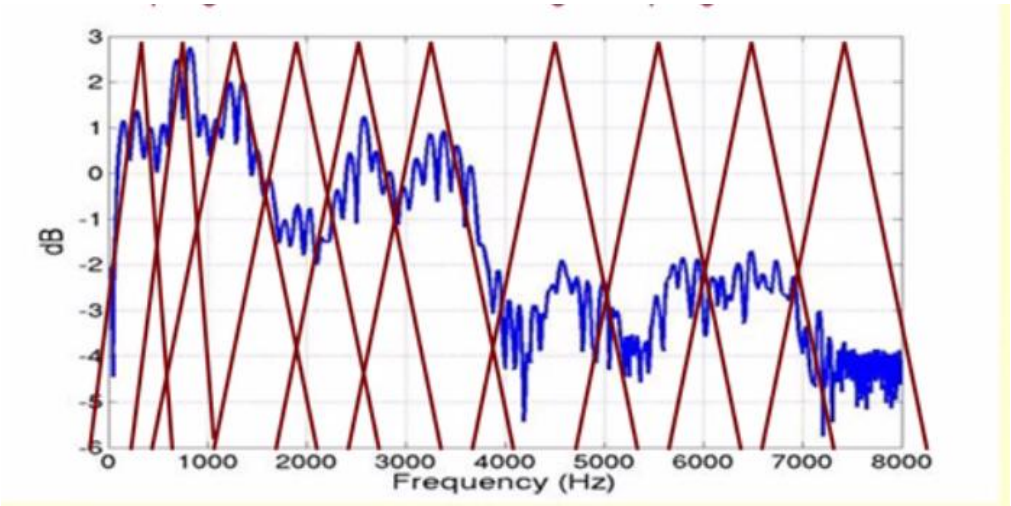

Figure 5.2.c: Mel-Filters [17]

Computation of the Mel spaced filter bank. This is a set of 20-40 triangular filters that is applied to the power spectral estimate. To calculate this filter bank energies, multiplication between each filter bank with the power spectrum then add the coefficients [16].

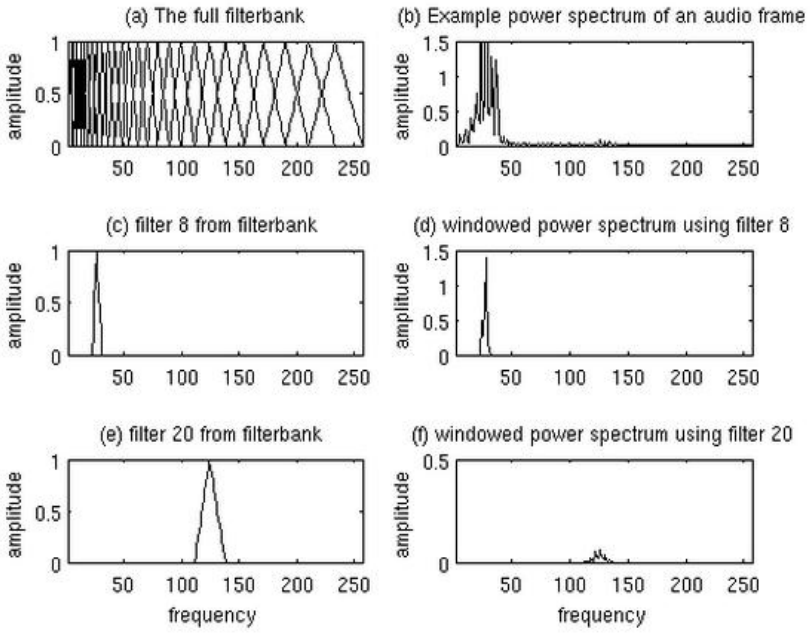

Figure 5.2.d: Mel-Filterbank[16].

Then take the log of these spectrums then convert it to Mel- scale. The Mel scale is a perceptual scale that is based on human hearing [18] which relates perceived frequency or pitch to its actual measure frequency. Incorporating this scale to the features will match it more closely to how human ears work as mentioned before since humans are better at observing changes in pitches. Then compute the DCT of the log Mel spectrum which reduces the data which leaves us a series of uncorrelated values. This process of computing DCT is done in order to convert the Mel spectrum back into the spatial domain. The results after applying DCT are the MFCCs.

All of these methods are implemented in Python using the mfcc function from the Talkbox toolbox. Talkbox is a set of python modules for speech/signal processing. This toolbox contains features such as Fourier-like transforms (DCT, DST, MDCT, etc) speech related functionalities: MFCC, mel spectrum etc... For every audio file in the dataset used for training undergoes a process where we generate each of the song's MFCCs. This approach has been successful in a lot of works such as speech and speaker recognition, including music. Below is the snippet of code that is used in generating MFCCs:

```
wav_file = 'C:\Users\\vjkcalag\\Desktop\\001. Motorhead - Ace of Spades.wav'
base_fn, FileExtension = os.path.splitext(wav_file)
data = base_fn + '.ceps' #cache so that it will be easy to read.

sample_rate, X = scipy.io.wavfile.read(wav_file)
ceps, mspec, spec = mfcc(X) #generate mfcc of audio file
numpy.save(data, ceps) #saves data into a numpy array(.npy)

print ceps
```

Figure 5.2.a Generating MFCCs of an audio file

```
[[ -8.84569914e+00  -2.40696607e+00  -4.98615221e-01 ...,  -5.41284991e-02
    3.30658776e-03   2.70186348e-03]
 [ -9.44600520e+00  -2.10198434e+00  -2.80560701e-01 ...,  -2.48919407e-02
   -3.24296748e-02  -1.11089783e-03]
 [ -8.56670824e+00  -2.64537144e+00  -3.27542403e-01 ...,  -7.97730138e-02
    8.37027659e-02  -2.09590642e-01]
 ...,
 [ -1.03240695e+01  -2.36649617e+00  -3.34818829e-01 ...,  -3.00756583e-02
   -1.47848679e-01  -3.29622188e-01]
 [ -1.17522119e+01  -2.24179548e+00  -2.91235845e-01 ...,  -6.96041323e-03
   -1.60220066e-02  -1.35290255e-02]
 [ -1.21202285e+01  -2.25189770e+00  -3.12537700e-01 ...,  -4.58661901e-02
   -4.55825077e-02  -3.03965278e-02]]
```

Figure 5.2.b Coefficients taken from an audio file.

As what we mentioned above a set of coefficients is extracted for each frame of the sound, taking all these values would be a lot for the classifier. The proponents followed the standard number of taking only 13 coefficients from each song which is used in the implementation. To do this we did is that we did an averaging per coefficient in all of the frames, an approach used in the book by Richert[19], also assuming that the start and end of each song is less genre specific and ignoring the first and last 10 percent.

```
wav_file = 'C:\Users\\vjkcalag\\Desktop\\001. Motorhead - Ace of Spades.wav'
base_fn, FileExtension = os.path.splitext(wav_file)
data = base_fn + '.ceps'

sample_rate, X = scipy.io.wavfile.read(wav_file)
ceps, mspec, spec = mfcc(X) #generate mfcc of audio file
numpy.save(data, ceps) #saves data into a numpy array(.npy)
#print ceps.shape
#print ceps

test = []

data = data +'.npy'
ceps = numpy.load(data)
num_ceps = len(ceps)

test.append(numpy.mean(ceps[int(num_ceps / 10):int(num_ceps * 9 / 10)], axis=0))

print test
```

Figure 5.2.c: Snippet used for averaging coefficients for each song.

```
[array([  2.05415970e+01,  -7.72621983e-01,  -6.80611792e-01,
          1.88447587e-01,   8.16150797e-02,  -1.26650635e-01,
         -1.75657819e-01,  -2.47901973e-01,  -1.00487181e-01,
         -2.62842502e-02,  -9.86102529e-02,   6.65964571e-02,
          1.23545443e-02])]
```

5.3 Automatic conversion of mp3 files.

For every mp3 file that is being fed, it will automatically be converted into wav file then set the following settings (channels, sample rate) and then feed it into the classifier. Using the *AudioSegment* function from the module *Pydub* we constructed the following code.

```
if FileExtension == ".mp3":

    sound = AudioSegment.from_mp3(sl)
    sound.export(converted_dir + fname + ".wav", format = "wav")

elif FileExtension == ".wav":

    sound = AudioSegment.from_wav(sl)
    sound.export(converted_dir + fname + ".wav", format = "wav")


sound = AudioSegment.from_wav(new_file)
sound = sound.set_channels(1)
sound = sound.set_frame_rate(22050)
sound.export(new_file, format = 'wav')
```

Figure 5.3.a. Conversion of mp3 files into wav.

5.4 Building the Classifier

Logistic Regression

In building the classifier, a function called *LogisticRegression()* which can be found from the python module *scikit-learn* will be used. The data inside the .npy file that has been generated through the application of *mfcc() function* in each file will then be extracted and used as a training set for the Logistic Regression.

Logistic Regression is a type of regression that predicts the probability of occurrence of an event by fitting the data to a logistic function. This regression method makes use of several predictor variables that may be numerical or categorical [15].

```
clf = LogisticRegression()
clf.fit(X_train, y_train)
clfs.append(clf)
```

Figure 5.3.a: Command used to create and train the classifier

The classifier will now generate a pattern for each genre out of the set of data that was created through Logistic Regression. This pattern of data will be used later in the testing part for comparison in determining the true genre of the song.

For further efficient use, the classifier generated will then be saved as a model to permanently be serialized to the disk.

```
joblib.dump(clf, 'saved_model/model_mfcc.pkl')
```

Figure 5.3.b: Command used to save the model into disk

After the process of Logistic Regression, a confusion matrix will then be created showing the accuracy of the classifier. It allows the visualization of the performance of the used algorithm.

As mentioned in *section 5.1,* at least 520 songs have been used for training and at least 130 songs per genre. In building the classifier, at least 90% of the songs in each genre is used to generate the projected accuracy of the classifier. Shown below is the table of data of the confusion matrix:

	Jazz	Classical	Pop	Rock
Jazz	56	16	11	35
Classical	15	88	0	12
Pop	4	0	75	18
Rock	9	1	23	105

Table 5.4.a. Data of the Confusion Matrix made by the Logistic Regression.

In building the tester, each audio file that will be tested must undergo the same process as the training dataset. Thus, converting it into an .npy file; as this is the only format that the built classifier could understand. The .npy file will now have its data extracted and be fed in the model for comparison. After the comparison was done, each genre will return a value containing the probability. The genre with the greatest value will be determined as the true genre of the song.

5.5 Testing the System

The GUI is created using the module *Tkinter* in Python. An audio file in .wav or .mp3 format will be imported and signals will then be plotted through the help of the module *matplotlib*. The interface contains an import menu that will open a file browser interface on which the user could pick a song. During this process the audio file is automatically converted into a wave file, mono channel, and a sample rate of 22,050Hz. There is a play button below the signal; if we press that button it will play a part of that selected song which is 15 seconds, while the system is playing the song the system is getting the values and generating the MFCCs. After the 15 seconds the predicted genre pops up inside another Tkinter window.

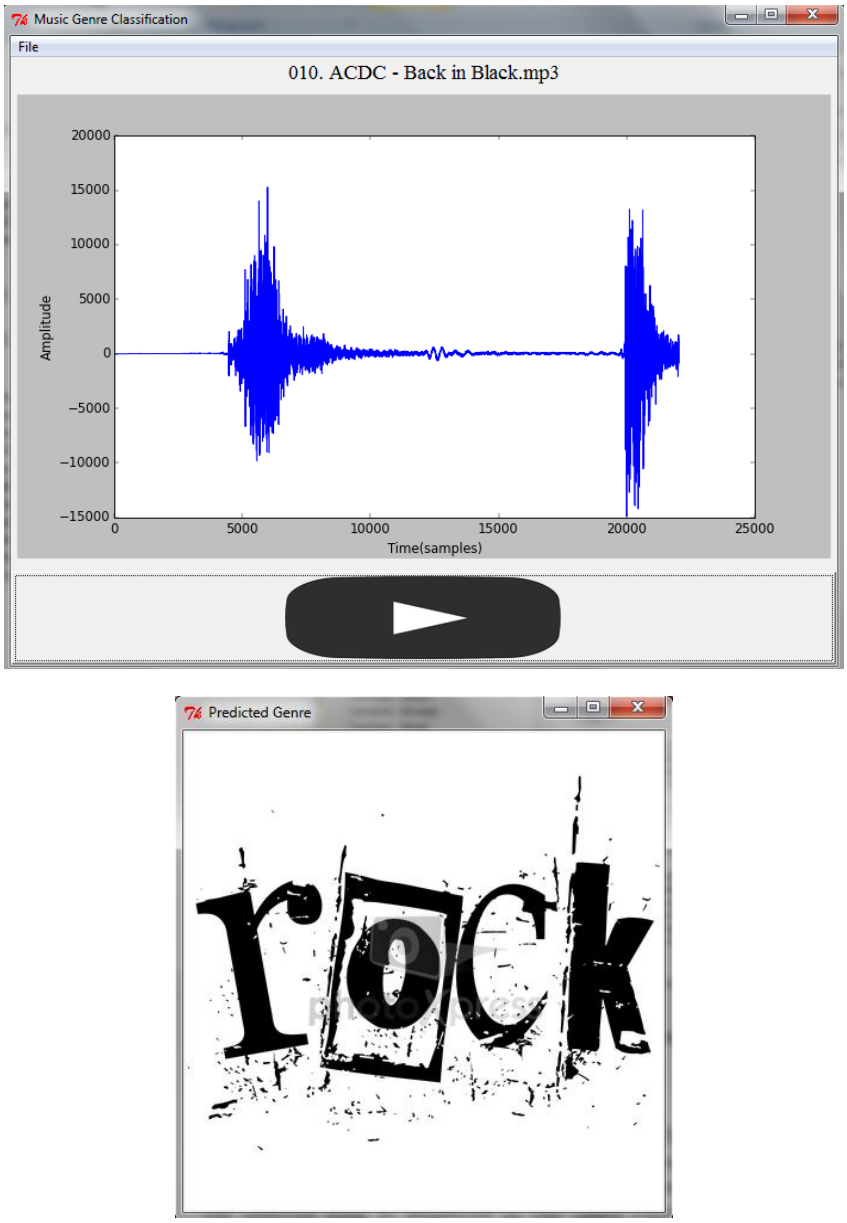

Figure 5.4: Screenshots of the Music Genre Classifier

The proponents used the following input format so that it will be uniformed since most of the songs in our training set, especially the GTZAN dataset has these default encoding (22 KHz sample rate, 352Kbps bit rate). We set our conversion in Audacity for the dataset into wav 16 bit mono at 22 KHz to get a uniform bit rate of 352 Kbps and we did the same for our automatic converter with .mp3 files. Testing songs with encoded in higher sample rate gives us errors from the Talkbox toolbox implementation due to memory and other constraints.

5.6 Test results

In the testing phase of the model we used various songs for testing which are downloaded from the internet. We used songs from: *The Best Rock Songs of All time, Billboard 100 Singles Chart, Elevator Music Band –Elevator Music,* and tracks from the ISMIR dataset. Below are the results of the testing that has been done:

	Jazz	Classical	Pop	Rock
Jazz	36	6	13	5
Classical	11	73	4	2
Pop	10	2	77	11
Rock	2	0	13	85

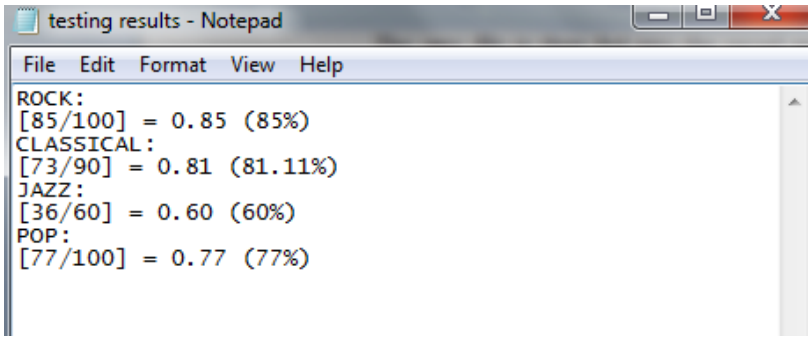

Figure 5.6: Results from testing.

The results are fair with classifying 4 genres. The rock and classical genres seems to be the easiest to classify, while the jazz produce very poor results. Classifying pop songs sometimes gets confused with rock but the values in the probability between the two is close enough. It also depends on the song, for instance pop rock songs get close results. This shows that the approach that is used also has it downfall in some areas.

6 CONCLUSION AND RECOMMENDATION

6.1 Conclusion

The proponents were able to achieve an output for the overall objective of the study which is to be able to implement a desktop application which is a music genre classifier using Audio feature extraction in python with the use of Mel-Frequency Ceptrum Coefficients as our only feature with the Logistic Regression. The approach was achieved using different implementations and different modules related to the field of audio signal processing. With the use of only one feature as our comparison between genres, it gave us fair results in the classification. It may not be a 100% accurate but it was able to distinguish these features within these categories.

The proponents found out that the classification for every song will differ depending on what classifier and what genre. Since music has been changing its structure over time, the music genre becomes a mix of different beats or rhythm or becomes a combination of a certain genre, thus making its structure more difficult than ever and posing a greater challenge in making an effective classifier. The results of the classification still shows that it is fair enough and can be used as a start to build a better classifier. Even if there are songs falling into wrong genre, it still shows decent probability on which genre the song truly belongs.

6.2 Recommendation

By using modules in python to construct the processing of an audio signal into a GUI, the system can now choose any file from the directory to classify. It allows the user to choose formats between .wav or .mp3 files. If file is .mp3 the system will convert it into a wave file for processing purposes and by setting it to a specific encoding. The proponents used only a few genres to classify, adding more genres would greatly improve the flexibility and the scope of the program.

The proponents only used the MFCC features to get the data from an audio file. Diving deeper into the audio to extract things, for instance, drum pattern and similar genre-specific characteristics would greatly improve the accuracy of the classifier. Also, since the proponents mentioned in conclusion that the structure of a song changes over time. Having a more up to date training set will greatly help the classifier in understanding more the structure of the modern songs.

7 **REFERENCES**

[1] Tzanetakis, G., & Cook, P. (2002). Automatic Musical genre classification of audio signals. *Speech and Audio Processing, IEEE transactions on*, *10*(5), 293-302. Retrieved from http://dspace.library.uvic.ca:8080/bitstream/handle/1828/1344/tsap02gtzan.pdf?sequence=1

[2] Talupur, M., Nath, S., & Yan, H. (2001). Classification of music genre. *Project Report for*, *15781*. Retrieved from http://www.cs.cmu.edu/~yh/files/GCfA.pdf.

[3] Burred, J. J., & Lerch, A. (2003, September). A hierarchical approach to automatic musical genre classification. In *Proc. 6th Int. Conf. on Digital Audio Effects' 03*. Retrieved from http://users.cis.fiu.edu/~taoli/pub/factors-music.pdf

[4] Chang, K. K., Jang, J. S. R., & Iliopoulos, C. S. (2010, August). Music Genre Classification via Compressive Sampling. In *ISMIR* (pp. 387-392). Retrieved from http://ismir2010.ismir.net/proceedings/ismir2010-66.pdf

[5] Chu, M.(2002) Automatic Music Genre Classification. Retrieved from http://djj.ee.ntu.edu.tw/music_genreclass.pdf

[6] Li, T., Ogihara, M., & Li, Q. (2003, July). A comparative study on content-based music genre classification. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (pp. 282-289). ACM.

[7] Seyerlehner, K., Schedl, M., Pohle, T., & Knees, P. (2010). Using block-level features for genre classification, tag classification and music similarity estimation. *6th Annual Music Information Retrieval Evaluation eXchange (MIREX-10), Utrecht, Netherlands*.

[8] Panagakis, Y., Kotropoulos, C., & Arce, G. R. (2010). Non-negative multilinear principal component analysis of auditory temporal modulations for music genre classification. *Audio, Speech, and Language Processing, IEEE Transactions on*, *18*(3), 576-588. Retrieved from http://poseidon.csd.auth.gr/papers/PUBLISHED/JOURNAL/pdf/Panagakis_TALP2010.pdf

[9] Doudpota, Sher Muhammad, Sumanta Guha, and Junaid Baber. "Mining movies for song sequences with video based music genre identification system."*Information Processing & Management* 49.2 (2013): 529-544.

[10] Tsunoo, Emiru, et al. "Audio genre classification using percussive pattern clustering combined with timbral features." *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*. IEEE, 2009.

[11] Priit Kriss. "Audio Based Genre Classsification of Electronic Music

[12]http://scikit-learn.org/stable/modules/svm.html#multi-class-classification

[13] John GLOVER, Victor LAZZARINI and Joseph TIMONEY. "Python For Audio Signal Processing. "

[14]http://aimotion.blogspot.com/2011/11/machine-learning-with-python-logistic.html

[15] http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

[16] Speech Technology - Kishore Prahallad. "Spectogram, Cepstrum, Mel-Frequency Cepstral Coefficients" retrieved from https://archive.org/details/SpectrogramCepstrumAndMel-frequency_636522

[17] Róisín Loughran, Jacqueline Walker, Michael O'Neill and Marion O'Farrell. ''The Use of Mel-Frequency Cepstral Coefficients in Musical Instrument Identification.'' Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.331.2898&rep=rep1&type=pdf

[18] Richert, Coelho. "Building Machine Learning Systems with Python"

[19] J.Urbano, D.Bogdanov, P.Herrera, E.Gomez and X.Serra "WHAT IS THE EFFECT OF AUDIO QUALITY ON THE ROBUSTNESS OF MFCCs AND CHROMA FEATURES?" .Retrieved from http://www.terasoft.com.tw/conf/ismir2014/proceedings/T103_326_Paper.pdf

[20] S.Gupta, J.Jaafar, W.F.Ahmad, A.Bansal “Feature Extraction using MFCC” Retrieved from http://airccse.org/journal/sipij/papers/4413sipij08.pdf