Machine learning Approaches for Music Genre Classification

Thesis Submitted in Partial Fulfillment of Requirements for the Award of the Degree of

Master of Technology in

COMPUTER SCIENCE & ENGINEERING

SUBMITTED BY

SHASHANK RATHORE

(2K15/CSE/17)

UNDER THE GUIDANCE

OF

MANOJ KUMAR

ASSOCIATE PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

BAWANA ROAD, DELHI-110042

(2015-2017)

CERTIFICATE

This is to certify that report entitled **SHASHANK SINGH RATHORE (2K15/CSE/17)** has completed the project titled “**Machine learning Approaches for Music Genre Classification**” under my supervision in partial fulfillment of the MASTER OF TECHNOLOGY degree in Software Engineering at DELHI TECHNOLOGICAL UNIVERSITY.

Project Guide

Mr. Manoj Kumar

Associate Professor
Department of Computer Science and Engineering
Delhi Technological University
Delhi -110042

DECLARATION

We hereby declare that the Major project-I work entitled "**Machine learning Approaches for Music Genre Classification**" which is being submitted to Delhi Technological University, in partial fulfillment of requirements for the award of degree of Master Of Technology (Software Engineering) is a bonafide report of Minor Project-II carried out by me. The material contained in the report has not been submitted to any university or institution for the award of any degree.

Shashank Singh Rathore

2K17/CSE/17

ACKNOWLEDGEMENT

I am very thankful to **Mr. Manoj Kumar** (Associate Professor, Computer Science Eng. Dept.) and all the faculty members of the Computer Science Engineering Dept. of DTU. They all provided immense support and guidance for the completion of the project undertaken by me.

I would also like to express my gratitude to the university for providing the laboratories, infrastructure, testing facilities and environment which allowed me to work without any obstructions.

I would also like to appreciate the support provided by our lab assistants, seniors and peer group who aided me with all the knowledge they had regarding various topics.

SHASHANK SINGH RATHORE

M. Tech. in Software Engineering

IV - SEM

Roll No. 2K17/CSE/17

ABSTRACT

Humans have the ability to distinguish between two types of music very easily by listening to the songs for a short duration only. This decision cannot be only made only on the basis of basic music features, BPM and different pitches. This requires deeper understanding of the music. Machine Learning has been able to predict the different genres of music in a large collection of data. We are here trying to use this behavior by using various machine learning approaches and their accuracy in prediction. For feature extraction purpose of music we have taken MFCC (Mel Frequency Coefficients) into consideration which has given us improved performance. By using different machine learning algorithms we are able to understand music digitally on a new level that will help use it in other applications effectively for future purposes.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

CHAPTER 1

INTRODUCTION

1.1 Motivation of the Work

Music is being created from a long time which has been collected into huge amounts of data. There is a need to handle this huge amount of data. Music can be classified into different types based on the set of conventions already defined. The various genres of music are Blues, Rock, Classical, Country, etc. All these type of music have different rhythms, instruments, vocals and tempo that help us identify which type of music it is. Different users have different tastes in music or people may listen to only a particular type of music based on their moods for which may require an efficient way to classify music. The previous way was to include the genre of the music manually by adding this information in the metatag of the music files but this way requires a lot of time and effort. Here machine learning comes into picture and help us to classify music automatically with preprocessing a little information. Classifying music by using machine learning methods can also tell us a lot about the underlying patterns in the music types. Imagine being able to search for the songs which are similar to your favorite songs which you have been enjoying for a while, or to use some software agent of some kind that could recommend your friend songs based on your likes. This can help us in lot of other applications which are in need of such information which include Music Retrieval System, Music Recommender system for different customers. Many current Companies are working on this to develop such efficient systems which include Gaana, Saavan, Spotify, Soundcloud and some products like Shazam.

1.2 Significance of the Study

Mel-Frequency Cepstral Coefficients (MFCC) The Mel frequency scale was originally developed in phonetics to help model the nonlinear nature of human auditory system. To find the cepstral coefficients of a window in an audio signal, the discrete Fourier transform of the Hamming window of the signal is filtered by a bank of triangle filters equally distributed on the Mel frequency scale. The filter outputs are then fed to a logarithmic function and a cosine transform. 12 cepstral coefficients were used in my experiments. 25 ms windows are taken every 10 ms, resulting in 1000 feature vectors per 10 second audio sample.

CHAPTER 2

LITERATURE SURVEY

Music Genre Classification is a known issue and a very significant one and many people have previously attempted to solve this problem using various techniques. We will see how they were different from what we are trying to achieve and what all we learned from their ideas and solutions to this problem.

There are many ways to extract features from the music one can be based on the content of the songs, the other can be based on the lyrics of the songs. The content based feature extraction considers beat, pitch and tone of the music. Content based features are of three types rhythmic content based features, timbral texture based features, pitch content based features. Timbral textual features have been used previously on speech recognition processes. They are calculated for short span of time intervals based of SFTT (Short Time Fourier Transformation). We are here considering only the techniques which are based on Timbral Feature extraction.

One of the timbral feature extraction methods is Spectral Contrast (OSC) or Octave-Based Modulation Spectral Contrast (OMSC) have also been in used in [15]. Octave-based Spectral Contrast calculates the spectral valley, spectral peak and their difference in each sub-band. For most music, the strong spectral peaks roughly correspond with harmonic components; while non-harmonic components, or noises, often appear at spectral valleys. The difference between Octave-based Spectral Contrast uses octave-scale filter, while MFCC uses Mel-scale filters, Spectral Contrast extracts the spectral peaks, valleys, and their differences in each sub-band while MFCC sums the FFT amplitudes. At the last step, Spectral Contrast feature uses a K-L transform while MFCC uses a DCT transform.

Wei Chai and Barry Varcoe at MIT Media Laboratory have used hidden Markov models to classify four different symbolic representations of folk music from three Western European countries [11]. HMM has been previously been used for speech recognition and voice recognition that made it an obvious choice for music genre classification. It is seen that on increasing the number of states in the model did not have a signification effect on the accuracy of the model and the simple left-right and strict left-right were able to give comparatively better results than more complex HMM models. A continuous-input hidden Markov model layout was created for each genre with random weights for the edges. Each state's observation distribution was modeled by a single Gaussian with 12-dimensional mean and 12 by 12 diagonal variance for MFCC and LPC features and 36-dimensional mean and 36 by 36 diagonal variance for MFCC supplemented by delta and acceleration values. Hidden state number was varied between 3, 4 and 5 states. HMM Model are successful in classifying a small collection of music with high degree of variance but when provided with a large collection of music with less degree of variance between the classes it wasn't so successful.

Tao Lio of Computer Science Dept. University of Rochester tried to capture music features using DWCHs (Daubechies wavelet coefficients histogram).It is breaking down of music signal using wavelet coefficients to give us different sub bands representing different frequencies using different characteristics. It is successful in capturing the local as well as global information of the music. He used this data as an input to Support Vector Machine and k nearest neighbor and Gaussian Mixture models.

CHAPTER 3

FEATURE EXTRACTION

3.1 DATA COLLECTION:

The ideal songs for this project were to get a good mixture of songs having variety of artists and different styles within the genre. The data we have used here is taken from Marsayas (Music Analysis, Retrieval, and Synthesis for Audio Signals) software, this dataset has been used earlier also. Its website provides access to database, GTZAN genre collection. It is an open source dataset and is available for anyone to use doesn’t need any copyright information and have a collection of different genres. It has been commonly used for the academia and research purpose work related things. Files in this database were collected in 2000-2001 from different sources like Radio, microphone recordings and CDs to encompass variety in the way the music was collected. The dataset consisted of 1000 track each of 30 seconds long. The 1000 tracks were then categorized into 10 genres and each genre having 100 tracks. The 10 that we are working on and are included in the dataset are Blues, Classical, Country, Disco, Hip-hop, Jazz, Metal, Pop, Reggae, Rock. All the tracks are 22050Hz Mono 16 bit audio files. Since the files in the dataset are in the au format, which is lossy beacause of compression we needed it to convert into auv format (which lossless) before we proceed further. With lossless compression, every single bit of data that was originally in the file remains after the file is uncompressed. On the other hand, lossy compression reduces a file by permanently eliminating certain information, especially redundant information. When the file is uncompressed, only a part of the original information is still there (although the user may not notice it). We generally use data in lossy format nowadays to save space on the storage devices but for our purpose we will be converting our data from lossy format to lossless format. Now, the script ceps.py has to be run. This script analyzes and converts each file in the GTZAN dataset in a representation that can be used by the classifier and can be easily cached onto the disk. This little step prevents the classifier to convert the dataset each time the system is run. This script extracts the MFCC features from the data. We use this cached data to train the classifier.

3.2 MFCC:

MFC in the field of sound processing means power distribution of a sound in a short span of time based on linear cosine transformation of a log power spectrum on a non-linear mel scale of frequency. They were introduced by Davis and Mermelstein in the 1980s and are being used till now. Before this Linear Prediction Coefficients (LPCs) and Linear Prediction Cepstral Coefficients (LPCCs) were the main feature type for automatic speech recognition (ASR). The steps to calculate the MFCC are:

1. Frame the signal into short frames.
2. Take the fourier transformations of the signal.
3. Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.
4. Take the logs of the powers at each of the mel frequencies.
5. Take the discrete cosine transformation of the list of mel log powers, as if it were signal.
6. MFCC are the amplitudes of the resulting signal.

The reasoning behind performing these steps are as follows, first of all an audio signal is changing constantly so if we want to work on the data we need to find something that is not fluctuating so we have taken short time takes that are short enough to capture a signal information and during which the signal doesn't change much i.e. statistically it is stationary. Next we want to know the power spectrum of that each of those small frames. We are using Fourier transformation that finds the frequency of the musical signal corresponding to amplitude at that instant. This data obtained still has a lot of information which is not required the data which is inaudible to the human ears. To identify this information we are using Mel frequency filterbank. As the frequencies starts to increase our filterbank starts to change from narrow to wide as we are now less concerned about variations and more about the frequencies at each spot. Now we take the logarithm of these filterbank energies this is also based on the human hearing concept as we have seen to double the sound we are hearing we have to put 8 times as much energy into it. The logarithmic compression makes our features match more closely to what human ears hear. Then our final step is DCT this is used because our filterbank energies are quite correlated to decorrelate them we are using DCT it ensures that we are able to use diagonal covariance matrices that can be used to model the features for example in HMM classifier. Let us take an example to understand how FFT is working suppose we have to sine waves first.wav and second.wav which contain the sound of 400 Hz and 3000Hz sine waves. In the below figure 2.2 we have plotted the FFT for the two waves and we can see that there is a spike at 400Hz for the first sine wave and a spike at 3000Hz for the second sine wave. When we combine the two waves keeping the volume of the 400Hz sound wave half of the second sine wave. We can see in the FFT of the combined sine wave that the spike for the second wave is almost double to the first sine wave. For a real music the FFT can be seen in the below figure 2.1.

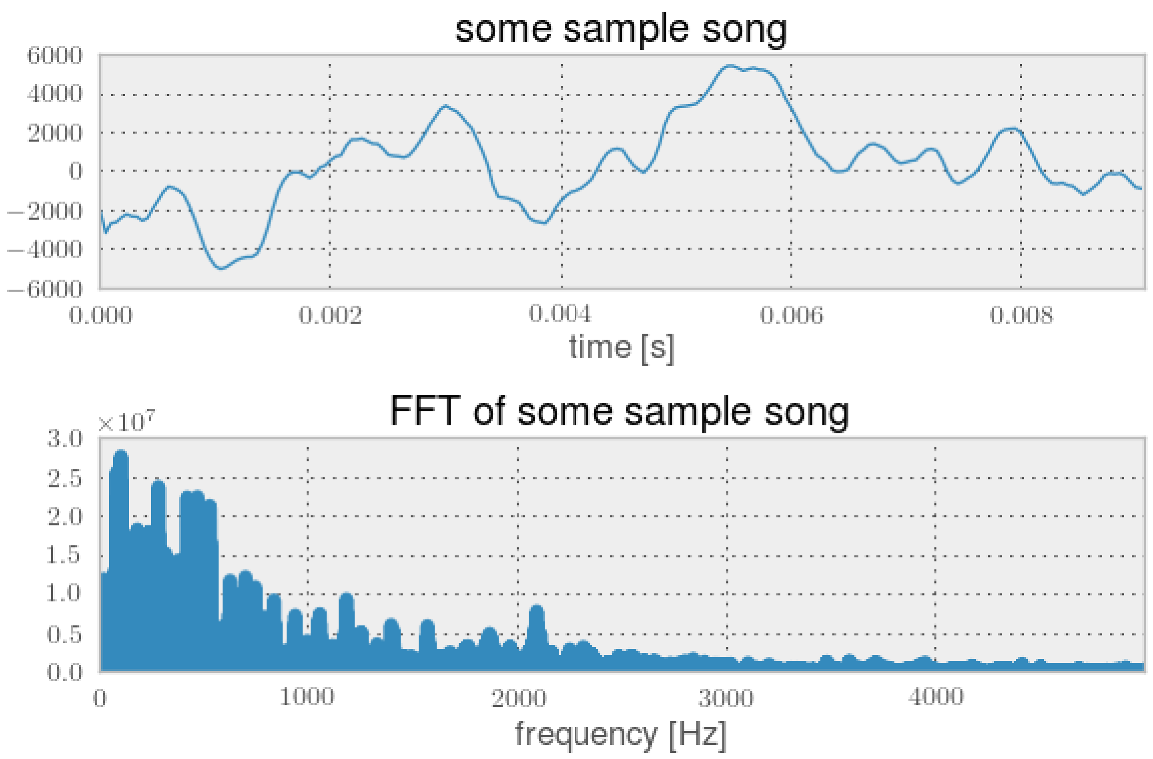

Figure 3.1

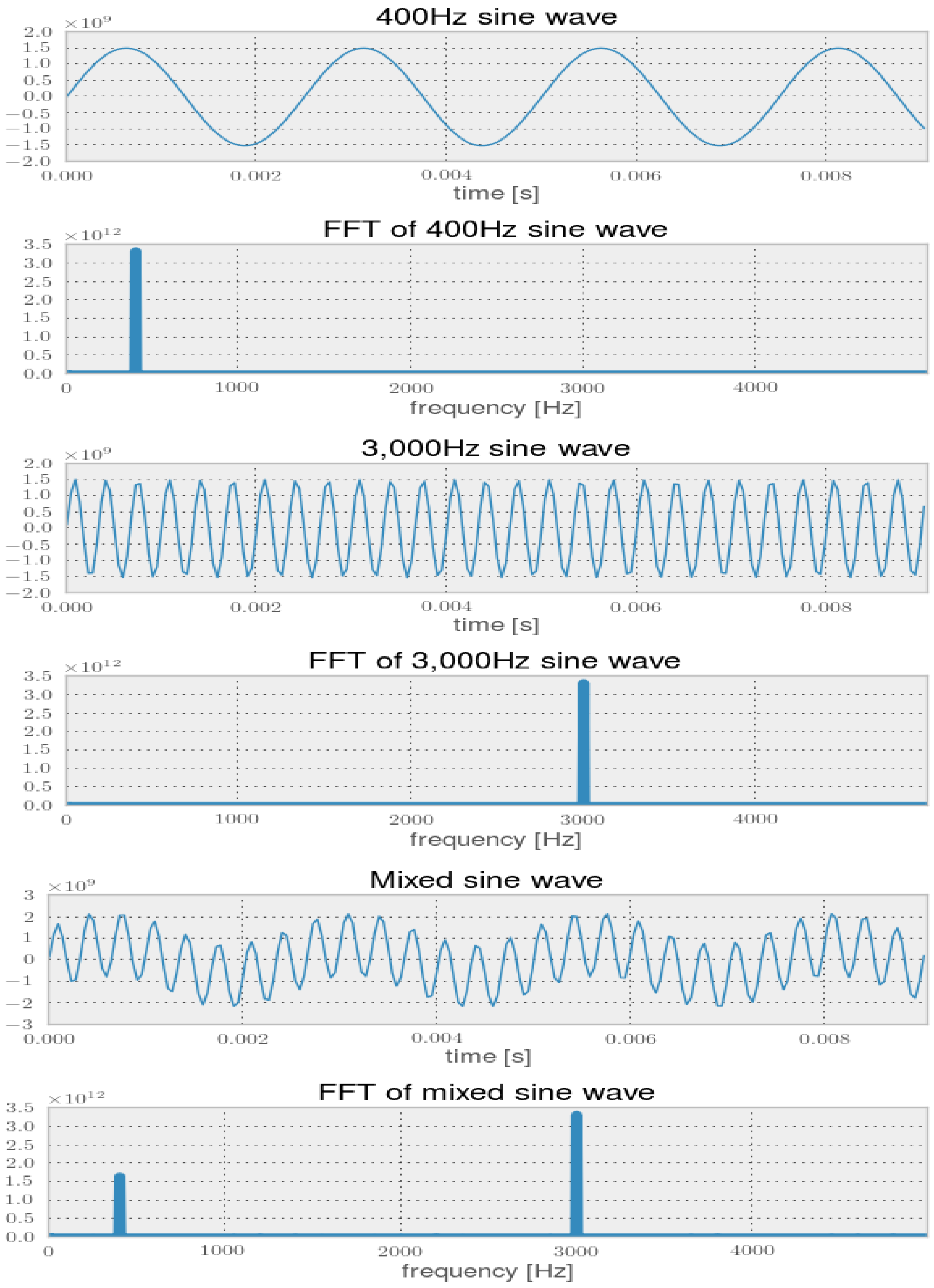
400Hz sine wave
×10⁹
2.0 1.5 1.0 0.5 0.0 −0.5 −1.0 −1.5 −2.0
0.000 0.002 0.004 0.006 0.008
time [s]
FFT of 400Hz sine wave
×10¹²
3.5 3.0 2.5 2.0 1.5 1.0 0.5 0.0
0 1000 2000 3000 4000
frequency [Hz]
3,000Hz sine wave
×10⁹
2.0 1.5 1.0 0.5 0.0 −0.5 −1.0 −1.5 −2.0
0.000 0.002 0.004 0.006 0.008
time [s]
FFT of 3,000Hz sine wave
×10¹²
3.5 3.0 2.5 2.0 1.5 1.0 0.5 0.0
0 1000 2000 3000 4000
frequency [Hz]
Mixed sine wave
×10⁹
3 2 1 0 −1 −2 −3
0.000 0.002 0.004 0.006 0.008
time [s]
FFT of mixed sine wave
×10¹²
3.5 3.0 2.5 2.0 1.5 1.0 0.5 0.0
0 1000 2000 3000 4000
frequency [Hz]

Fig 3.2

3.3 CONFUSION MATRIX:

As this is a multiclass problem we can solely rely on telling whether a particular song we have choosen is of a specific genre for example rock, classical, etc. Our algorithm we will giving an percentage of similarity with the available genres. There will be some genres which will be easily distinguished but some genres will be closely similar to each other and will be confused with each other time and again. Finding the similarity between the genres and how well our algorithms are able to distinguish between them can be done by Confusion matrix.

$$\begin{bmatrix} 26 & 1 & 2 & 0 & 0 & 2 \\ 4 & 7 & 5 & 0 & 5 & 3 \\ 1 & 2 & 14 & 2 & 8 & 3 \\ 5 & 4 & 7 & 3 & 7 & 5 \\ 0 & 0 & 10 & 2 & 10 & 12 \\ 1 & 0 & 4 & 0 & 13 & 12 \end{bmatrix}$$

It predicts the distribution set that the classifier has predicted for the test set for every genre. Here we have considered 6 genres therefore we have 6 by 6 matrix. The first row in the matrix tells us that for 31 classical songs 26 have been correctly predicted to classical songs, 1 has been predicted as Jazz song, 2 as country song and 2 to be metal. As we can see that for 31 classical songs 26 were correctly predicted but 5 were misclassifications. The second row gave us more confusing results out of the 24 jazz songs only 7 were correctly predicted rest all were mispridicted. As the data will be large and the genres will be more we want a much better way to visualize the data to get a better intuition into it. So will try to color map the matrix to get a better visualization of the data. In the below graph Fig. 2.3 we can see that our classifier is not providing 100% accuracy. For and ideal situation where the classifier is able to predict the correct output the squares in the diagonal from left-top corner to bottom-right corner will be dak in color and all the rest will of light color. In the graph we can see that only the row for classical genre is dark in color while all other are faded one so it is only predicting classical with accuracy and in the rest of the genres it is not able to give us good results. For displaying confusion matrix we are using

- From sklearn.metrics import confusion_matrix
- cm= confusion_matrix(y_test,y_pred)

- print(cm)

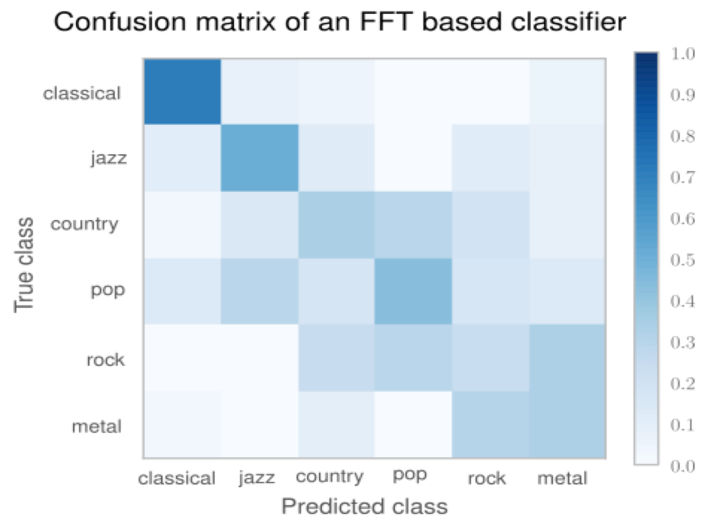

Fig 3.3

After that we have used matplotlib package available in python and pylab library in it to plot this graph based on the confusion matrix. The matshow() function of the pylab helps us in plotting this graph.

3.4 Receiver Operator Characteristic (ROC)

Using confusion matrix is not wholly reliable to measure the performance of the classifier. We need something more to get the proper idea of how the classifier is performing for this we looked into precision and recall curves to get a better understanding of the classifier. There is a cousin of the precision recall curves which is called as Receiver Operator Characteristic (ROC). ROC measures similar characteristics as precision and recall but provide different insight on the classification measure. The precision and recall method are better suitable when our positive class is much smaller than the present negative classes and we are more focused towards the positive class while the ROC provides the classification measure in general scenario. To understand the differences more clearly let us take the above scenario which we have taken while discussing Confusion matrix in classifying country songs. The graph below shows the ROC curve for classification of the country songs vs the rest of the genres.

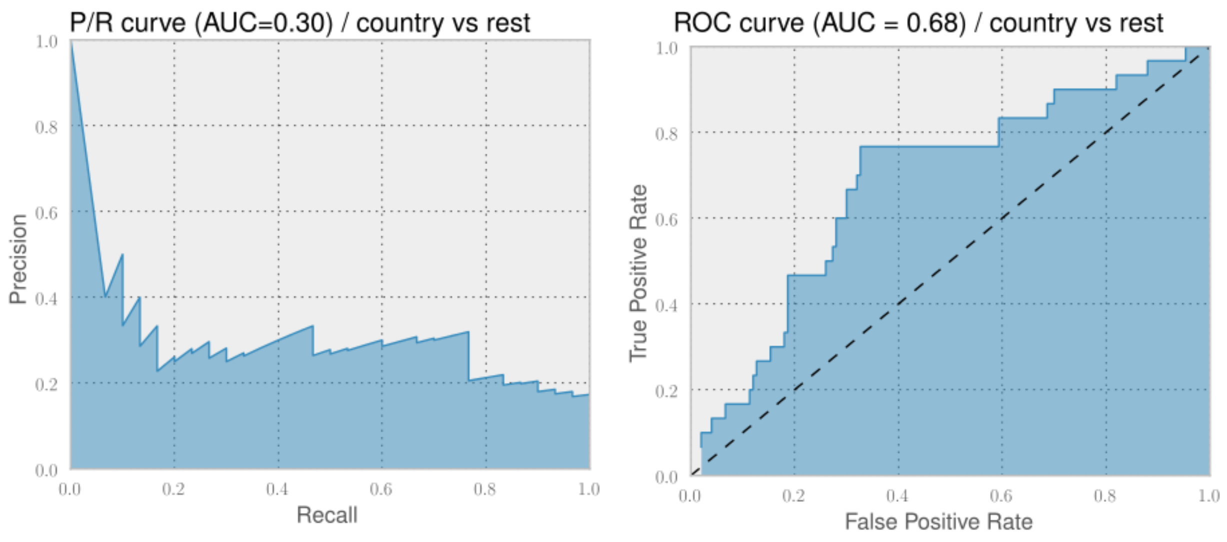

Fig 3.4

In the above figure the left graph is for P/R curve which in the ideal case should be going form left-top corner to the top-right corner for which the ideal value of AUC (Area under Curve) would have been AUC=1. The ROC curve is plotted against false positive rate and true positive rate. The right hand side of the figure shows the ROC curve for country vs rest which in the ideal case should have going from the bottom left corner to top-left corner and then to top-right corner whereas a random classifer will be the one where it is going form bottom left to the top-right as show in the graph with a dotted line this would have an AUC=0.5. When we are considering a measure of evaluating a classifier for the same data and then considering only one of them would be sufficient and the higher value of AUC for one will also mean a higher value for the other, therefore it is not appropriate to compare the two graphs with each other and we can take only of the two graphs into consideration when checking the correctness of the classifier and here we will be focusing only on the ROC curves. In ROC curve false positive rate measure truly negative results which were falsely identified as positive contrasting to which the P/R curve identifies positive examples which were truly identified as such. Moving forward we can also see that both P/R curve and ROC curve deals with binary classification problem and as our problem is a multi-classification one we will plot one vs rest curve for all the six genres which we have taken as example. As we will get six ROC curves as shown in Fig 2.5 we have seen that our example classifier works only for the classical genre will try further algorithms to improve this scenario.

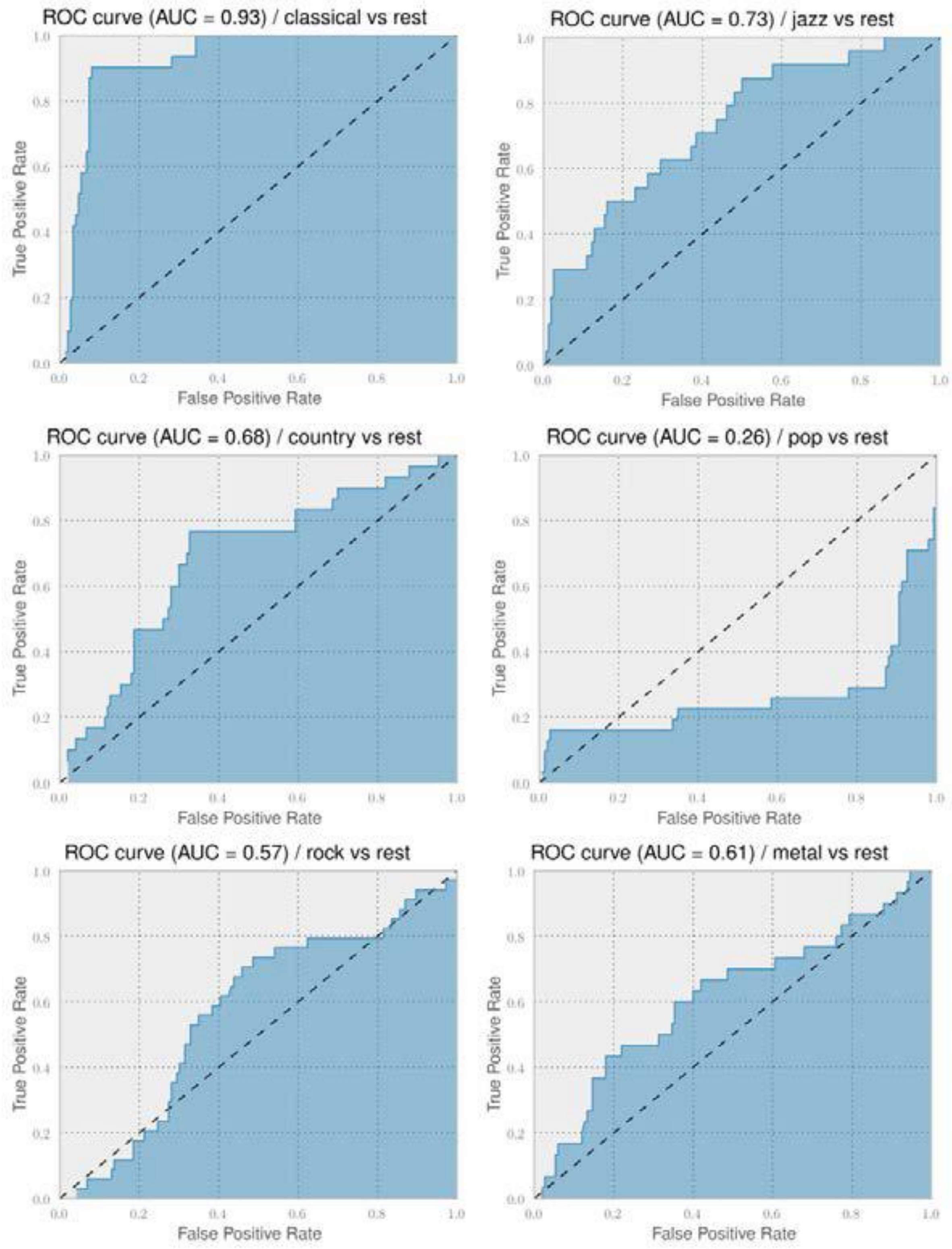
ROC curve (AUC = 0.93) / classical vs rest
ROC curve (AUC = 0.73) / jazz vs rest
ROC curve (AUC = 0.68) / country vs rest
ROC curve (AUC = 0.26) / pop vs rest
ROC curve (AUC = 0.57) / rock vs rest
ROC curve (AUC = 0.61) / metal vs rest
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

Fig 3.5

3.5 PCA (Principal Component Analysis):

In our project after calculation MFCC for the training data we had obtained a large matrix of around 3050 rows and 39 columns. As most of the machine learning algorithms which we have applied on the dataset usually take vectors as inputs so we have applied PCA to transform the matrix to a large vector. Now what the PCA does is it identifies the pattern among the data so that it could reduce the dimensions of the data and still able to represent the data well. PCA is generally used where the dimension of our data is 3 or more than by applying PCA and reducing the dimension of the data we are also able to visualize the data which makes it more meaningful. The PCA is most commonly performed on a covariance matrix i.e. the data should be in standard form and numeric in nature.

There is one more term which is closely heard with PCA is MDA (Multiple Discriminant Analysis) the difference between them is that in PCA we are trying to find the components which maximizes the variance in our dataset whereas in DMA in addition to finding the variance we are also considering the classes meaning we are trying to find the maximum variance within the classes when our dataset is having multiple classes.

We here have considered PCA only to reduce our dimensions. In PCA good subspace for us will be when we choose an value of k such that k<d and and the new k-dimensional data represent our dataset well. In PCA later we have to compute eigenvector form the dataset (or alternatively covariance matrix) to form scatter-matrix. Every eigenvector computed will have an eigenvalue which will give us information about the length and magnitude of eigenvector. Our dataset is in a good subspace it all the eigenvalues have similar values, but if some eigenvalues are much higher than other eigenvalues we might consider eigenvectors which are related to larger eigenvalues since they are large and may contain more information about the data. Since Eigen values which are close to zero may contain very less of information so we may consider dropping them.

So the steps in PCA approach can be summarized as follows:

- Take the complete dataset without considering the classes in it.
- For every dimension of the dataset calculate mean i.e d-dimensional mean vector.
- Computing the Covariance Matrix or Scatter-Matrix.

- After that calculate the eigen vector and their corresponding eigen values.
- Take k eigenvectors which have the largest corresponding eigen values and form a k * d dimensional matrix Z.
- Use this matrix Z to convert the samples into new subspace. This can be understood by the mathematical equation y= Z^T * x (where x represents one sample of the order d *1)

Below is an example of our dataset after applying the PCA to reduce it to three dimensions.

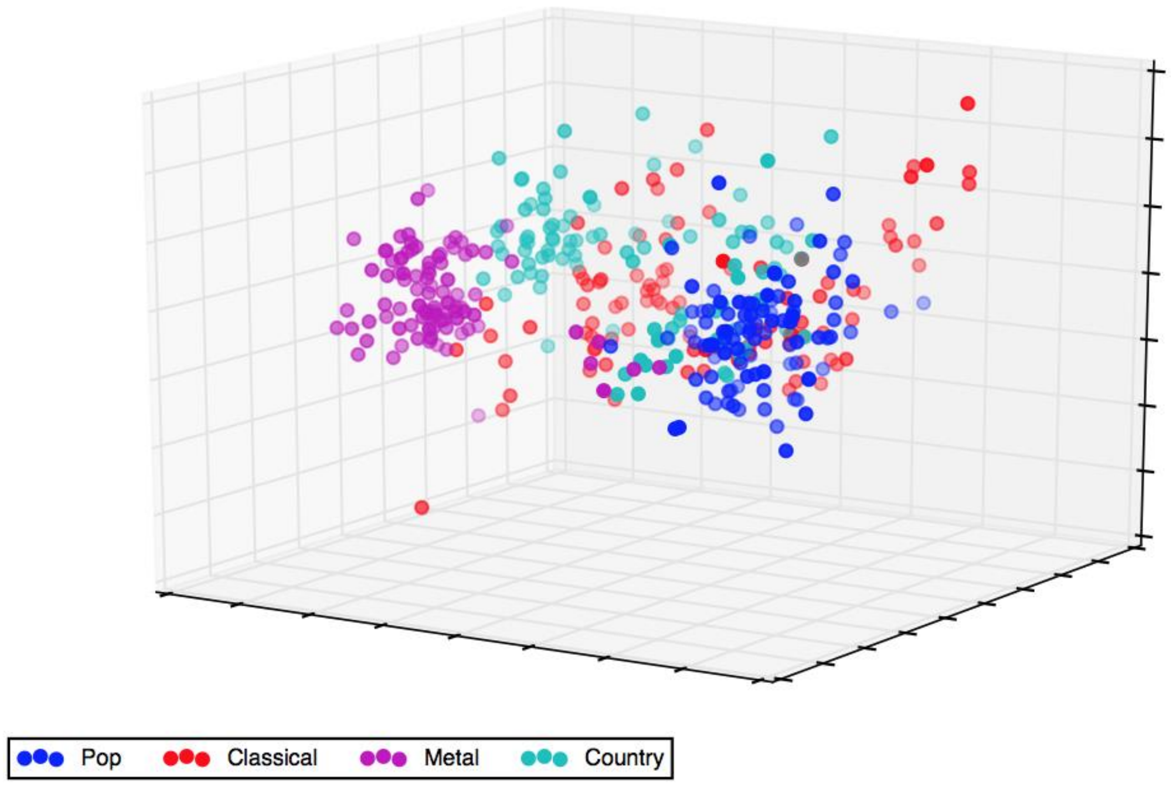

Fig 3.6 Visualization after applying PCA

CHAPTER 4

METHODS

4.1 Support Vector Machines:

The first classifier that we will be using is Support Vector Machines the reason for choosing them is they generalizes well and computationally efficient (just a convex optimization problem) and very successful in practical applications, robust in high dimensions (no problem of overfitting). SVMs is a supervised learning method for classification and regression. We are using python to implement SVMs where SVC, NuSVC and LinearSVC are classes capable of performing multi-class classification on a dataset. Here we have the data of the form of training set (x1, x2, x3,..) where xi is the features of the songs and class labels (y1,y2,y3,…) where yi is the set of genres. If the data is linearly non-seperable, then non-linear SVM will be applied. For implementing SVM in this project we will be importing the svm module from sklearn package. As we are using nonlinear SVM we are going to use NuSVC function of the svm module with the default value of the kernel set to "rbf" (Radial basis function) and degree set to 5. When we are using the rbf function two parameters are the most important to account for

- c
- gamma

c is common for all the kernels in the SVM and it act as replacement for misclassification and the decision functions simplicity. The value of gamma tell us how much weightage is given to a particular training example.

SVM are very effective to use in high dimensional spaces even when the number of dimensions are greater than the number of samples but when the number of dimensions are gereater than the number of sample it may give poor results. It does not considers all the points contained in the decision function (only subset of those points called as support vectors) therefore also being a memory efficient classifier.

The first step in implementing Support Vector Machines is choosing a Loss Function and then minimizing the cost function to get minimum cost on our testing data. Let us consider the example of loss function that is Hinge loss function for a single training example x:

$$l(x)=\max\{0,1-x\}$$

The cost function is then clearly figured out from that point and running stochastic gradient descent to limit the cost function, which gives us a hypothesis function which will be capable of classifying the data similar to our test data. Since our data is so diverse it cannot be separated by a linear support vector machine we will be using nonlinear support vector to classify our data. Below figure 4.1 shows the conceptual diagram of the process of using the Support Vector Machines. The training process analyses our music collection frames to categorize into suitable genres.

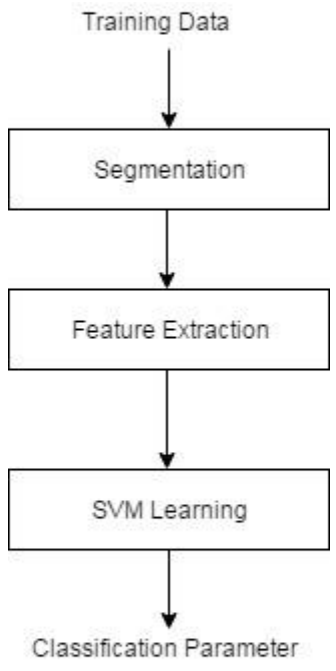

Fig 4.1 SVM Learning Process

4.2 Neural Networks:

The next approach we are using to classify our data is the use of Neural Networks. Neural networks have been previously used in many applications for classification purposes. The have two very basic steps of training a neural network first using a feed forward network to give the calculated result and second backpropagation to modify the weights of the edges to classify the data. The architecture of the neural network is composed of the edges and nodes which are decided by the user and according to need these nodes are divided into layers. The connection between the nodes is acyclic in nature. The connection for the input nodes is set by the user and for all other nodes the activation is computed by the summation of all the incoming activations which is then passed to a function to give us a value which is lying between 0 and 1. One of the most commonly used function for this purpose is hyperbolic tangent function.

Now our neural network architecture is established to train this network and modify the weights which have been selected as default we will apply gradient descent algorithm. Gradient descent then calculates the error at each node which is then used to correct the weights of the network to give us better results. The weights are adjusted till the network start giving approximately correct answer for every set of inputs. The final output is needed for each set of inputs so that we can compute the error moving backwards for every layer. Once our network starts giving corresponding output for each of the inputs in the training data we can use this network to predict the output for our test data.

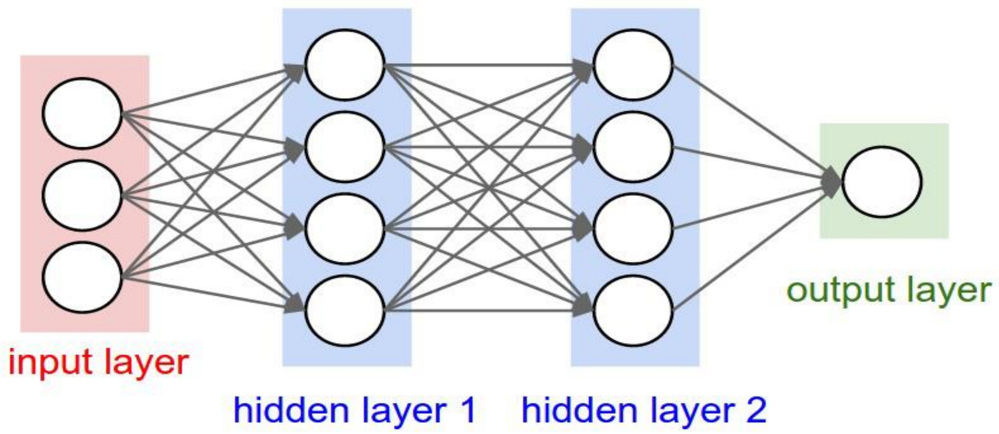

Fig 4.2

Above is an example of a neural network with two hidden layers.

4.3 Decision Trees:

Decision trees are a very simple way of including if else condition based rules into the classifier. For example: If Student has (marks>75) and (participated in co-curricular activities) and (family is poor) then he will be awarded with an scholarship. It is one of the most powerful modeling technique to be used. Decision trees try to sort the input from top to bottom from root to the leaves to classify the input to some output class. The below chart explains the reasons of using a decision tree and where to use them.

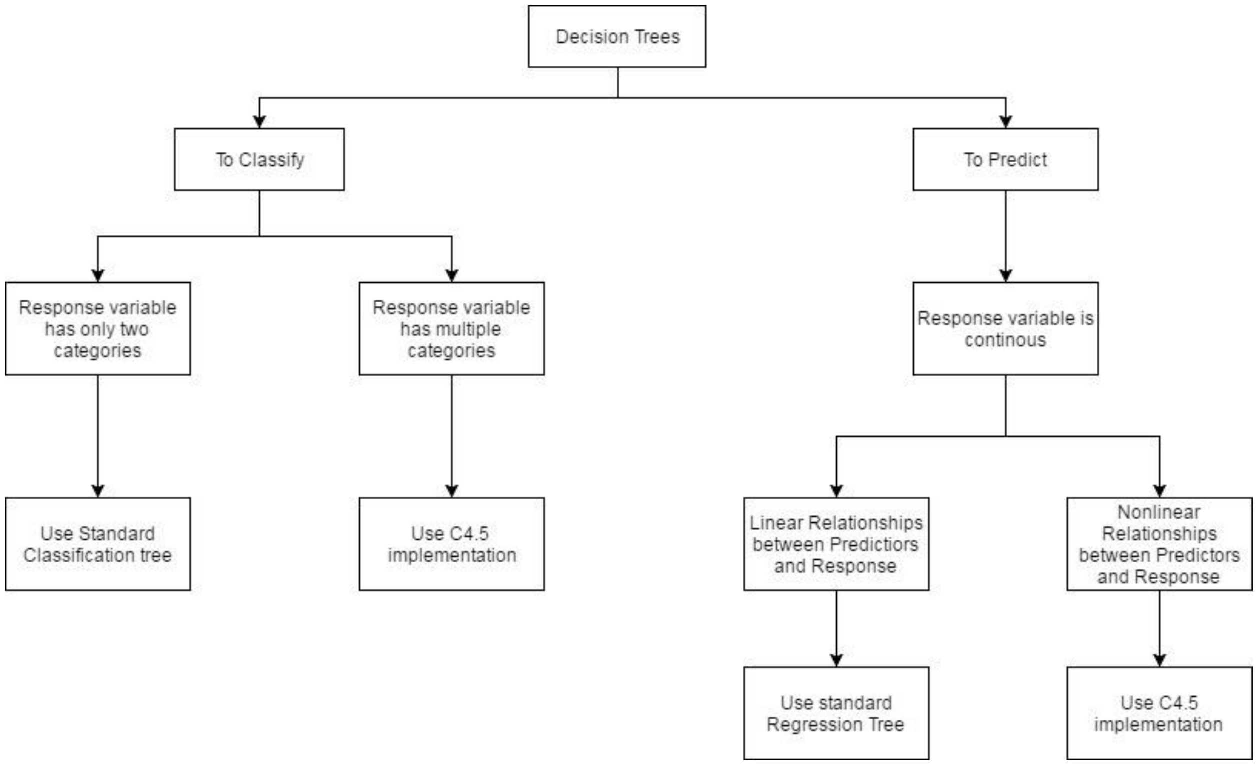
Decision Trees
To Classify
To Predict
Response variable has only two categories
Response variable has multiple categories
Response variable is continous
Use Standard Classification tree
Use C4.5 implementation
Linear Relationships between Predictiors and Response
Nonlinear Relationships between Predictors and Response
Use standard Regression Tree
Use C4.5 implementation

Fig 4.2

There are two aspects to see while constructing a decision tree which attribute to choose and where to stop. The attribute which we have to take depends on Information Gain and Entropy gain. Termination criteria tell us where to stop. Entropy is the measure of uncertainty in the data

$$\text{Entropy(S)} = \sum_{\lim i=1\ to\ l} -|S_i|\ /|S|\ *\ log_2(|S_i|/|S|)$$

S= set of examples

S_i=subset of s with value v_i under the target attribute i= size of the range of the target attribute

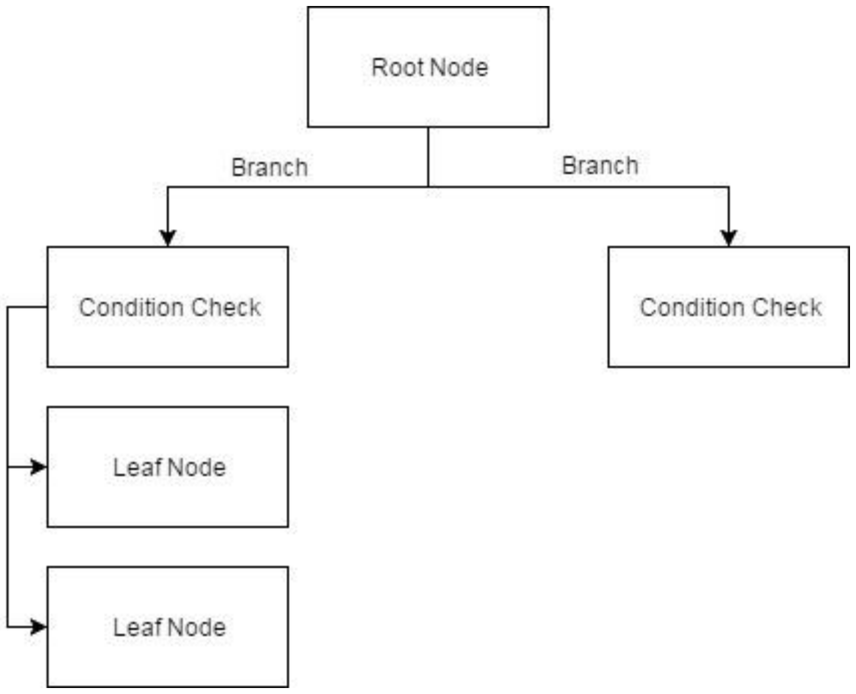

Fig 4.4 Decision tree terms

Termination Conditions:

- All the records at the node belong to one class
- A significant majority fraction of records belongs to one class
- The segment contains only one very small number of records
- The improvement is not substantial enough to warrant making the split

There can also be problem of overfitting with the decision trees as they may grow deeply enough to perfectly classify the training examples when there is noise in the data. When the training sample is too small to model the true target function we have have to resort to pruning the decision trees but generally it is not required. The ways of pruning the trees are

- Stopping the tree from growing more before it reaches the point where it perfectly classifies the data
- First allowing the tree to over fit the data and then prune it.

Building Decision trees is very costly because they take a lot of space as they are exponential in nature. But there are some efficient algorithms present to create efficient decision trees. One such algorithm is Hunts Algorithm we will be using this algorithm through python package for training our data.

Hunts Algorithm:

- Scan the training data and find the most appropriate attribute for the first node.
- Divide the attribute set into parts based on this attribute.
- Repeat the above process for child nodes using those subset of attributes.

In the sklearn package of python there is module sklearn.tree.DecisionTreeClassifier which we have used to create a decision tree for this problem. The input format for the DecisionTree Classifier is of the form where it takes two arrays, the first array is a two dimensional array with [samples, features] and the other array is of the class labels [sample_labels]. After performing the fit operation the model can be used to predict the class.

The reason why we have considered decision trees is it is a technique which is most commonly used for handling multi-output classification problems. It uses a white box model i.e if there a scenario which is of particular interest to us the explanation of the scenario is can be easily explained by the Boolean logic whereas in a black box such is not the case. Once the decision tree is created its running time is logarithmic in nature in relation to the number of data points used in the training of the tree. The drawbacks of using the decision trees can be that it can create overly complex decision trees that do not generalize the data well. The problem of overfitting which can be avoided by pruning (already discussed) and setting the minimum number of attributes required for the leaf nodes and limiting the depth of the tree. Slight variations sometimes can give totally different results so they can be unstable. Constructing a optimal decision tree is a NP-Complete problem as a result practical decision tree problem use heuristic approach like greedy approach.

4.5 K-Nearest Neighbours

Our last approach is one of the earliest applied techniques for the classification problem. K-Nearest Neighbour algorithm is very simple to understand it considers the entire training dataset. When a prediction has to be made for our testing data instance the kNN algorithm will search the entire training dataset to find the k most similar instances those k most similar instances are summarized and analysed to predict the resulting class to which the testing dataset instance belongs. The criteria based on which we measure the similarity of data is based on the type of data if we are dealing with real – valued data Euclidean distance can be used, for other types of data Hamming distance can be used. kNN is a instance based, competitive and lazy learning algorithm. Instance based in the sense that it uses instances of the training data to predict the class of the testing data instance. It is competitive learning algorithm beacause each instance of the training data is competing on the basis of the similarity out of which only k gets selecteed, We say that K-Nearest Neighbours is a lazy learning algorithm because it does not build any model until we want to predict some unseen data. The working of the kNN algorithm comes down to one basic thing similarity index. One of the most powerful choice of similarity index is Eucledian Distance which is what we have used in our code. The Euclidean distance is given as:

$$\mathrm{d}(x, \acute{x}) = \sqrt{(x_1 - \acute{x}_1)^2 + (x_2 - \acute{x}_2)^2 + (x_3 - \acute{x}_3)^2 + \cdots + (x_n - \acute{x}_n)^2}$$

Once we have decided the similarity measure the next steps are as follows:

- This algorithm now computes the d between the unseen testing data instance x and with complete training set to find the most similar k instances. The k instances which are closest to testing instance can be formed into a set Z.
- It then finds the conditional probability of each class with the set Z to give us the most probable class for the given test data.

and finally our test data x gets assigned to the most probable class of the all.

Simple example to understand our idea here is that suppose we are coloring each each sample of our training data with the labels and then choosing the k nearest samples for the testing data. An illustration of which can be seen in the below figure 4.4

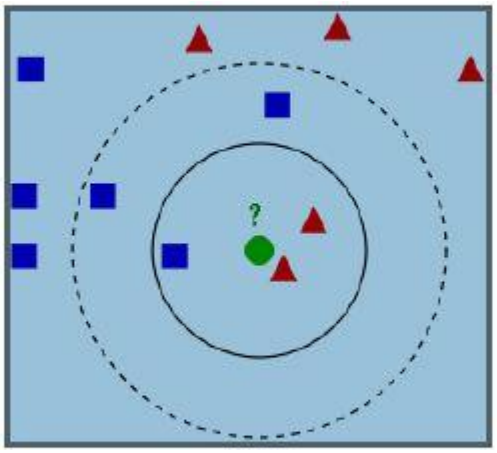

Fig 4.5

In the above figure let us suppose we have two classes one is blue squares and the other is red triangle. All these classes have some space in which they lie which we call feature space since here each instance of these classes will have x and y coordinate they will lie in a 2D space depending upon the number of features it can have N-dimensional space. For this example we are considering 2D space only. Now we have to find the class of the green circle to which class does it belong whether blue squares or red triangles. The basic approach will be to associate it with the nearest neighbours class which is red triangle but it may not be wholly correct as it may be surrounded by blue square but just red triangle is nearest to it in which case it would be incorrect classification. So we decide an appropriate value of k which will give us the best results like in the above example for k=3 it belongs to red triangle but for k=5 it belongs to blue square. We can also see that for k=4 it is tie therefore it is best if we choose an odd value for k. Some plus points of kNN algorithm which made us use it in our project is it fairly simple and easy to implement and with zero or very low training it can be employed to work. It can directly work on the multi classification problems whereas other algorithms mainly deal with binary classification. Its one advantage can also act as disadvantage because of its on the go run it is very expensive on testing data as in comparision to Neural Networks where training phase may take some time but its testing phase is very fast comparatively. We have used sklearn package in python to implement kNN algorithm. kNeighboursClassifier is a function present in neighbours library of skelarn module.

CHAPTER 5

RESULTS AND ANALYSIS

The dataset we have used “GTZAN genre collection” has 10 different genres, however we have tested our methods on the following 6 genres: Classical, Jazz, Country, Pop, Rock, Metal. The reason why we have considered 6 genres out of 10 are: Firstly, there was some genres were more similar to each other than others which made it difficult for predicting for example blues and jazz. Secondly by having all the genres the number of classes would have been very large which would not have given us better results with many of the above machine learning algorithms.

In our dataset out of 100 songs of each genre we have used 70 songs for the training purpose and the rest 30 song for the testing part. We know there could be a high possibility of over fitting the data as each of our song was represented by an extremely large matrix of 3050 by 39. Because of this reason we have applied PCA to our dataset and then applied the above algorithms over the new smaller dataset. But after applying we saw that PCA was able to improve the result of only K Nearest Neighbors algorithm while all the other algorithms performed better without the application of the PCA. Greater part of our algorithms were greatly precise on the training set, giving accuracy around 93% which increased the likelihood overfitting of our data.

	Metal	Classical	Pop	Country	Jazz	Rock
Metal	27	0	0	2	0	1
Classical	2	25	0	2	1	0
Pop	0	0	27	2	0	1
Country	1	2	0	25	0	2
Jazz	1	1	1	1	24	2
Rock	1	0	1	0	1	27

Table 4.1 **Neural Network**

	Metal	Classical	Pop	Country	Jazz	Rock
Metal	27	0	0	1	1	1
Classical	3	20	0	4	0	1
Pop	0	0	26	2	2	0
Country	2	1	2	22	0	3
Jazz	0	1	2	0	26	1
Rock	0	0	2	0	4	24

Table 4.2 **SVM Linear Kernel**

	Metal	Classical	Pop	Country	Jazz	Rock
Metal	29	0	0	1	0	0
Classical	5	18	0	6	0	1
Pop	1	0	27	0	2	0
Country	1	4	1	24	0	0
Jazz	0	2	1	0	25	2
Rock	1	1	2	1	4	21

Table 4.3 **SVM Polynomial Kernel**

	Metal	Classical	Pop	Country	Jazz	Rock
Metal	26	0	0	2	2	0
Classical	7	20	0	1	1	1
Pop	0	0	28	0	0	2
Country	2	4	0	23	0	1
Jazz	0	1	1	3	24	1
Rock	1	0	2	0	1	25

Table 4.**3 k-NN with PCA**

	Metal	Classical	Pop	Country	Jazz	Rock
Metal	26	3	0	0	1	0
Classical	1	27	1	0	0	1
Pop	0	2	20	8	0	0
Country	0	0	7	20	0	3
Jazz	0	5	0	0	23	2
Rock	0	0	0	0	4	26

Table 4.5 **Decision Tree**

To know the effectiveness of these algorithms in detail we may look at the precision and recall values of these algorithms and we have also looked at the F_1 score of these algorithms. Since our problem is a multi-classification one we have calculated the F_1 score for each of the genres and then taken the mean average of those values. All these values are given in the below table:

Model	Test-Static	Metal	Classical	Pop	Country	Jazz	Rock	F-Score
Neural Network	Recall	93%	83%	93%	90%	92%	90%	0.9
	Precision	90%	93%	100%	79%	83%	87%	
SVM-Linear Kernel	Recall	97%	67%	93%	78%	87%	89%	0.847
	Precision	78%	95%	93%	83%	90%	85%	
SVM-Polynomial Kernel	Recall	97%	63%	97%	80%	94%	91%	0.838
	Precision	81%	83%	97%	77%	85%	76%	
K-NN with PCA	Recall	93%	73%	93%	77%	88%	82%	0.842
	Precision	76%	81%	100%	82%	88%	73%	
Decision Tree	Recall	90%	93%	67%	67%	70%	77%	0.793
	Precision	96%	90%	65%	67%	76%	70%	

Table 4.7 **P/R Values**

In the above table we can clearly see that Neural Network has performed very well in comparison to all the other algorithms. All the other algorithms have also done pretty decent job in the classification one of the reasons for this good performance of these algorithms is MFCC which has given us good representation of our dataset as stated in the Feature extraction section.

We have seen P/R (Precision and Recall) values as we have earlier discussed in Feature extraction chapter that there is one more to measure the effectiveness of a Classifier i.e ROC curves. We have already seen the difference between the let us look at the difference in to formula of the two methods.

$$\text{Recall}=\frac{TP}{TP+FN} \qquad \text{Precision}=\frac{TP}{TP+FN}$$

$$\text{FPR}=\frac{FP}{FP+TN} \qquad \text{TPR}=\frac{TP}{TP+FN}$$

Where TP= True Positive (Predicts Positive when instance is Positive)

FP= False Positive (Predicts Positive when instance is Negative) Error Type -I

FN=False Negative (Predicts Negative when instance is Positive) Error Type-II

TN=True Negative (Predicts Negative when instance is Negative)

Since we have seen that the Neural Networks has performed the best out of all the algorithms we will see the ROC curves of Neural Networks. The ROC curves and P/R curves both are available for the binary classification problems so we will take one versus all other genres to see the ROC curve for the neural network classifier. The below figure 5.1 shows the six ROC plots for each genre. In the previous ROC plot we had seen that only classical was able to perform but here we can clearly see that Neural network has been able to predict all the genres quite successfully and we can even see that Jazz and metal are at almost 1.0 AUC.

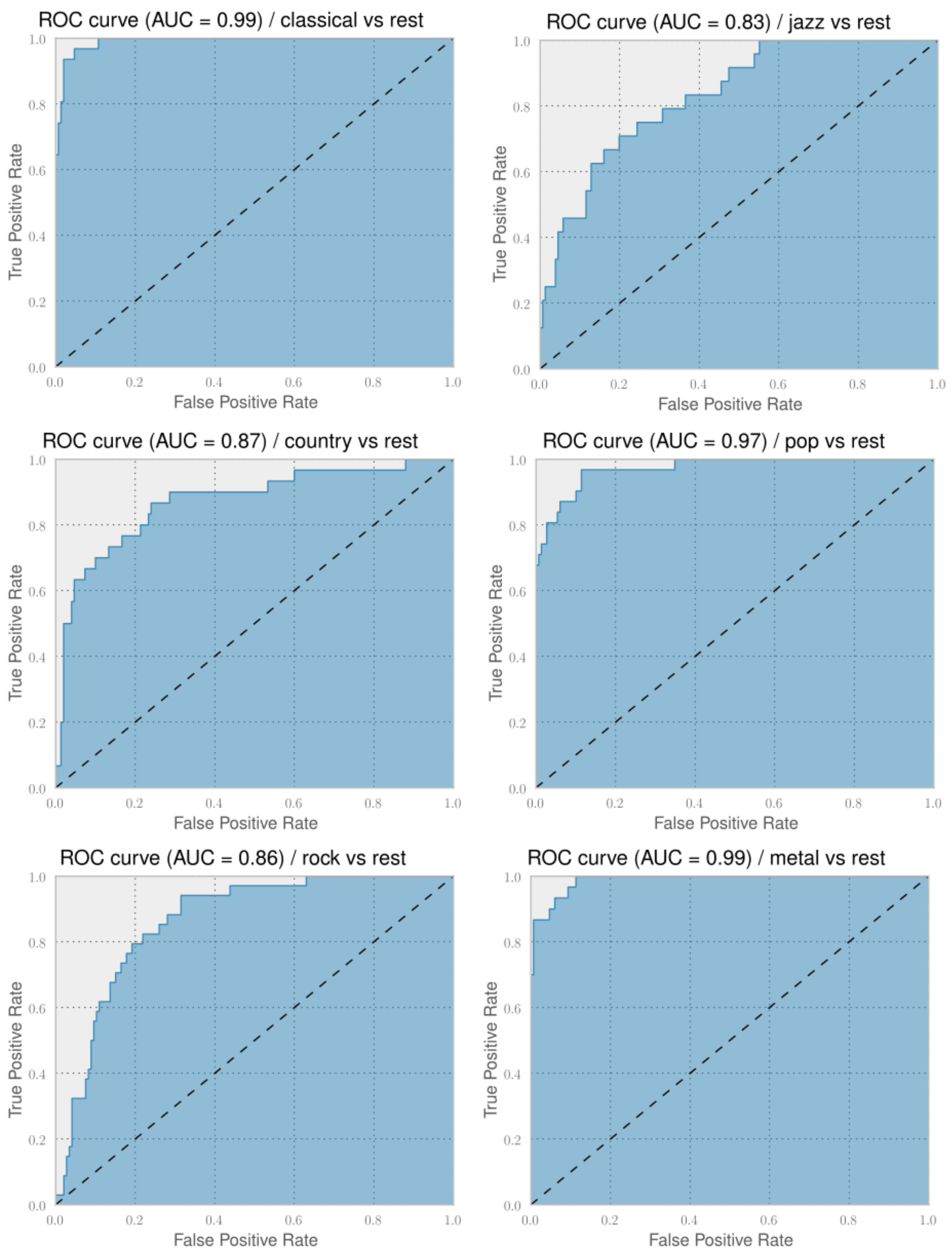

Fig 5.1 ROC Curves for Neural Network

CHAPTER 6

CONCLUSION AND FUTURE WORK

We have been effectively able to build software which could classify different genres of music. It can be seen as the starting step in building music recommender systems, it can be thought of as a well laid foundation for future projects. We have seen that Neural Network has worked the best in all the algorithms we have applied probably because of the nature of the data as it was a very large dimensional continuous vector. One more thing that we came to our notice was the absence of overfitting in this type of data to be surprising.

In this Project we have seen what MFCC feature are and how they accurate they are in representing the feature of music files. We have seen that MFCC has outperformed LPC and LPCC and have achieved a far greater accuracy. But to increase the accuracy of our results furthermore we can also look for other feature extraction methods like Distortion Discriminant Analysis it has been gaining popularity recently and may provide some improvement over MFCC. We would also like to see some changes in the MFCC by varying the length of the sample data chosen in MFCC and frequency of windowing done. We could also be able to see new and interesting results if we are able to relate tempo, beat, and pitch in this classification system.

In Future we would also like to see a large & complex dataset with more genres and increased number of samples of each genre, and cross check our algorithms with new data. We would see how our algorithms perform for real world data where there are thousands of genres with very subtle feature differences. For future purposes we could also implement some more algorithms to see if it could give us better results than Neural Networks.

REFRENCES

1. Li, Tao, Mitsunori Ogihara, and Qi Li. "A comparative study on content-based music genre classification." *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, 2003.
2. Haggblade, Michael, Yang Hong, and Kenny Kao. "Music genre classification." *Department of Computer Science, Stanford University* (2011).
3. Lippens, Stefaan, Jean-Pierre Martens, and Tom De Mulder. "A comparison of human and automatic musical genre classification." *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*. Vol. 4. IEEE, 2004.
4. Tzanetakis, George, and Perry Cook. "Musical genre classification of audio signals." *IEEE Transactions on speech and audio processing* 10.5 (2002): 293-302.
5. Li, Tao, and Mitsunori Ogihara. "Toward intelligent music information retrieval." *IEEE Transactions on Multimedia* 8.3 (2006): 564-574.
6. West, Kristopher, and Stephen Cox. "Features and classifiers for the automatic classification of musical audio signals." *ISMIR*. 2004.
7. Allwein, Erin L., Robert E. Schapire, and Yoram Singer. "Reducing multiclass to binary: A unifying approach for margin classifiers." *Journal of machine learning research* 1.Dec (2000): 113-141.
8. Umapathy, Karthikeyan, Sridhar Krishnan, and Shihab Jimaa. "Multigroup classification of audio signals using time-frequency parameters." *IEEE Transactions on Multimedia* 7.2 (2005): 308-315.
9. McKinney, Martin, and Jeroen Breebaart. "Features for audio and music classification." (2003).
10. Aucouturier, Jean-Julien, and Francois Pachet. "Representing musical genre: A state of the art." *Journal of New Music Research* 32.1 (2003): 83-93.
11. Erzin, Engin. "Automatic classification of musical genres using inter-genre similarity." *IEEE Signal Processing Letters* 14.8 (2007): 521-524.
12. Meng, Anders, et al. "Temporal feature integration for music genre classification." *IEEE Transactions on Audio, Speech, and Language Processing* 15.5 (2007): 1654-1664.
13. Lidy, Thomas, and Andreas Rauber. "Evaluation of feature extractors and psycho-acoustic transformations for music genre classification." *ISMIR*. 2005.
14. Panagakis, Ioannis, Emmanouil Benetos, and Constantine Kotropoulos. "Music genre classification: A multilinear approach." *ISMIR*. 2008.

15. Kim, Hyoung-Gook, Nicolas Moreau, and Thomas Sikora. "Audio classification based on MPEG-7 spectral basis representations." *IEEE Transactions on Circuits and Systems for Video Technology* 14.5 (2004): 716-725.

16. Lambrou, Tryphon, et al. "Classification of audio signals using statistical features on time and wavelet transform domains." *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. Vol. 6. IEEE, 1998.

17. Lee, Chang-Hsing, et al. "Automatic music genre classification using modulation spectral contrast feature." *Multimedia and Expo, 2007 IEEE International Conference on*. IEEE, 2007.

18. Xu, Changsheng, et al. "Musical genre classification using support vector machines." *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*. Vol. 5. IEEE, 2003.

19. Lu, Lie, Hao Jiang, and HongJiang Zhang. "A robust audio classification and segmentation method." *Proceedings of the ninth ACM international conference on Multimedia*. ACM, 2001.

20. Shan, Man-Kwan, and Fang-Fei Kuo. "Music style mining and classification by melody." *IEICE TRANSACTIONS on Information and Systems* 86.3 (2003): 655-659.

21. Chai, Wei, and Barry Vercoe. "Folk music classification using hidden Markov models." *Proceedings of International Conference on Artificial Intelligence*. Vol. 6. No. 6.4. sn, 2001.

22. Dannenberg, Roger B., Belinda Thom, and David Watson. "A machine learning approach to musical style recognition." (1997).

23. Matityaho, Benyamin, and Miriam Furst. "Neural network based model for classification of music type." *Electrical and Electronics Engineers in Israel, 1995., Eighteenth Convention of*. IEEE, 1995.

24. Soltau, Hagen, et al. "Recognition of music types." *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. Vol. 2. IEEE, 1998.

25. Han, Kyu-Phil, et al. "Genre classification system of TV sound signals based on a spectrogram analysis." *IEEE Transactions on Consumer Electronics*44.1 (1998): 33-42.

26. Pye, David. "Content-based methods for the management of digital music." *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*. Vol. 4. IEEE, 2000.

27. Jiang, Dan-Ning, et al. "Music type classification by spectral contrast feature." *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*. Vol. 1. IEEE, 2002.

28. KAR, AMLAN, and CHAITANYA AHUJA. "MUSIC CLASSIFICATION USING NEURAL NETWORKS."

29. Burges, Christopher JC, John C. Platt, and Soumya Jana. "Distortion discriminant analysis for audio fingerprinting." *IEEE Transactions on Speech and Audio Processing* 11.3 (2003): 165-174.

30. Rabiner, Lawrence R. "A tutorial on hidden Markov models and selected applications in speech recognition." *Proceedings of the IEEE* 77.2 (1989): 257-286.

31. Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks." *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013.

32. Matityaho, Benyamin, and Miriam Furst. "Neural network based model for classification of music type." *Electrical and Electronics Engineers in Israel, 1995., Eighteenth Convention of*. IEEE, 1995.

33. Soltau, Hagen, et al. "Recognition of music types." *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. Vol. 2. IEEE, 1998.

34. Li, Dongge, et al. "Classification of general audio data for content-based retrieval." *Pattern recognition letters* 22.5 (2001): 533-544.