Automated Music Genre Classification

Arjun Jajoo
adjajoo@asu.edu

Garvit Khandelwal
gkhandel@asu.edu

Pradhuman Swami
pswami1@asu.edu

Siddhant Prakash
sprakas9@asu.edu

Abstract

Music genres form a natural way to organize audio since they share similar rhythm and texture. We build an automated genre classification model that will use features representing timbral, rhythmic and pitch analysis of the audio. We train various classifiers like k-NN, SVM, Logistic Regression, Neural Network on the GTZAN dataset provided by MARYSAS[1]. We are able to get good accuracy using ensemble voting classifier and support vector machine on both 10-genre and 4-genre classification.
Keywords: Genre, classification, k-NN, SVM, GTZAN dataset, MFCC, ensemble classifiers

1 Introduction

With the advent of digital music, it has become very important to group music files for various tasks like search-retrieval and recommender systems. Manual annotation of such a huge dataset is an impossible task, and hence ”Automatic Music Genre Classification” has been a widely studied research topic in the field of Multimedia Information Retrieval (MIR).

Music streaming applications like Pandora[26] and iTunes[27] find it very useful to categorize the songs based on genres so that they can give better song recommendations to the end user. Search based applications like Shazam[28] also use music classification to speed up the searching time. We try to address the domain of automated music genre classification in this project. The metadata of songs do not contain the genre in general, so there is a necessity to find ways to annotate genres of music files. In this project, we try to build a machine leaning model that can automatically classify any music file into different genres.

This work is organized as follows. Section 2 mentions the problem statement about this project. Section 3 provides a literature review of the related works ongoing in the field of Music Genre Classification. In section 4, we provide the details of the dataset used in this project. Section 5 includes the feature extraction and selection details and Section 6 covers the overall approach where in the classification models are explained. The various experiments conducted throughout this project are talked about in detail in section 7. Next, we present our conclusions and learnings in section Section 8. Possible extension to our work has been mentioned in Section 9.

2 Problem Description

Our project aims at building a machine learning model to classify a music song into different genres namely blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae and rock. We also try to explore the genres that are have differentiating features and focus on a 4-genre classification problem in the later half of our project. We use content based features from audio analysis in time and frequency domain and use a 30 second audio clip as one training example. We have used GTZAN dataset for our classification task that is explained in detail in section 4. We build an interface that will take input as any song, and by segmenting it into 30 second clip, it will predict the genre of the song.

3 Related Work

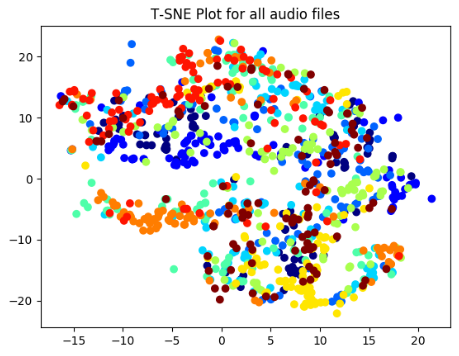
Automated Music Genre Classification has been studied by numerous researchers and still remains a challenging topic in Music Information Retrieval(MIR) community due to the fuzzy nature of features and the ambiguity associated with human perception of genres. Ground breaking work in this field was performed by Tzanetakis, et al in [1], by using acoustic features through audio analysis done on a dataset consisting of 1000 audio files. This dataset, now provided by MARSYAS, has been widely used in approaching this problem. This work has also proposed various content based features which we are using in our approach as well.

In recent works, researchers have also experimented with other features like a combination of audio-visual in [6], and lyrics based classification done by Howard et al in [7]. In the work of [5],C.H. Lee et al proposed that Mel-Frequency Cepstral Coefficients(MFCC) was a dominant content based feature that works well in music genre classification due to its natural modeling of human auditory perception and helps in achieving higher accuracy.

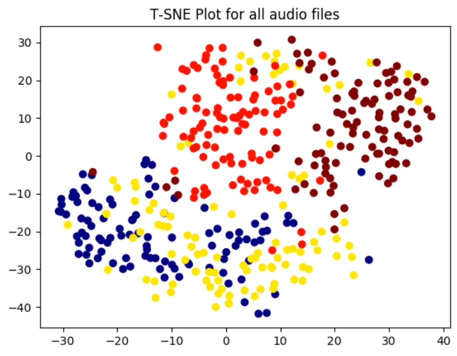
In this project, we try to experiment various models like SVM, k-NN, etc and an ensemble of these models for various content based features that describe the texture, pitch and rhythm of any audio clip.

4 Dataset

For this project, we have used GTZAN dataset [1] provided by MARSYAS website. The dataset contains 1000 audio clips of 30 seconds each. The music files are spread across 10 genres, namely blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae and rock. The dataset is evenly spread across the 10 genres, i.e. 100 files corresponding to each genre. Each audio file is 30 seconds long and is encoded at 22050Hz, 16 bit per sample. We have used two-thirds of the dataset as training set and one-third for testing.



(a) Visualization 10-Genre



(b) Visualization 4-Genre

Figure 1: T-SNE Plot for GTZAN dataset

For music genre prediction, we first tried to analyse the data by plotting the T-SNE plot [8] in 2 dimensions. As evident from the above figure, 10-class genre prediction is very complex as the data is not separable. However, after various experiments, we were able to see that 4-class genre prediction is relatively simpler when we use genres as : Classical, Jazz, Metal and Pop. This is because these genres have very distinct feature space and also easy to perceive from a human perspective.

5 Feature Extraction and Selection

5.1 Feature Extraction

Feature extraction and selection is the first step in any machine learning model. Once the significant features are extracted, one can train various models for the task of classification. For our problem statement, we need to compute the features from the audio analysis of input music file.

We extracted the features corresponding to three domains that answer different characteristics of an audio file. These are : Timbre, Rhythm and Dynamic Pitch. Since the audio file is continuous, we needed to segment each audio into shorter clip to get the changes in characteristics over time. Since the texture or rhythm of a song is prevalent for a short duration, our choice of segmentation frame had to be small. We chose to split the audio into 30 sec analysis window that will be classified into the different genres. Each analysis window is further divided into frames of 250 milliseconds. Feature computation takes place at the frame level firstly. We used various content based features as suggested by [1] and added other combinations of features to improve the accuracy.

The extraction of features are done in either time domain or frequency domain analysis.

Time Domain: Any audio file is stored as time varying waveform where we have discrete samples representing the signal. For a file having sampling rate of 22050Hz, we store 22050 samples, each sample being 16 bit integer. The time domain analysis leads to various features like amplitude of the signal, the mean amplitude(energy) or the zero-crossing rate.

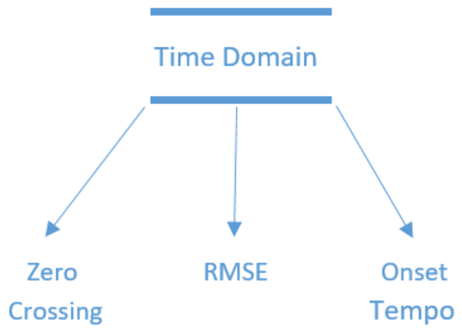


Figure 2: Time Domain Features

Frequency/Spectral Domain: Apart from the time domain representation, the audio file can also be represented as sum of sinusoidal waves of different frequencies. Such a representation where we plot the frequency intensity over the entire bandwidth is called the spectral domain representation of the audio file. To get the frequency spectrum of any signal, we use Fast Fourier Transform and get the coefficients corresponding to each frequency bin. Using the frequency spectrum, we can analyze the primary frequencies of any signal. We use features like spectral centroid, spectral roll off in the frequency domain analysis.

Now we describe each feature in detail. There are three kinds of features that are used for Music Genre Classification. These are :

- **Timbral Features**: Timbral texture features give the changes in frequency spectrum in a signal over time. When the music is a mixture of sounds, the changes in texture is a very useful differentiating factor. Such features are widely used in tasks like Speech-Music discrimination, speech recognition. When the rhythm and pitch is not sufficient to classify the music file, the timbral features are observed to be the key feature. Percussion instruments are generally having differentiating timbral features. In order to compute the timbral features, we first take the short time fourier transform(STFT).

  Before we understand the timbral features in detail, let us look at the basics of short time fourier transform.

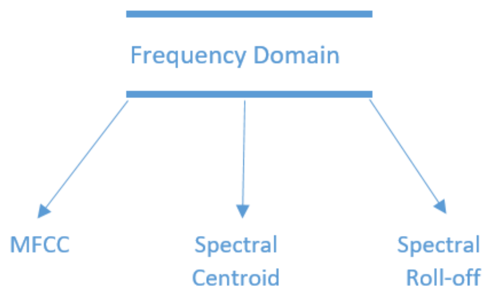


Figure 3: Frequency Domain Features

Short time fourier transform [23], [29] is applied on a continuous signal by first breaking them into shorter signals like frames. To maintain the continuity, generally the window function is overlapped over one another. Convoluting with some window function like Hamming Window or Hanning window tends to reduce the noise after the fourier transform. The equation for short time fourier transform is as follows:

$$\text{STFT}(x(t))(\tau,\omega) = X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)\exp^{-j\omega t}\,dt \tag{1}$$

x(t) is the time varying audio signal

w(t) is the window function like Hanning/Hamming window

$X(\tau,\omega)$ is the frequency domain signal

For the purpose of discrete audio signal, we compute STFT as:

$$\text{STFT}(x[n])(m,\omega) = X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w(n-m)\exp^{-j\omega n} \tag{2}$$

The function $X(m,\omega)$ is a complex function depicting the phase and magnitude of the audio signal over frequency and time.

The next step is to compute the spectrogram or the frequency coefficients. This is done by taking the magnitude of the STFT.

$$\text{spectogram}(x(t))(\tau,\omega) = |X(\tau,\omega)|^2 \tag{3}$$

Now, we explain the various timbral features that we are using.

– Mel-Frequency Cepstral Coefficient(MFCC):Mel-Frequency cepstral coefficients are the most widely used timbral audio feature and finds its use in almost all speech recognition and speech classification problems.

  In order to compute MFCCs, first, the signal is divided into multiple short frames. For each frame, the short time fourier transform is computed. The frequency spectrum is then plotted on a Mel-scale that consists of 2 filters - linear upto frequency range of 1KHz and logarithmic after that.

$$M(f) = 1125\log(1+\frac{f}{700}) \tag{4}$$

  The Mel-scale models the human auditory perception and hence is a useful feature to use in such problems. Triangular window mapping is used to plot the frequency spectrum on the mel-scale. Below is a graphical representation of the triangular window used.[24]

  After taking log of the output, we calculate the Discrete Cosine Transform and save the first 12 coefficients as the rest of the coefficients are less meaningful. Here is an example of all the steps involved in MFCC computation.

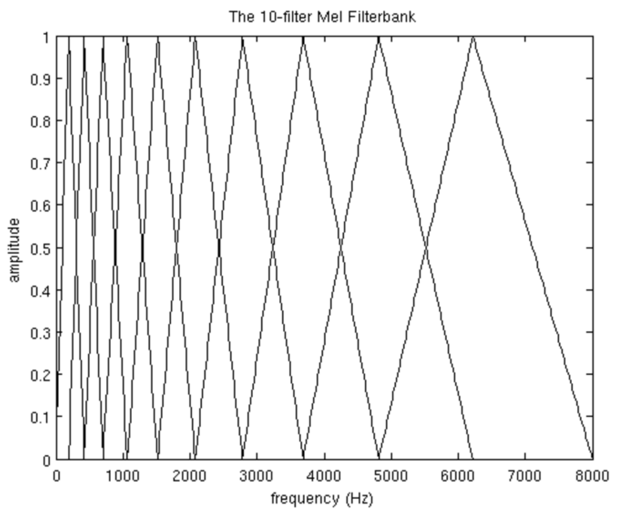


Figure 4: Mel Frequency Triangular Window

- Spectral Centroid : Spectral centroid is a way to measure the brightness of a sound. It is the mean frequency component in the frequency spectrum of any audio signal. For any frame of 250ms, we get the Spectral centroid using below equation:

$$\text{Spectral Centroid} = \frac{\sum_{k=1}^{n} kF[k]}{\sum_{k=1}^{n} F[k]} \tag{5}$$

  Where F[k] is the amplitude corresponding to the kth frequency bin.
  Spectral centroid is useful in finding the centre of gravity of the frequency spectrum. It is observed [30] that the spectral centroid for pop/rock music is higher than that of classical music samples. Also, for rock music, the fluctuation in spectral centroid is more than that of classical music. Such variations in this feature made it worth experimenting for our problem of Genre classification.

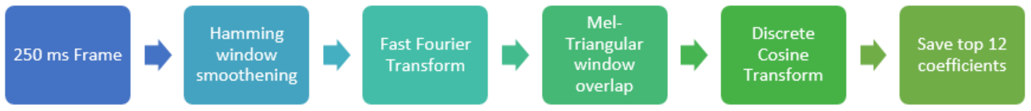

250 ms Frame
Hamming window smoothening
Fast Fourier Transform
Mel-Triangular window overlap
Discrete Cosine Transform
Save top 12 coefficients

Figure 5: Flow Diagram of MFCC Computation

- Spectral Roll-off: Spectral roll-off is measured as the frequency component that is above 85% of the frequency spectrum for any audio signal. In order to get the roll-off, first we calculate the frequency spectrum. Next we calculate the roll-off as below:

$$\sum_{n=1}^{R_t} M_t[n] = 0.85 \sum_{n=1}^{N} M_t[n] \tag{6}$$

• **Rhythmic Features**: Rhythm is a very natural way for humans to identify genres among different music files. The rhythm could be estimated through various measures like beat, tempo, rubato and gives us the periodicity of the frequency components for any music file. In order to take the rhythmic characteristics for our classification model, we estimated the onset tempo.

  - Estimated Tempo: The tempo of any music file is estimated based on the average beat per minute of the most dominant frequency in that file. First the onset autocorrelation is plotted for the entire duration and the first periodicity, thus observed is marked as the estimated tempo of the track.

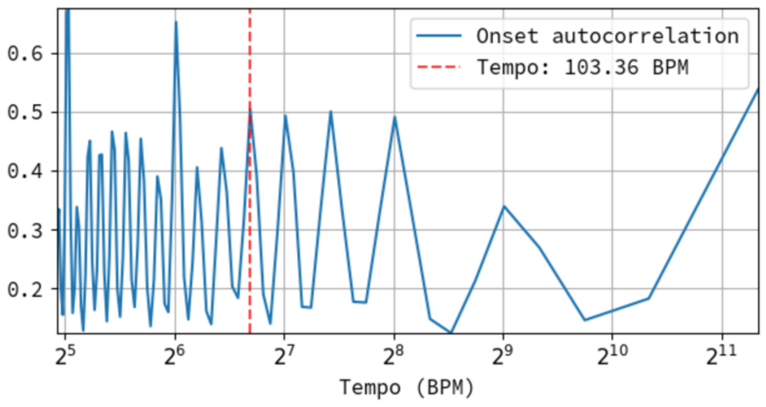


Figure 6: Tempo Estimation

The above figure, taken from Librosa documentation [2], shows the onset autocorrelation and the estimated tempo from the plot.

- **Pitch/Dynamic Features**: Pitch and other dynamic features for any audio file is useful in identifying variations in the sound over time. Features like zero crossing rate and energy tend to indicate the noisiness or the entropy in the signal.
  - Short Time Energy: Short time energy is a time domain feature computed by taking the mean square of amplitudes of the samples within a frame.

$$E_n = \frac{1}{N}\sum_m [x(m)w(n-m)]^2 \tag{7}$$

    Where, En = Energy of the frame
    x(m) = sample
    w(n-m) = Window Function

  - Low Energy: Another way to interpret the short time energy over the entire sample is to compute the low energy for the music file. As discussed above, the short time energy tends to give the mean energy of an individual frame. To aggregate this for the entire sample, low energy is defined as the percentage of frames that have RMS energy less than the mean RMS energy of all the frames.
  - Zero Crossing Rate: As evident from the name itself, zero crossing rate of a signal is the number of times the signal changes sign from positive to negative or vice versa. We get the zero crossing rate for any frame as per below equation:

$$Z_t = \frac{1}{2}\sum_{n=1}^{N} |sign(x[n]) - sign(x[n-1])| \tag{8}$$

  Although the pitch related features are predominantly used for distinguishing music against speech content, the aggregation of these features with others have shown to improve the results of music genre classification.

5.2 Feature Aggregation

In the above section, we have explained the steps required to calculate the features for a frame. Apart from the estimated tempo and low energy feature, rest all the features are computed on a frame by frame basis for a time interval of 250ms. In order to aggregate the features for the entire window of 30 seconds, we used the gaussian mixture assumption as per [4]. Each feature is estimated as

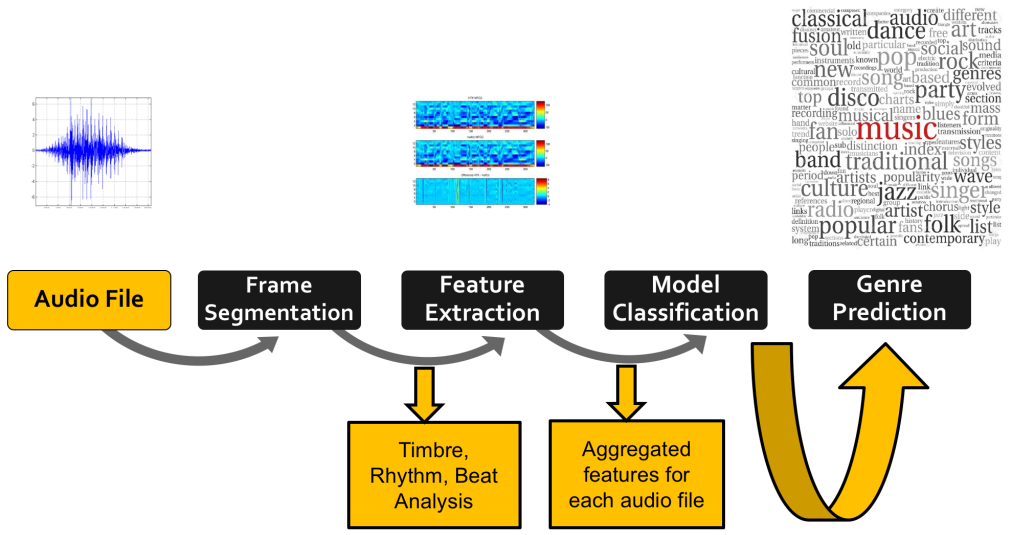


Figure 7: Overall System Implementation

random variable from a Gaussian mixture. Hence, mean and variance are the typical values that are used for each feature.

For representing the Mel-Frequency cepstral coefficients, we had to use 12 coefficients for each frame. And we had 1200 frames for the entire window, which resulted in a very huge feature dimensionality. With our aggregation technique, now we represent MFCCs as 12 mean MFCCs and 12*12 covariance matrix for the different coefficients resulting in 156 features.

Similarly, for every other feature, we compute the mean and variance over the entire 30 second window. Our resulting input feature dimensionality, thus becomes 164 that we use in our model as explained in the next section.

6 Methodology

In the previous section, we looked at different ways in which we extract the features for our classification task. In this section, we explain the complete pipeline from input music file to output genre label that we obtain for the given dataset. We can divide our method in the given steps.

1. Window selection,
2. Segmentation and noise reduction,
3. Feature computation and aggregation by frames,
4. Model creation and training on feature vectors, and
5. Genre classification on trained model.

The overall system diagram for the above method is shown in Figure 7.

6.1 Window selection

The first part of the feature extraction process is to select audio window to extract its features. We fix our length of audio to be **30 seconds**. This helps us give a uniform length for all audio tracks and avoid biasing the audio towards a particular genre by virtue of it being longer or shorter compared to other tracks.

6.2 Segmentation and noise reduction

Once we extract the 30 seconds audio, we then separate the frames from them. We segment each audio into frames of 250 millisecond. The overlapping frames are segmented using hamming window over 100 milliseconds which is use to smoothen the noise. Hamming window [9] is a typical sliding window used in signal theory to smoothen the energy distribution of a signal and overcome the noisy variations in the signal.

6.3 Features computation and aggregation by frames

Now we have our frames, we apply the methods mentioned in Section 5, and get the features out of the audio frame-by-frame. After extraction, we aggregate the features for the entire 30 second window by frames, using a Gaussian assumption. The assumptions is that every feature is a Gaussian random variable picked for different frames.

Thus, we obtain the following mentioned different types of features.

- New features - MFCC (12 mean and 12*12 covariance matrix)
- spectral centroid (mean and standard deviation)
- spectral roll-off (mean and std deviation)
- onset tempo (mean estimated tempo)
- zero crossing rate (mean and std deviation)
- short time energy
- low energy mean

6.4 Model creation and training on feature vectors

Using the above process we obtain a 164 dimensional feature vector which we then use to train our various classification models. A number of classification techniques have been studied in literature for various different multi-modal data. For our given data and for our particular task, we studied and found out the classifiers which has proven to be a step ahead than all other types of classifier. Here, we describe each of the classifiers and their training procedure for model creation.

6.4.1 k-NN Classifier

k-NN is an instance based learning algorithm in which we assume an independent and identical distribution of data points from some probability distribution [10]. Each data point is represented as a vertex point on a graph and the edge weights of the graph is assumed to represent the similarity between the data points. For symmetric k-nearest neighbor, the graph is an unweighted graph. The most common approach studied in the literature is to connect each vertex to its k-nearest neighbors, i.e the k most similar data points in its vicinity. The similarity is calculated using a distance function between two nodes. The lower the distance function, the similar the nodes are. We have used L_2-norm as the distance function in our experiments.

Thus, for classification purpose, in training each of the data points are represented in the 164 dimensional feature space with each data point, i.e audio sample in our case, represented by a 164 dimensional vector. The k-nn model created contains the graph representation of all the training data points along with their labels.

A representative diagram for k-NN classification can be seen in Figure 8.

6.4.2 SVM Classifier

Support Vector Machine is a supervised learning technique in which the goal is to come up with a single decision boundary to classify all the data points in to the given classes. Given a set of labeled data points, in the training process the goal is to come up with a good enough decision boundary which will minimize the error in classification of data points.

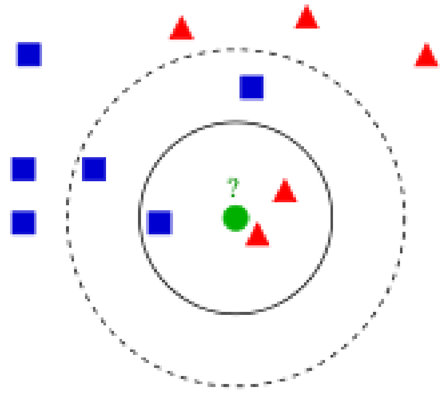


Figure 8: k-NN classification

SVM Formulation

Support vector machine [11] works by estimating the support vectors (Figure 9) which are the data points closest to the decision hyperplane, separating the classes. The error is calculated by summing the distance of the mis-classified examples from the decision hyperplane. Since ours is a multi-class labeling problem, and support vector machine finds hyperplane for classifying two classes. Thus, we come up with two approaches for converting the two class SVM classification to multi-class classification.

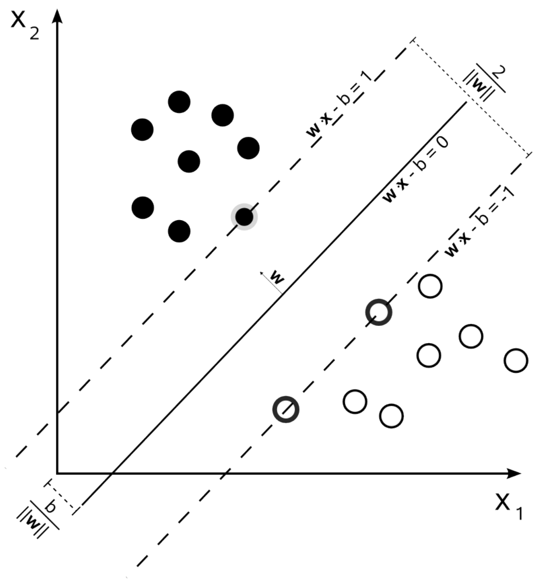


Figure 9: Support Vector Machine classification

Let w denote the vector orthogonal to decision boundary and b denote a scalar "offset" term, then we can write the decision boundary as,

$$w^T x + b = 0$$

The margin for the classification task can be defined as follows.

$$\begin{aligned} w^T x_i + b &> +c \qquad \text{for all } x_i \text{ in filled circle class} \\ w^T x_i + b &> -c \qquad \text{for all } x_i \text{ in open circle class} \end{aligned} \tag{9}$$

Jointly the Equation (1) can be written as,

$$(w^T x_i + b)y_i > c$$

Now, the margin in terms of the variables can be defined as,

$$\begin{aligned} m &= \frac{w^T}{||w||}(x_{i^*} - x_{j^*}) \\ &= \frac{2c}{||w||} \end{aligned} \tag{10}$$

Thus, our Maximum Margin classification problem for SVM is given by the following equation.

$$\begin{aligned} \max_{w,b} \quad & \frac{c}{||w||} \\ \text{s.t} \quad & y_i(w^T x_i + b) \geq c, \qquad \forall i \end{aligned} \tag{11}$$

We can eliminate c by fixing it value to 1, since it just scales the problem, to formulate a much cleaner problem, given by,

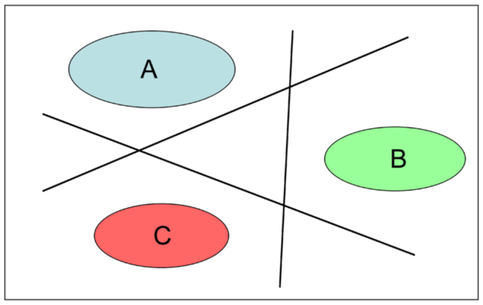
$$\begin{aligned} \max_{w,b} \quad & \frac{1}{||w||} \\ \text{s.t} \quad & y_i(w^T x_i + b) \geq 1, \qquad \forall i \end{aligned} \tag{12}$$

This is the Support Vector Machines formulation, which is believed to be the best “off-the-self” supervised learning algorithm.

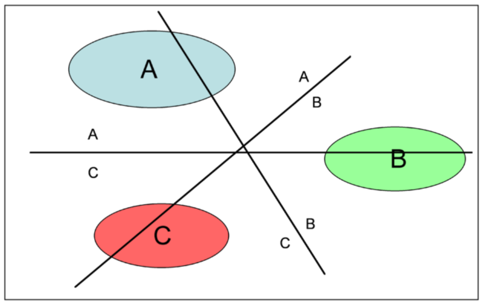
SVM Approaches for Multi-class Classification problem

There are mainly two approaches followed for multi-class classification, viz. one-vs-one approach and one-vs-rest (all) approach[17].

One-vs-one approach, as the name says, builds model by picking two class and then classifies for the given two first. Then, after ascertaining that data doesn’t lie in one class, it builds a classifier for the other versus another class of the classes which was left. The approach can be visualized as in Figure 10b. The other approach is one versus rest(all). In this approach, the the model is created for classifying one class versus all the other classes. Then among the other classes, one class is picked and the classifier is made for rest of the classes, thus classifying for each class separately. One vs rest approach can be visualized as in Figure 10a.



(a) One vs Rest approach



(b) One vs One approach

Figure 10: Different approaches in SVM Classification

In this way, number of classifiers required is reduced for one versus rest multi-class classification, as compared to one versus one multi-class classification.

6.4.3 Artificial Neural Network

Neural network is a semi-supervised approach which has proved to be of significant advantage in recent years in classification task. We have used a multi-layered perceptron (MLP) model to build the classifier for our task. MLP is a feed-forward network model which maps a set of input to an appropriate set of output [18]. The general architecture of a multi-layered perceptron can be seen in Figure 11.

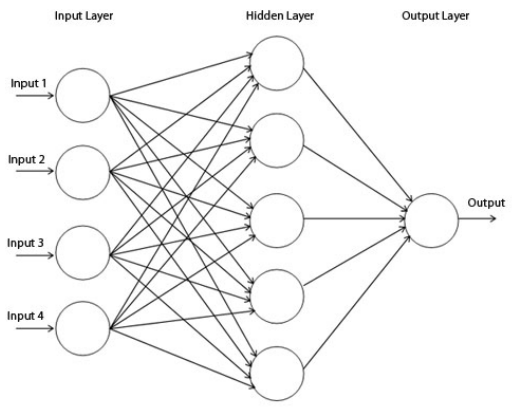


Figure 11: Multi-layered Perceptron Model

As we can see from the Figure 11, the multi-layered perceptron model has an input layer, at least one hidden layer and an output layer, which predicts the label of the input data. Each node of each layer except the input layer is a perceptron or neuron in itself, which is the basic unit of a neural network. The weights at each layer are learned through the back-propagation algorithm which updates weights based on the error obtained from the ground truth labeled data and prediction. Output of each layer goes through a non-linear activation function, which induces the non-linearity required to learn complex models given the data.

Perceptron: Basic unit of Artificial Neural Networks

The input to a perceptron such as the one given in Figure 12 is a d-dimensional vector. Next, we initialize the weights and the threshold for convergence of the model. The weights may be set to 0 or any small random values. Different random initialization of weights provide different starting point for the algorithm, which in turn help in better convergence of the model.

After initialization, we learn the model using a single perceptron by the following steps for each data point i in our training set D, on the input x_i and desired output $\hat{y_i}$.

1. Compute the output,

$$\begin{aligned}\hat{y_i} &= g_P[w^T x_i] \\ &= g_P[w_0 x_i^{(0)} + w_1 x_i^{(1)} + w_2 x_i^{(2)} + ... + w_d x_i^{(d)}]\end{aligned} \tag{13}$$

   where, $x_i^{(0)} = 1$ and g_P is the activation function for the perceptron. There are various types of activation functions used in the literature. Some of them are, 'Sigmoid', 'tanh', 'ReLU' etc. (Figure 13). We have used the 'tanh' activation function in our experiments.

2. Estimate the error of the network $e(w(t)) = y_i - \hat{y_i}$.

3. Update the weights

$$w(t+1) = w(t) + e(w(t))x_i$$

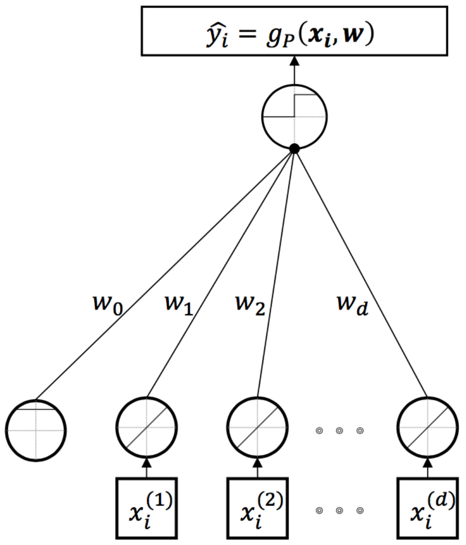
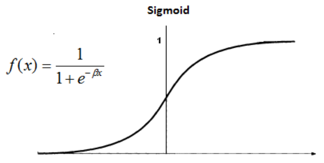
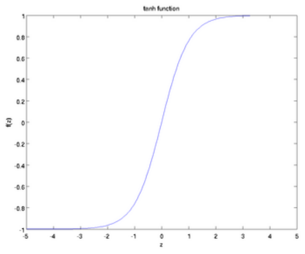


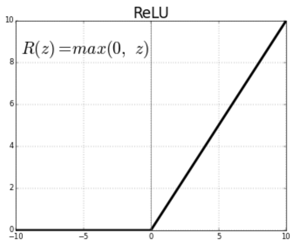
Figure 12: A Single Perceptron



(a) Sigmoid Function



(b) Tanh Function



(c) ReLU Function

Figure 13: Non-linear Activation Functions [19]

Back Propagation Algorithm and Gradient Descent

```
Data: $X_i := x_i^1, x_i^2, ...x_i^d, Y_i;$    $\forall i \in (1, m); W_j(t)$    $\forall j \in (0, d)$
Result: updated $W_j(t+1)$
initialization $W(0)$;
while convergence do
    for each training example $X_i$ do
        prediction $\hat{y}_i = g_P[w^T x_i]$;
        compute error $e(w(t)) = y_i - \hat{y}_i$ at the output units;
        compute $\Delta w_h$ for all weights from hidden layer to output layer;
        compute $\Delta w_i$ for all weights from input layer to hidden layer;
        update network weights $W_j(t+1)$ calculated through gradient descent[20] to find a local
          minimum;
    end
end
```

Algorithm 1: Back Propagation Algorithm

6.4.4 Logistic Regression

The logistic regression classifier uses the logistic (sigmoid) function to build a model for classification. This is a probabilistic approach which computes the probability of a data point lying in a given class, $p(y|x)$. The conditional distribution which is a Bernoulli is given by,

$$p(y|x) = \mu(x)^y (1 - \mu(x))^{(1-y)} \tag{14}$$

where μ is the logistic function and is given by,

$$\mu = \frac{1}{1 + e^{-w^T x}} \tag{15}$$

as shown in Figure 14

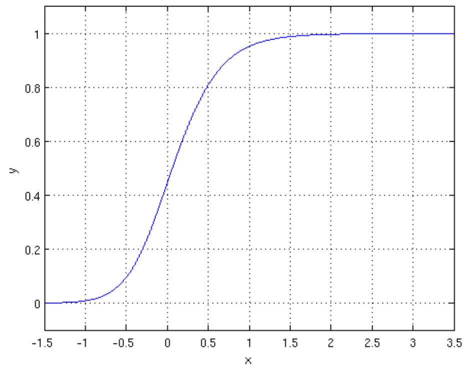


Figure 14: Logistic Function

The training of a logistic regression classifier involves estimating the parameters w which will maximize the conditional likelihood of the training data.

Let the training data be given by $D = (x_1, y_1), ..., (x_N, y_N)$. Then the data likelihood is given by,

$$\Pi_{i=1}^{N} P(x_i, y_i; w)$$

and the conditional likelihood will be given by,

$$\Pi_{i=1}^{N} P(x_i|y_i; w)$$

Thus, the final objective function becomes,

$$\begin{aligned} l(w) &= \text{argmax}_w \; ln(\Pi_{i=1}^{N} P(x_i|y_i; w)) \\ &= \text{argmax}_w \sum_{i=1}^{N} ln(P(x_i|y_i; w)) \end{aligned} \tag{16}$$

Substituting equation 14 and 15 in 16, we get the final objective function as,

$$l(w) = \text{argmax}_w \sum_{i=1}^{N} ln(P(x_i|y_i; w)) = \sum_{i=1}^{N} (y_i - 1) w^T x_i - \sum_{i=1}^{N} ln(1 + e^{-w^T x_i}) \tag{17}$$

Thus, to train the model we estimate the weights by optimizing the given objective function using the gradient ascent algorithm which is given as follows.

```
**Data:** $\{(X^i = \{x_1^i, x_2^i, ..., x_d^i)\}, Y^i\}_{i=1}^n$
**Result:** $W^{opt} = \{w_0, w_1, ..., w_{d+1}\}$
initialization $W^0$ = random($w_0, w_1, ..., w_{d+1}$), $\eta$;
while until convergence do
    for i := 1...n do
        $y_i := \frac{1}{1+exp(-(W^T X_i))}$;
        $W^{'} := W + \eta \sum_i (Y^i - y_i) X_i$;
    end
end
```

Algorithm 2: Logistic Regression Training Algorithm

6.4.5 Decision Tree

Decision tree is a non-parametric supervised learning method. It learns a function in the form of a decision tree, which approximates the discrete value targets. Basically, decision tree tries to infer the labels based on different criterion, and its values. The different combination of values for different criteria differentiates one class from the other.

For our classification tasks, decision tree creates a model that predicts the target label using a set of simple decision rule. Decision trees use a simple if-then-else construct on the decision rules to predict the outcome of a particular instance. For example, Figure 15, tries to predict the outcome for the concept of "*PlayTennis*"[21]. The outcome categorized as 'Yes', 'No', is based on three decision rules, viz. 'Outlook', 'Humidity' and 'Wind'. Based on decisions made by each level, the outcome may be generated directly, as for 'Outlook' criteria as 'Overcast', in which case the outcome or label was 'Yes', or, may take another decision rule to decide the outcome. The tree grows exponentially with number of decision rules, and generally these are not able to generalize over complex data well.

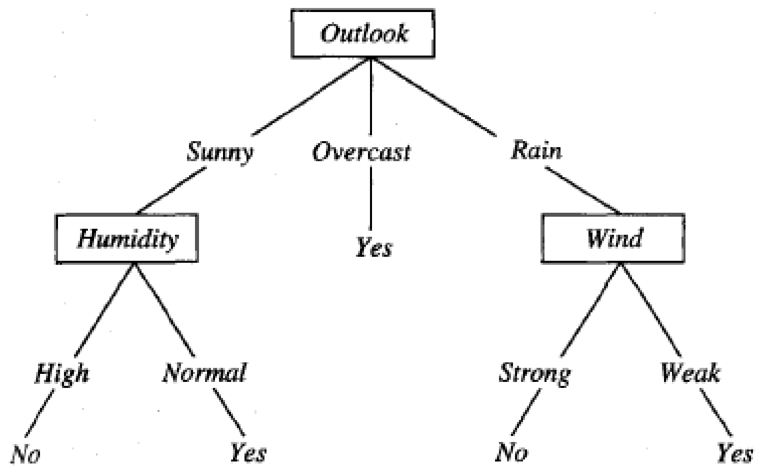


Figure 15: Decision Tree Classification

6.4.6 Naive Bayes Classifier

Naive Bayes is another probabilistic approach in we use Bayes rule, as in Bayes classifier, for computation of posterior probability $P(Y|X)$ to classify data based on the input X to one of the classes given by Y [22]. The difference between Bayes and Naive Bayes is the assumption of **conditional independence** in Naive Bayes which states that, X is conditionally independent of Y given Z, if the probability distribution governing X is independent of the value of Y, given the value of Z. Mathematically, conditional independence is expressed as given in Equation 18

$$P(X = i|Y = j, Z = k) = P(X = i|Z = k) \qquad \forall(i, j, k) \tag{18}$$

Thus, in Naive Bayes classification, the interpretation of conditional independence assumption is that each feature becomes independent of each other given a class. Thus, the joint likelihood of a feature given a class becomes,

$$P(X_1, ..., X_2|Y) = \Pi_i P(X_i|Y) \tag{19}$$

Thus the decision rule for Naive Bayes becomes,

$$\begin{aligned} y^* &= \text{argmax}_y P(y)P(x_1, ..., x_n|y) \\ &= \text{argmax}_y P(y)\Pi_i P(x_i|y) \end{aligned} \tag{20}$$

The training algorithm for Naive Bayes is same as Bayes classifier under the conditional independence assumption and is given in the following Algorithm.

```
Data: trainData$\{\{(X^i = \{x_1^i, x_2^i, ..., x_d^i)\}, Y^i\}_{i=1}^n\},\ x_{1..n}^i \in [1, K]$
Result: posLikelihood$_{Kxd}$, negLikelihood$_{Kxd}$, $posCount, negCount$
initialization posFreq$_{Kxd} := 1,$ negFreq$_{Kxd} := 1,$ posLikelihood$_{Kxd} := 1,$ negLikelihood$_{Kxd} :=$
  $1, posCount := 0, negCount := 0$;
for i := 1...n do
    if $Y^i == 1$ then
        for j := 1...d do
            posFreq($x_j^i, j$) := posFreq($x_j^i, j$) + 1;
        end
        posCount := posCount + 1;
    else
        for j := 1...d do
            negFreq($x_j^i, j$) := negFreq($x_j^i, j$) + 1;
        end
        negCount := negCount + 1;
    end
end
for i := 1...m do
    for j := 1...d do
        posLikelihood($x_j^i, j$) := posFreq($x_j^i, j$) / posCount;
        negLikelihood($x_j^i, j$) := negFreq($x_j^i, j$) / negCount;
    end
end
```

Algorithm 3: Naive Bayes Training Algorithm

6.4.7 Ensemble of all Classifiers

In the last method, we take the prediction of all approaches and use a voting method to predict the classes based on "popular" vote mechanism. The predictions for each of the method is generated and then for each data point in the test set, the predicted labels are counted. The label which gets the maximum vote by all the different classifiers is allocated to that data point.

6.5 Prediction of genre on test data set

The prediction on data set was done using 3-fold cross validation to tune for hyper-parameters as well as get the best accuracy out of the models. We also computed the confusion matrices for each of the classifiers along. Finally, the results of each classifier was verified by averaging over at least 5 iterations to get the final accuracy on the data set.

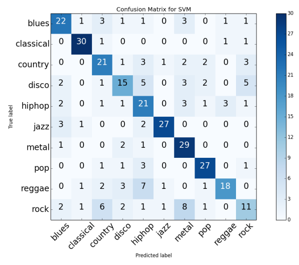
7 Experiments and Results

7.1 Model Classification For 10 Genres

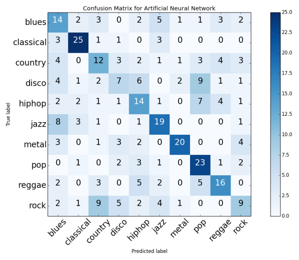
Referring to our dataset we applied our models to 10 genres classification namely blues, classical, country, disco, hip-hop, metal, jazz, pop, reggae and rock. From observation we found out that support vector machine outperformed every other classification model in our experiment with 66.48% accuracy. Accuracy for all models are shown in Table 1.

10 Genre Accuracy	
Models	Accuracy
Support Vector Machine	66.48%
Ensemble	62.48%
Logistic Regression	59.39%
Neural Network	47.81%
Decision Tree	42.60%
K-Nearest Neighbor	47.51%
Naive Bayes	47.14%

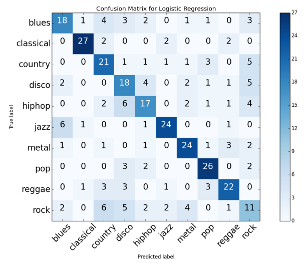
Table 1: 10 Genre Accuracy



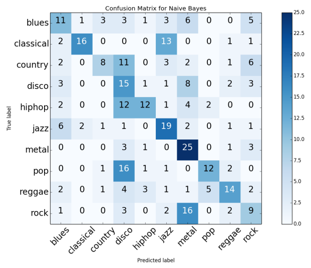
(a) Support Vector Machine



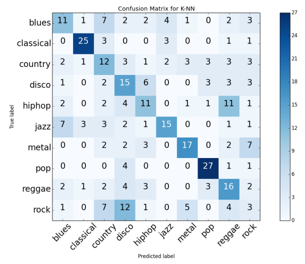
(b) Neural Network



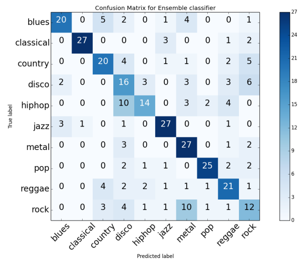
(c) Logistic Regression



(d) Naive Bayes



(e) K-Nearest Neighbor



(f) Ensemble

Figure 16: 10 Genre Classification

As from the confusion matrix it is pretty evident that classical, jazz, metal and pop genres are classified with best accuracy. So as suggested and experimented by various researchers we performed 4 genre classification experiment to observe the accuracy.

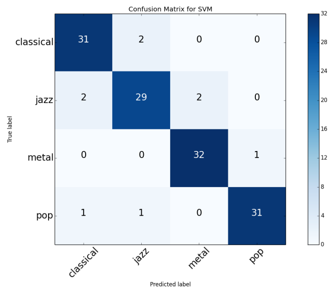
7.2 Model Classification For 4 Genres

The 4 genre classification classify audio in classical, jazz, metal and pop genres. The accuracy of this classification was very high and we achieved close to 90 percent accuracy in SVM. Other models also performed really well in 4 genre classification. The results for 4 genre classification are shown in Table 2.

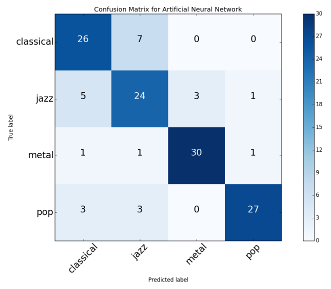
Above confusion matrix shows the accuracy of models on 4 genre classification.

4 Genre Accuracy	
Models	Accuracy
Support Vector Machines	90.44%
Logistic Regression	91.36%
K-Nearest Neighbor	81.96%
Decision Tree	81.96%
Neural Network	84.92%
Ensemble	91.66%
Naive Bayes	79.84%

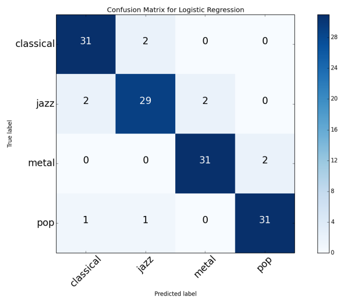
Table 2: 4 Genre Accuracy



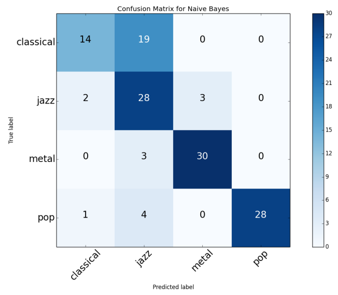
(a) Support Vector Machine



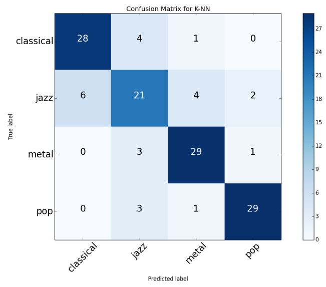
(b) Neural Network



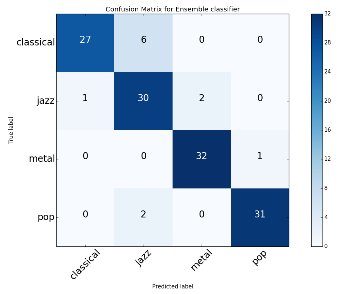
(c) Logistic Regression



(d) Naive Bayes



(e) K-Nearest Neighbor



(f) Ensemble

Figure 17: 4 Genre Classification

7.3 Dimensionality Reduction: Principal Component Analysis

Since the input dimensionality was very high and our goal was to create a model that can classify the genres in realtime, we experimented by applying dimensionality reduction for dimensionality curse and we used PCA for this task. As observed from accuracy vs number of dimension curve we found out that around 35 principal components would be optimum to balance accuracy vs performance for this task. This experiment was performed on 10-Genre classification and accuracy of model was taken average over 3 iterations and then plotted against number of principal components. Figure 18 shows the plot of accuracy vs number of principal components.

7.4 Feature Combinations

As we were studying various features and we found out that MFCC were dominant features so we decided to experiment our models for 4 genre classification with MFCC only features and without MFCC other features on same classification using our models. The results are shown in Table 3 and 4.

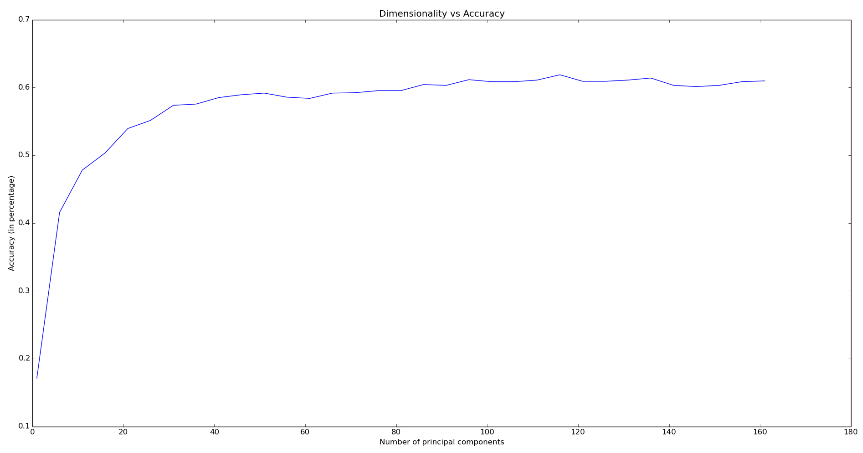


Figure 18: Principal Component Analysis

4 Genre MFCC	
Models	Accuracy
Support Vector Machine	88.18%
Ensemble	89.39%
Logistic Regression	87.42%
Neural Network	82.42%
Naive Bayes	76.36%
Decision Tree	76.66%
K-Nearest Neighbor	69.08%

Table 3: 4 Genre MFCC Accuracy

We observed that although MFCC is dominant feature but the best accuracy can be observed only using the combination of MFCC and other features as discussed earlier in our experiments.

7.5 Hyper Parameter Tuning

Hyper parameter tuning for KNN

We have experimented with different k as a hyper-parameter in our experiments. We decided to vary k in the range of [1, 10] and see the variation in results, and whether we can pick out the best value for k for our classification task. The results of hyper-parameter tuning for different k is shown in Table 5.

Hyper parameter tuning for SVMs

We have also experimented with different types of decision boundary. We can have either a linear decision boundary, for which we use the $linearsvc$ classifier of the scikit learn package. Or we can have a non-linear decision boundary, in which we can have multiple types of decision boundary like polynomial with varying degree and kernelized svm classification with various types of kernel. For our study, we have restricted to using linear decision boundary, polynomial decision boundary with degree 2 and 3 and kernelized decision boundary with 'RBF' kernel. We use the svc classifier to use the kernelized and polynomial models. The results of hyper-parameter tuning for different kernels and type of classifier used is shown in Table 6.

4 Genre Other Features	
Models	Accuracy
Support Vector Machine	79.39%
K-Nearest Neighbor	76.66%
Logistic Regression	78.78%
Neural Network	74.24%
Ensemble	81.36%
Naive Bayes	78.78%
Decision Tree	78.78%

Table 4: 4 Genre Other Features Accuracy

k	Accuracy
1	81.06%
2	80.60%
3	82.87%
4	82.27%
5	83.19%
6	81.97%
7	81.66%
8	80.90%
9	80.15%
10	80.29%

Table 5: k-NN Classifier parameter tuning

Kernel	Type	Accuracy
Linear	linearsvc	90.45%
Polynomial(d=2)	svc	88.86%
Polynomial(d=3)	svc	87.72%
RBF	svc	28.03%

Table 6: SVM Classifier parameter tuning

8 Conclusion

From our experiments, we analyzed the accuracies of different machine learning models - Support Vector Machine(SVM), Decision Tree, Artificial Neural Network, Logistic Regression, k-NN and Ensemble based on Majority Vote. Based on our study, we were able to conclude that SVMs performs better in general to other classifiers. Logistic Regression and Ensemble method also performed quite well. If performance is a criteria to select a model, ensemble methods should be avoided.

The best accuracy we achieved for 10-genre classification was 66.48% and the best accuracy we achieved for 4-genre classification was 91.66%.

We also tested many sound features both in time and frequency domain - MFCC, Spectral Centroid, Spectral Roll-Off, Onset Tempo, Zero crossing rate, short term energy and low energy. Out of the features we tested, we got best accuracy with MFCC.

The Gaussian assumption that we used to aggregate sequential data proved to be very effective for achieving good accuracy.

In order to achieve our goal to develop real-time genre classifier system, we reduced training time by reducing the dimensionality of our feature. This was achieved by applying PCA of the input feature. Based on our plot between the number of dimensions vs accuracy, we found out that the best balance between accuracy and performance is achieved at $d = 35$.

9 Future Work

- We tried to work on content based audio feature in our experiments. The features can be extended to lyrics in combinations with these features.
- Visual representations of frequency spectrum is also another addition that can be used.
- With the advent of deep learning, there are state of the art solutions to music genre classification including Convolutional Neural Network and Recurrent Neural Networks.
- Apart from the Gaussian assumption that we used, there are various techniques to aggregate the frame features. We can extend our work to explore other aggregation techniques.

Tools and Technologies

SkLearn[31] - We used the sklearn python package for various classification models like SVM, k-NN, Logistic Regression, Artificial Neural Network and getting accuracy and confusion matrix.

Librosa[2] - After researching various tools [25] to get features from raw audio files, we used Librosa python package for feature extraction. It uses ffmpeg plugin for reading audio files and has the functionality of analysing in both time and spectral domain.

T-SNE[8] - We used T-SNE plots in sklearn.manifold package to visualize the data in a 2-D plane for analysing the feature set corresponding to all the genres.

PyCharm[32] - We used pycharm as IDE for our code development.

Individual Contribution

Name	ASU ID	ASU E-Mail	%	Contribution
Arjun Jajoo	1211262049	adjajoo@asu.edu	25%	Data Collection. Feature Extraction, Logistic Regression, Evaluation, Presentation, Report
Garvit Khandelwal	1211261542	gkhandel@asu.edu	25%	Data Collection, Feature Extraction, Dimensionality reduction using PCA, KNN, Evaluation, Presentation, Report
Pradhuman Swami	1211012475	pswami1@asu.edu	25%	Gaussian Feature Extraction, Neural Network, Nave Bayes, Evaluation, Presentation, Report
Siddhant Prakash	1211092724	sidprakas9@asu.edu	25%	SVM, Decision Tree, Ensemble, Evaluation, Presentation, Report

Table 7: Individual Contribution of each team member

Acknowledgments

We thank Prof. Hanghang Tong for his guidance throughout this course and project work. The material covered in coursework helped us in performing experiments on various models that were discussed. We would also like to thank TA Boxin Du for his help during coursework.

References

[1] Tzanetakis, George, and Perry Cook. ”Musical genre classification of audio signals.”IEEE Transactions on speech and audio processing10.5 (2002): 293-302.

[2] McFee, Brian, et al. ”librosa: Audio and music signal analysis in python.”Proceedings of the 14th python in science conference. 2015.

[3] Haggblade, Michael, Yang Hong, and Kenny Kao. ”Music genre classification.” Department of Computer Science, Stanford University (2011).

[4] Camenzind and Goel 2013 Tom Camenzind and Shubham Goel. 2013. #jazz Automatic Music Genre Detection. (2013).http //cs229.stanford.edu/proj2013/CamenzindGoel-jazz_Automatic_Music_Genre_Detection.pdf

[5] Lee, Chang-Hsing, et al. ”Automatic music genre classification based on modulation spectral analysis of spectral and cepstral features.” IEEE Transactions on Multimedia 11.4 (2009): 670-682.

[6] Schindler, Alexander, and Andreas Rauber. ”An audio-visual approach to music genre classification through affective color features.” European Conference on Information Retrieval. Springer International Publishing, 2015.

[7] Howard, Sam, Carlos N. Silla Jr, and Colin G. Johnson. ”Automatic lyrics-based music genre classification in a multilingual setting.” Proceeedings of the Thirteenth Brazilian Symposium on Computer Music. Vol. 34. 2011.

[8] Maaten, Laurens van der, and Geoffrey Hinton. ”Visualizing data using t-SNE.” Journal of Machine Learning Research 9.Nov (2008): 2579-2605.

[9] ”Window function.” Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 27 April 2017.

[10] Kumar, D. Pradeep, B. J. Sowmya, and K. G. Srinivasa. ”A comparative study of classifiers for music genre classification based on feature extractors.” Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER), IEEE. IEEE, 2016.

[11] ”Support Vector Machine.” Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 27 April 2017.

[12] Abe, S. (2003) Analysis of Multiclass Support Vector Machines. International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA 2003), 385-396.

[13] Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):955-974.

[14] Kijsirikul, B. & Ussivakul, N. (2002) Multiclass support vector machines using adaptive directed acyclic graph. Proceedings of International Joint Conference on Neural Networks (IJCNN 2002), 980-985.

[15] Vapnik, V. (1995) The Nature of Statistical Learning Theory. Springer-Verlag, London.

[16] Vapnik, V. (1998). Statistical Learning Theory. Wiley-Interscience, NY.

[17] Aisen B., ”A Comparison of Multiclass SVM Methods.”, December 15, 2006. http://courses.media.mit.edu/2006fall/mas622j/Projects/aisen-project/

[18] ”Multi-Layer Perceptron.” Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 27 April 2017.

[19] ”Activation function.” Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 27 April 2017.

[20] ”Gradient descent.” Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 27 April 2017.

[21] Michalski, Ryszard S., Jaime G. Carbonell, and Tom M. Mitchell, eds. Machine learning: An artificial intelligence approach. Springer Science & Business Media, 2013.

[22] ”Naive Bayes classifier.” Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 27 April 2017.

[23] ”Short-time Fourier transform.” Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 27 April 2017.

[24] ”Mel Frequency Cepstral Coefficient (MFCC) tutorial” http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

[25] Moffat, David, David Ronan, and Joshua D. Reiss. ”An evaluation of audio feature extraction toolboxes.” Proceedings of the 18th International Conference on Digital Audio Effects (DAFx-15), Trondheim, Norway. 2015.

[26] ”Pandora Internet Radio.” Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 27 April 2017.

[27] ”iTunes.” Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 27 April 2017.

[28] ”Shazam.” Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 27 April 2017.

[29] ”Fast Fourier Transform.” Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 27 April 2017.

[30] ”Spectral Centroid” https://ccrma.stanford.edu/ unjung/AIR/areaExam.pdf

[31] Pedregosa, Fabian, et al. ”Scikit-learn: Machine learning in Python.” Journal of Machine Learning Research 12.Oct (2011): 2825-2830.

[32] PyCharm https://blog.jetbrains.com/pycharm/