Music Genre Classification

ARITIFICAL INTELLIGENCE PROJECT

Under the supervision of
Mr. Rishabh Kaushal
Assistant Professor
Department of IT

Submitted by:
TEAM 5
Ayushi Gupta - 03501032017
Prarthana Kandwal - 01701032017
Ragini Sharma - 01101032017
Shivalika Goel - 01001032017
Subasree Arvind - 06701032017
Surmayi Sharma - 00401032017

Department of Information Technology
Indira Gandhi Delhi Technical University for Women
Kashmere Gate, Delhi - 110006

May 2020

Contents

ABSTRACT : With the increasing popularity of music streaming platforms, automatic music genre classification has generated considerable interest in the field of artificial intelligence. Genre classification categorizes music, making it easier for the users to access music, as well as the marketing companies. In this work, we analyse the GTZAN dataset that consists of 1000 songs, 10 genres each having 100 tracks, to extract patterns and features to identify genres accurately. We accomplish this by applying varied classical machine learning algorithms from the simple KNN algorithm to more complicated ones like neural network perusing over which works best in predicting the output genre aptly. Different features were found to have different effects on the outcome.

1

Introduction

There has been a shift in the Music industry towards digital distribution through streaming applications like Spotify, Soundcloud, Wynk etc. The music available in these application is increasing rapidly. At present, Spotify lists more than 50 million songs. These application try to aim at classifying music on the basis customer recommendation to attract more users. Music Classification is also done based on the genres. To classify music in their respective genres,machine learning algorithms can be used.

In this work, we implemented machine learning algorithms like SVM, Neural Networks, Naive Bayes, Logistic Regression and KNN on 1000 audio files and classified them in one of the 10 labels.

The practical relevance of the music genre classification is to track down societies with specific interests and discover related songs. This feature will contribute in the improvement of generating playlist and provide better music recommendations to the users.

1.1 Problem Statement

The aim of the project is to use machine learning algorithms that would take music files(.mp3,.au) as input and provide the genres as output. Some of the Music Genres that we have taken into consideration are Rock, Pop, Classical, Hip Hop, Disco, Country, Reggae, Metal, Jazz and Blues.

2

Literature Survey

2.1 Research questions

(1) How accurately can the input file be classified into the correct genre?

(2) What features can be extracted from the audio files?

(3) How to visualize and prepare the extracted features?

(4) Which combination of features give the highest accuracy?

(5) What techniques can be applied to improve accuracy?

(6) How can this classifier be used to improve recommendations in music streaming applications?

3

Related Work

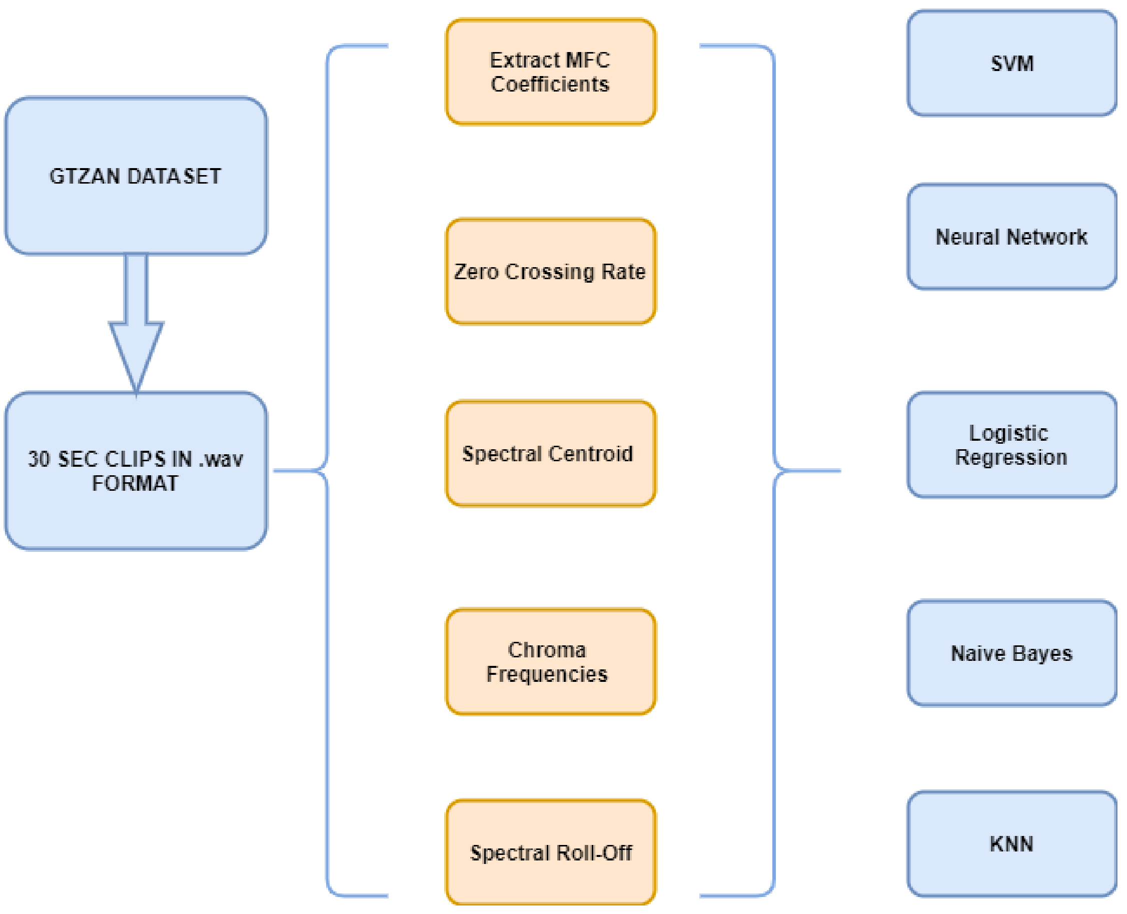
Automatic music genre classification has been a popular topic of research in past years. Several attempts have been made to generate an effective machine learning model to classify music into different genres.

According to [1] and [2], it was observed that an in depth analysis was made on the data and the features extracted from it. The resultant dataset was made to pass through various classifiers such SVM, Regression, Neural Networks and KNN. A maximum accuracy of 68.07% was achieved by [1] using Non-linear SVM(RBF Kernel). As per [2], the classifier that gave the highest accuracy of 84 was SVM(Polynomial Kernel).

Our main aim with this project was to come up with a model that improves upon the accuracies of the [1] and [2]. After a rigorous analysis, we inferred that more emphasis was given to the dataset and its features, but there was a scope to explore the classifiers in a more comprehensive manner. Also, there were some classifiers which were not analysed and could have given potentially better results. Moreover, the classifiers which had been implemented could have performed better with the use of techniques such as parameter fine tuning and feature engineering.

So, the focus with this project was to deeply understand the dataset and the classifiers to implement a high accuracy model using the above mentioned techniques.

Work Flow


GTZAN DATASET
30 SEC CLIPS IN .wav FORMAT
Extract MFC Coefficients
Zero Crossing Rate
Spectral Centroid
Chroma Frequencies
Spectral Roll-Off
SVM
Neural Network
Logistic Regression
Naive Bayes
KNN

Step 1: The raw dataset was converted into a suitable format(.wav files)

Step 2: The following features were extracted from the audio files:

- MFCC
- Zero Crossing Rate
- Spectral Centroid
- Chroma Frequencies
- Spectral Roll Off

Step 3: The following classification algorithms were applied:

- SVM
- Neural Network
- Logistic Regression
- Naive Bayes
- KNN

Step 4: Parameter tuning was performed on those on algorithms which gave a good accuracy, while feature engineering was done for the algorithms with low accuracy.

Step 5: The highest accuracy model was chosen.

5

Data Preparation

5.1 Dataset

For our initial analysis, we used the GTZAN dataset. This dataset is available on the MARYSAS website (http://marsyas.info/downloads/datasets.html), and was used in “G. Tzanetakis and P. Cook. Musical genre classification of audio signals IEEE Transactions on Speech and Audio Processing”.

The dataset contains audio samples separated by genres, with 100 audio clips per genre in .au format. These genres are - classical, disco, hip-hop, jazz, metal, pop, reggae, rock, blues and country. Most of the music records available can be classified as one of these genres and thus, this dataset serves as an ideal input for the analysis.

The length of the audio clips is 30 seconds, and are 22050Hz Mono 16-bit files. The samples are incorporated from an assortment of sources such as radios, microphone recordings etc. to ensure that the results aren’t biased towards any specific source. The dataset will be split in a ratio of 0.8 : 0.2 (training : testing).

Number of Instances = 1,000
Number of Attributes = 5
Label Information: Label Present

5.2 Data Production

The classification cannot directly work on raw data. It needs to be transformed into more meaningful representations. We convert the samples from .au to .wav format, so that we can use python's wave module to read the audio files.

We have majorly used two libraries to perform audio processing and playback:

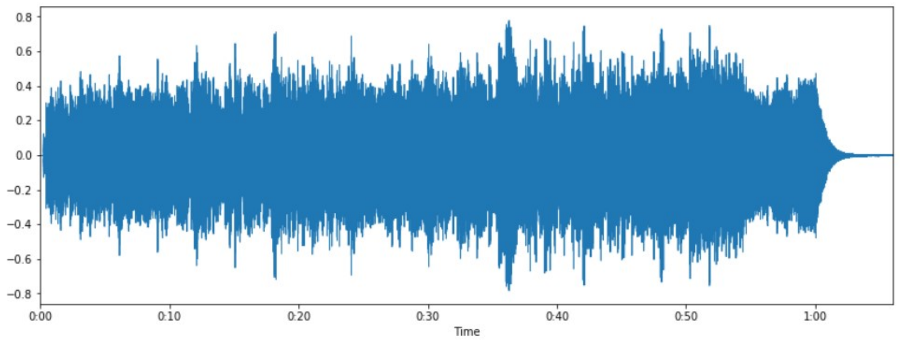
1. Librosa
Audio signals are analyzed using the Librosa Python module. It is more inclined towards music analysis and is an ideal choice for this project. It effectively builds a Music Information Retrieval (MIR) system.

2. IPython.display.Audio
Audio can be played directly in a Jupyter notebook using this library.

5.3 Data Visualisation

5.3.1 Waveform

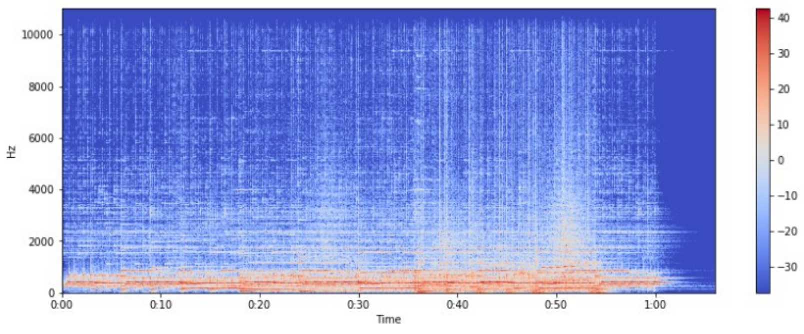
Waveform is the graphical representation of an audio sample, as a function of time. The plot can be generated using librosa.display.waveplot. Given below is the visual representation of the amplitude envelope of a waveform.



Scale: X-axis:1cm=10sec Y-axis:1cm=0.2m

5.3.2 Spectrogram

A spectrogram is a visual portrayal of the range of frequencies of different signals as they change with time. Spectrograms are also referred to as voicegrams, sonography or voiceprints. 3D representations of this data are known as waterfalls. In the below displayed plot, the horizontal axis is time while the vertical axis is frequency. The spectrogram is displayed using Librosa.display.specshow.



5.4 Feature Extraction

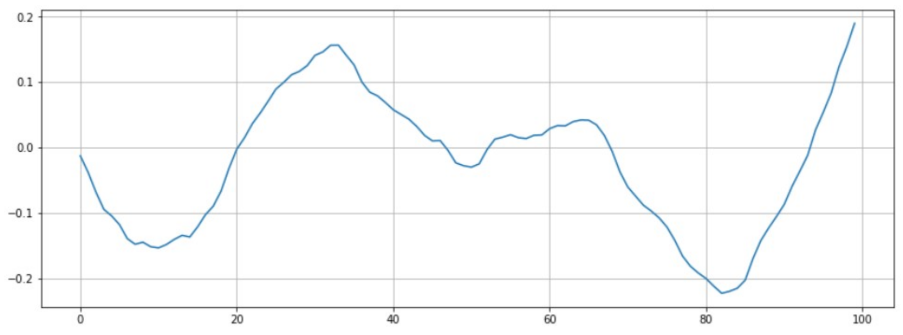
Multiple features can be used to describe a single audio signal. However, not every feature is relevant for music genre classification and we needed to identify the ones that are. Feature extraction is the process by which we extract features from the samples and use them for analysis.
After sufficient research, we found that the following features can be used to give accurate results:

5.4.1 Zero Crossing Rate

As is visible in the waveform, the signal changes its sign multiple times. The rate of change of signal from positive to negative and back is known as zero crossing rate.

This feature has been formerly used in music information retrieval and speech recognition. It can also be used to distinctly identify music genres. For example, it has higher values when the audio is highly percussive like in metal and rock.

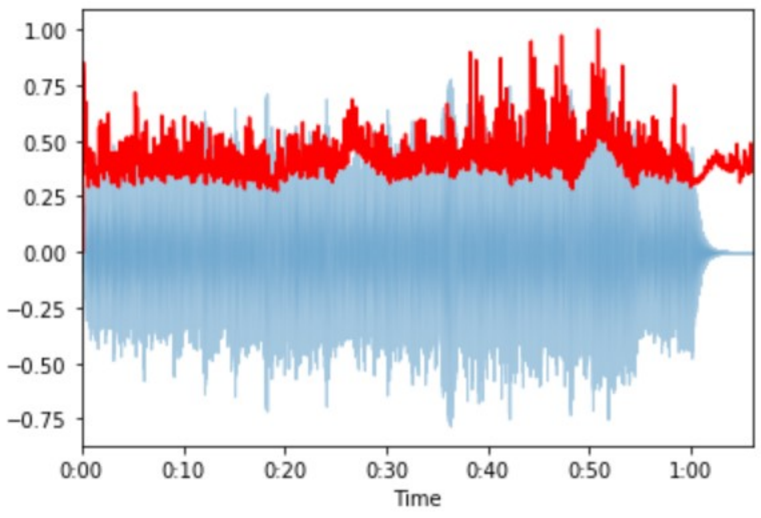
In the sample audio clip shown below, the zero crossing rate would be 5. The x-axis shows increasing time and the y-axis represents amplitude.



Scale: X-axis: 1cm = 12sec Y-axis: 1cm = 0.1m

5.4.2 Spectral Centroid

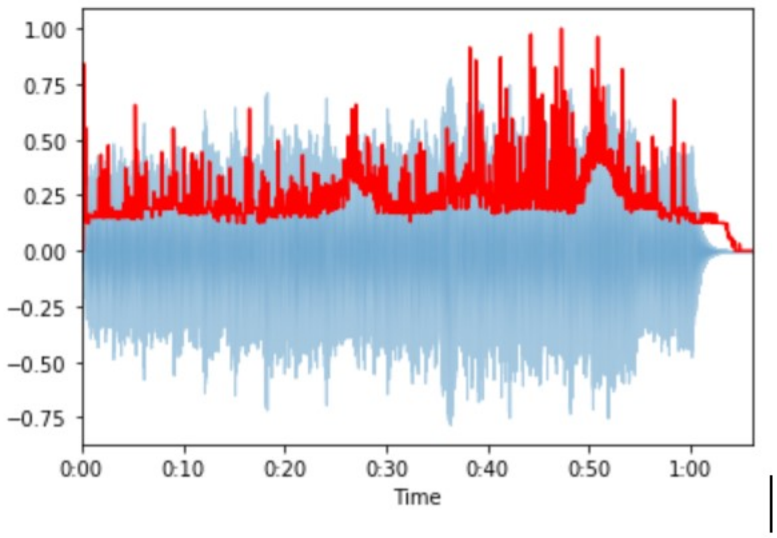
It is the weighted mean of the frequencies of an audio signal, and basically represents where its centre of mass is located. For example we consider two genres, blues and metal. A song from the metal genres has more frequencies towards the end (spectral centroid towards the end), while a blues song is the same for its entire length (spectral centroid near the middle). Thus, we have used this feature to differentiate genres.
The spectral centroid for each frame in a signal is computed using the librosa.feature.spectral_centroid. function.
In the below plot, we observe that the frequencies have been uniformly distributed throughout the duration with a slight spike towards the end. This causes the spectral centroid to be in the center skewed a little towards the end, signifying classical or blues music. The input sample belonged to the classical genre thus, justifying the validity of this feature.



Scale: X-axis: 1cm = 10sec Y-axis: 1cm = 0.25Hz (Normalized frequency)

5.4.3 Spectral Rolloff

This feature depicts the frequency below which a fixed fraction (generally 0.85) of the cumulative spectral energy resides. It distinguishes voiced speech from unvoiced speech. The spectral rolloff is calculated using the function librosa.feature.spectral_rolloff.



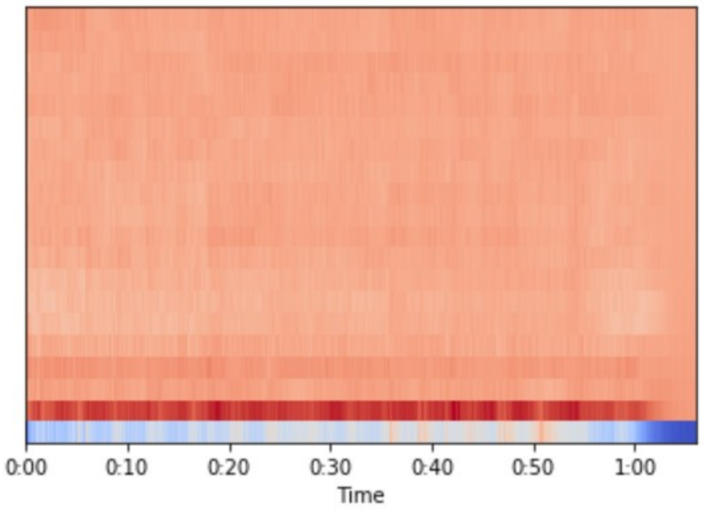
Scale: X-axis: 1cm = 10sec; Y-axis: 1cm = 0.25Hz (Normalized frequency)

5.4.4 Mel-Frequency Cepstral Coefficients

The MFCCs of a signal are a collection of features (usually about 10–20) which describe the spectral characteristics of the signal according to the Mel-frequency scaling.

The Mel-scale models human auditory response. Pitch is not perceived in a linear manner by the human auditory system, and the Mel-scale performs the mapping between actual frequencies and perceived pitch. This enables our models to analyse the data from a more human perspective.

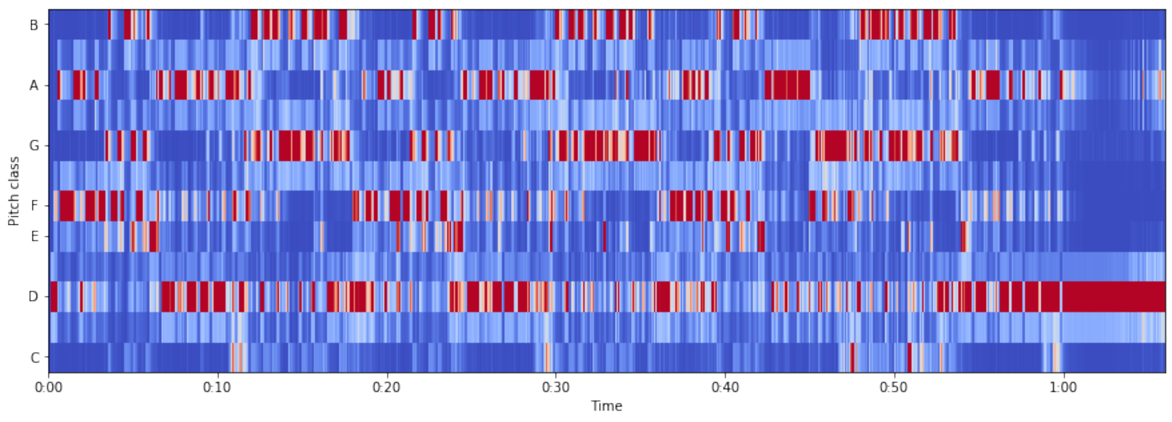
MFCCs are computed using librosa.feature.mfcc.



5.4.5 Chroma Frequencies

Each note in a music sample belongs to one of twelve pitch classes. Chroma features describe audio samples by projecting the entire spectrum onto 12 bins, each representing a distinct pitch class.

The twelve classes are (C,C#,D,D#,E,F,F#,G,G#,A,A#,B).
Chroma frequencies can be computed using librosa.feature.chroma_stft.



5.5 Understanding Data

To understand our dataset and its features we visualised it using graphical representations.

- The column of filename was dropped using .drop() function as it was a futile feature.

- The data was wrangled for any null values using functions .info() , .isnull() and .isnull().sum(). A heatmap was created to represent the same.

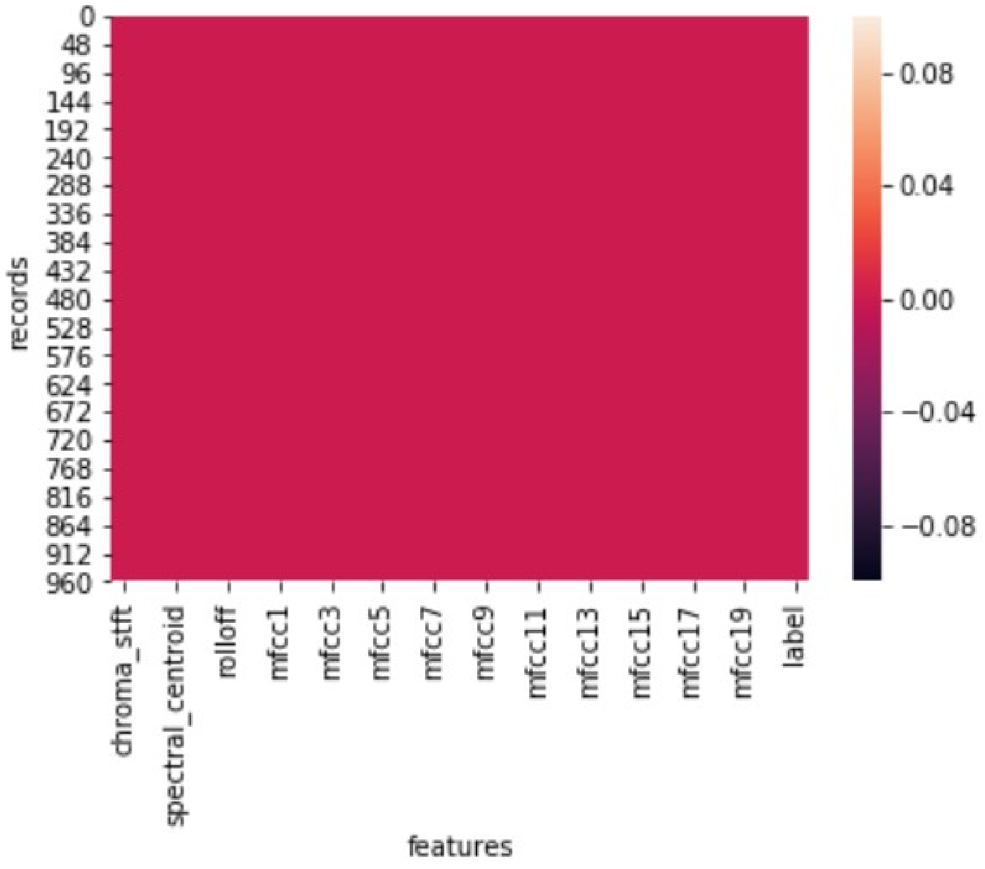


Fig:Heatmap

The whole heatmap is of uniform color, crimson, which is the indicator of 0. Hence, this exhibits that there are no null values in our dataset.

- Histograms of individual features were constructed to realize the range of their majority values.The x-axis denotes the feature and the y-axis denotes the frequency of the values. Below are the histograms.

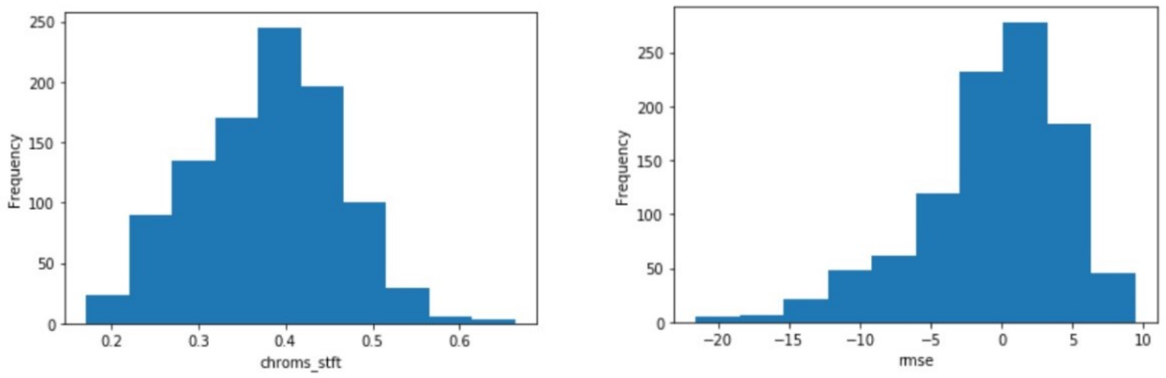


Fig1: Histograms of features chroma_stft and rmse

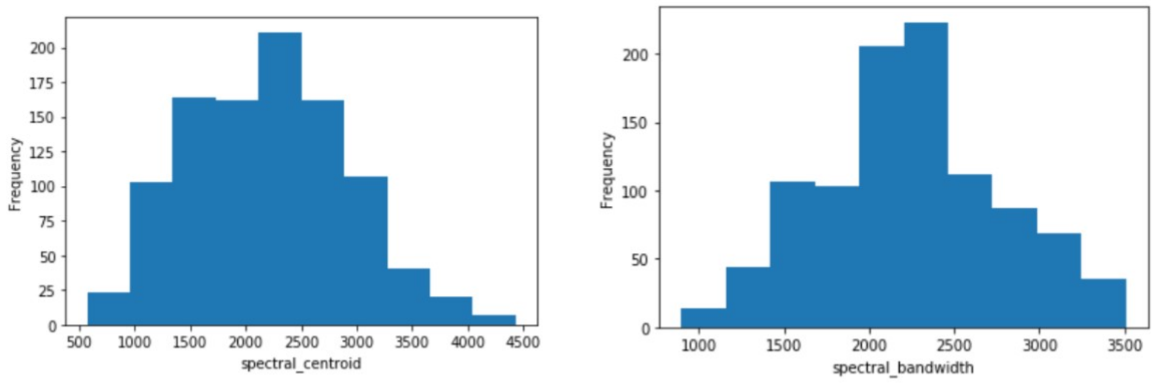


Fig2: Histograms of features spectral_centroid and spectral_bandwidth

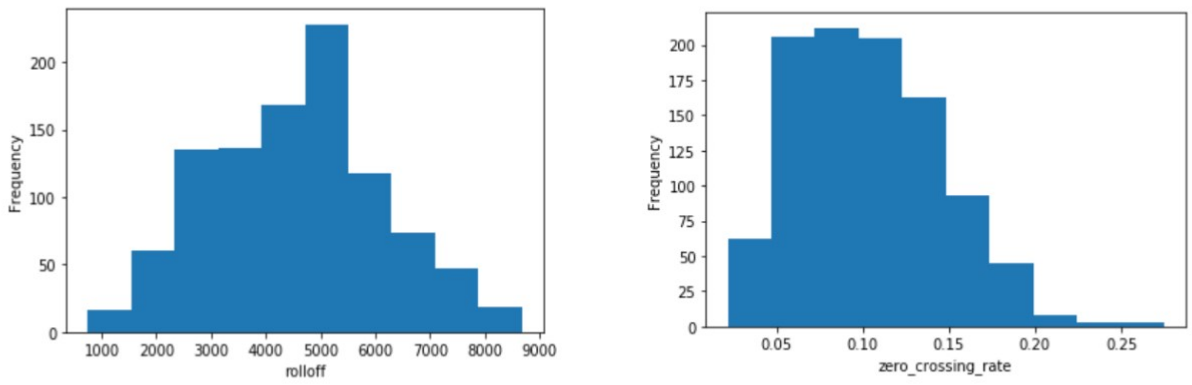


Fig3: Histograms of features rolloff and zero_crossing_rate

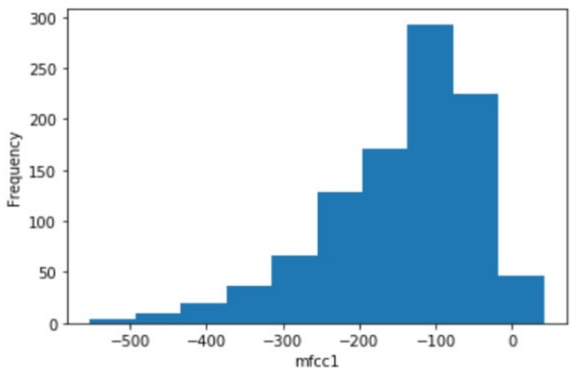


Fig4: Histogram of feature mfcc1

- Lastly, we modeled boxplots of each feature with labels to comprehend the distribution of a feature based on different genres.The x-axis represents the labels and the y-axis represents different features.Below are the boxplots.

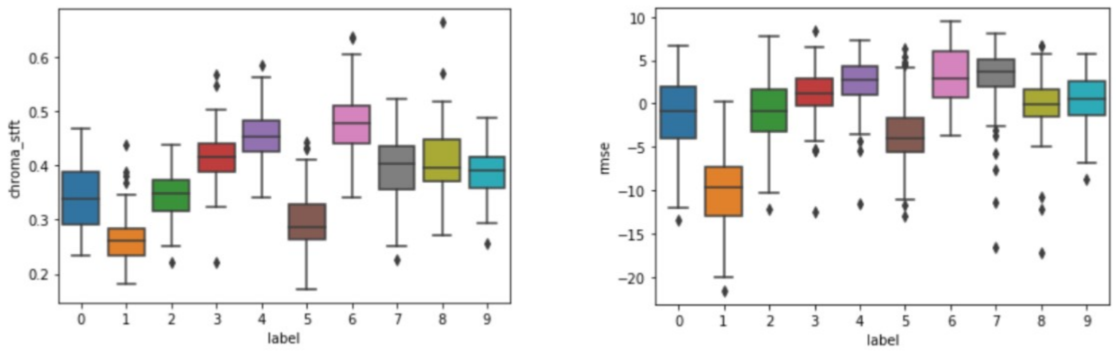


Fig1: Boxplots of features chroma_stft and rmse with the labels

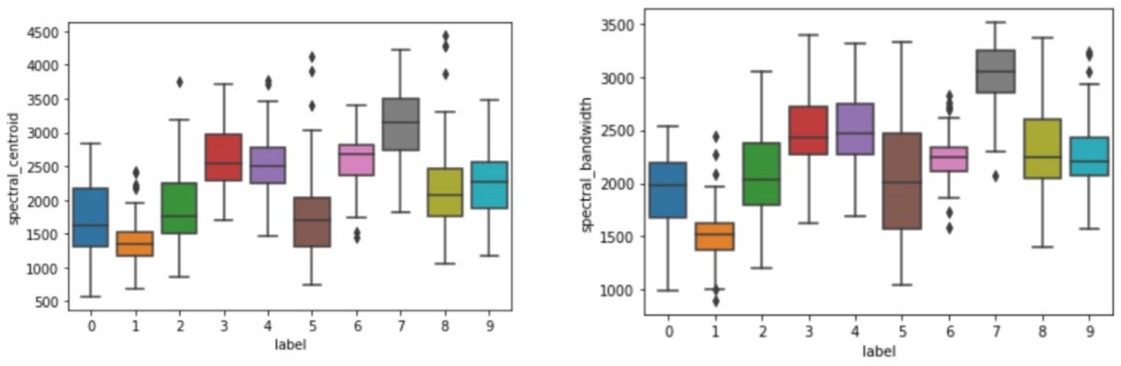
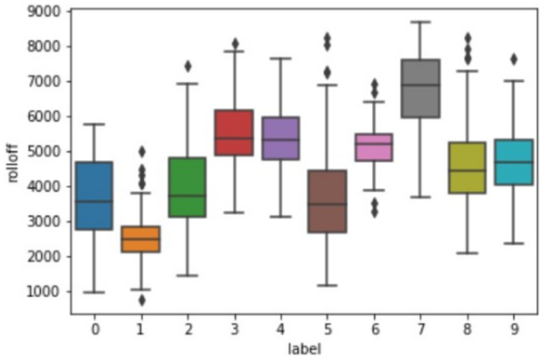


Fig2: Boxplots of features spectral_centroid and spectral_bandwidth with the labels



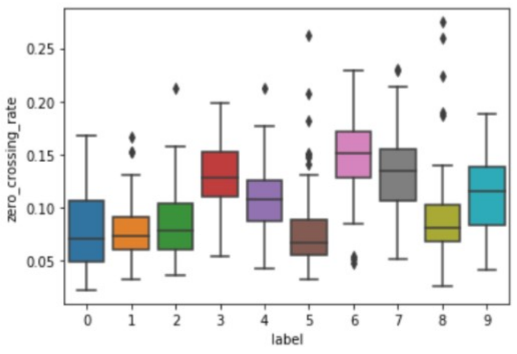


Fig3: Boxplots of features rolloff and zero_crossing_rate with the labels

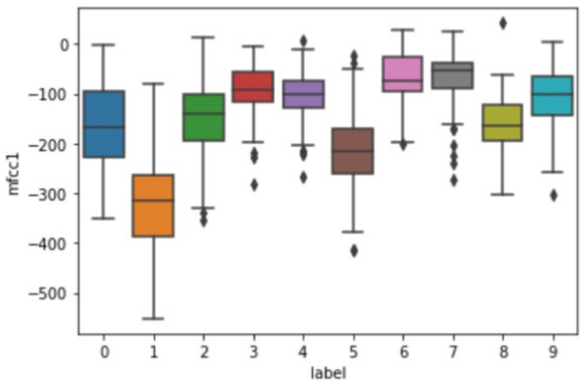
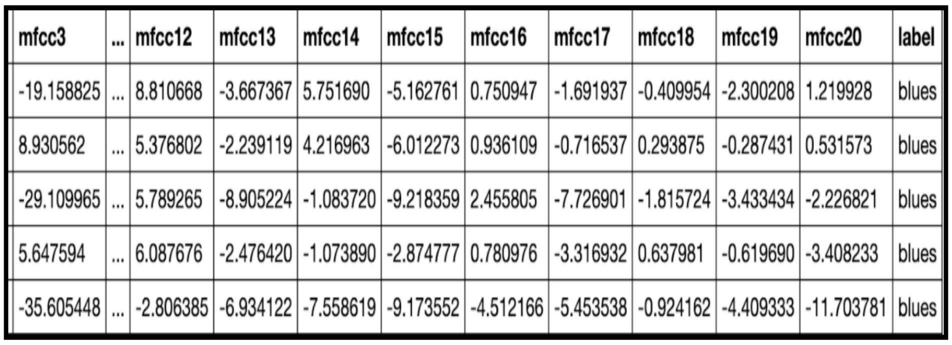


Fig4: Boxplot of feature mfcc1 with the labels

5.6 Data Preprocessing

- After we extracted the features from the dataset, the next step was to preprocess the data.
- In data preprocessing, the aim was to convert the raw data set into a format which was understandable to the classifiers.

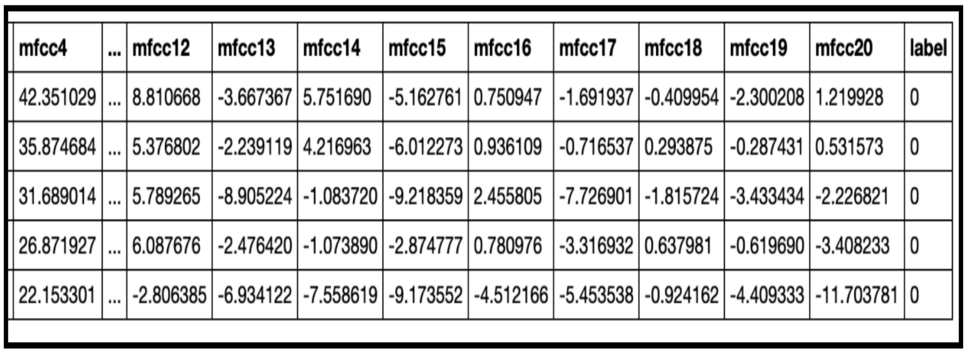
DATA BEFORE PREPROCESSING



mfcc3	...	mfcc12	mfcc13	mfcc14	mfcc15	mfcc16	mfcc17	mfcc18	mfcc19	mfcc20	label
-19.158825	...	8.810668	-3.667367	5.751690	-5.162761	0.750947	-1.691937	-0.409954	-2.300208	1.219928	blues
8.930562	...	5.376802	-2.239119	4.216963	-6.012273	0.936109	-0.716537	0.293875	-0.287431	0.531573	blues
-29.109965	...	5.789265	-8.905224	-1.083720	-9.218359	2.455805	-7.726901	-1.815724	-3.433434	-2.226821	blues
5.647594	...	6.087676	-2.476420	-1.073890	-2.874777	0.780976	-3.316932	0.637981	-0.619690	-3.408233	blues
-35.605448	...	-2.806385	-6.934122	-7.558619	-9.173552	-4.512166	-5.453538	-0.924162	-4.409333	-11.703781	blues

- Following steps were taken to perform preprocessing:
  1. All the names of the labels were encoded and a unique numeric value was associated with each of them.
  2. Unnecessary columns of data such as the name of the audio files were removed.
  3. The data was splitted into training and testing data.

DATA AFTER PREPROCESSING

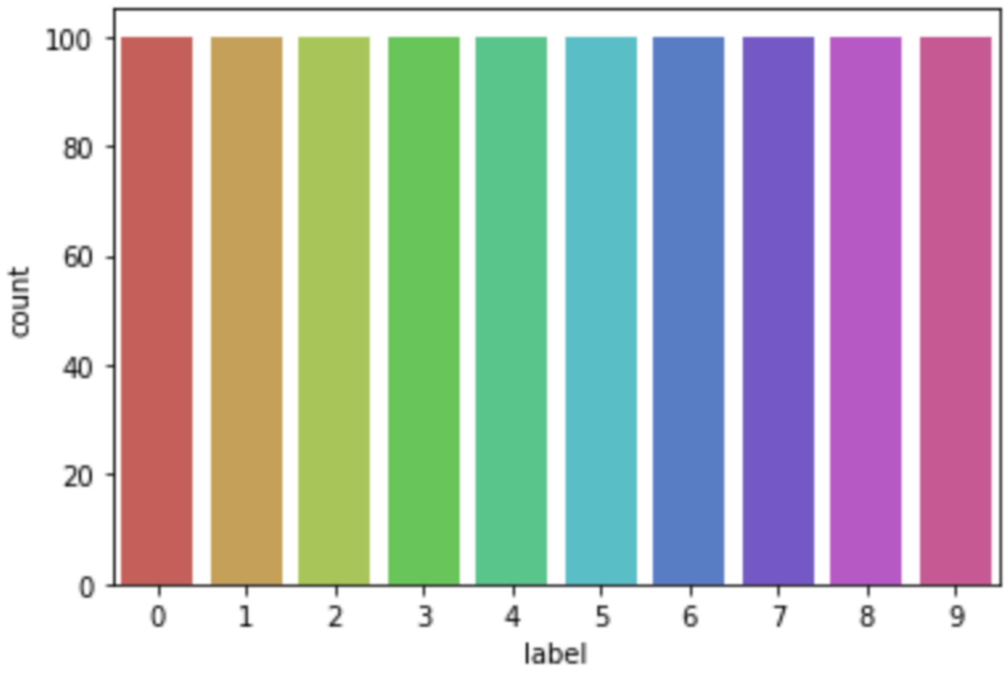


mfcc4	...	mfcc12	mfcc13	mfcc14	mfcc15	mfcc16	mfcc17	mfcc18	mfcc19	mfcc20	label
42.351029	...	8.810668	-3.667367	5.751690	-5.162761	0.750947	-1.691937	-0.409954	-2.300208	1.219928	0
35.874684	...	5.376802	-2.239119	4.216963	-6.012273	0.936109	-0.716537	0.293875	-0.287431	0.531573	0
31.689014	...	5.789265	-8.905224	-1.083720	-9.218359	2.455805	-7.726901	-1.815724	-3.433434	-2.226821	0
26.871927	...	6.087676	-2.476420	-1.073890	-2.874777	0.780976	-3.316932	0.637981	-0.619690	-3.408233	0
22.153301	...	-2.806385	-6.934122	-7.558619	-9.173552	-4.512166	-5.453538	-0.924162	-4.409333	-11.703781	0

NUMERIC LABELS

CLASS LABEL	ENCODED VALUE (DATA)	ENCODED VALUE(CONFUSION MATRIX)
Hiphop	0	A
Disco	1	B
Rock	2	C
Classical	3	D
Country	4	E
Jazz	5	F
Reggae	6	G
Metal	7	H
Blues	8	I
Pop	9	J

GRAPHICAL REPRESENTATION



- The data set was ready to be used by the classifiers.

  INPUT: Testing data

  OUTPUT: Above mentioned Labels

6

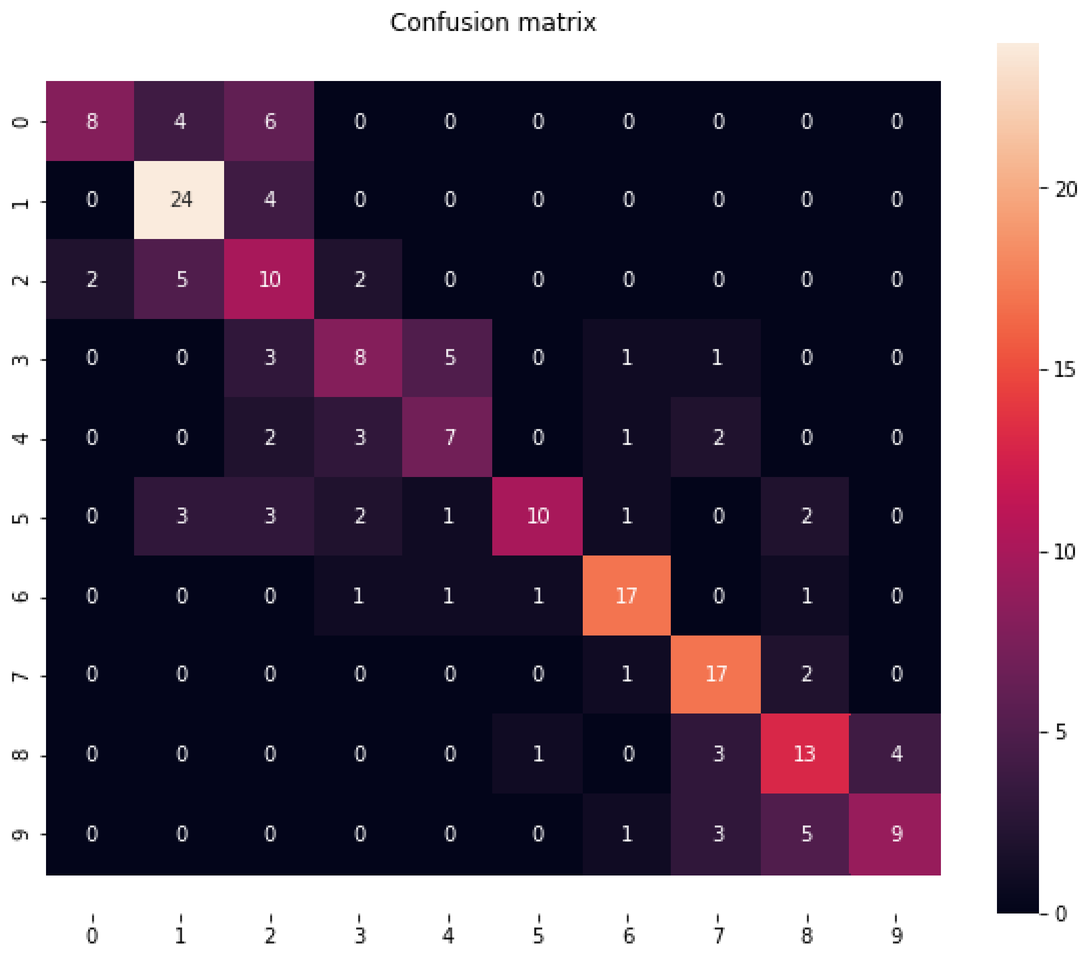
Algorithms

6.1 K Nearest Neighbours

6.1.1 Description

We used K-nearest neighbors (KNN) algorithm for classification of data points. It is an easy and simple to use algorithm which uses “feature similarity” i.e it helps in predicting the class of a new datapoint on the basis of how similar it is to other data points. It is called a non-parametric algorithm because it uses the data to form the model without any prior knowledge. It also doesn’t have any explicit training phase thus making training faster (lazy algorithm). We used the sklearn.neighbors classifier from scikit library to implement KNN algorithm. For every song in the test set, its euclidean distance was calculated from every other song from the training set. Subsequently, ‘k’ nearest neighbours were chosen and the label that was the majority amongst these neighbours, was assigned.

6.1.2 Graphical Representation Of The Results



	0	1	2	3	4	5	6	7	8	9
0	8	4	6	0	0	0	0	0	0	0
1	0	24	4	0	0	0	0	0	0	0
2	2	5	10	2	0	0	0	0	0	0
3	0	0	3	8	5	0	1	1	0	0
4	0	0	2	3	7	0	1	2	0	0
5	0	3	3	2	1	10	1	0	2	0
6	0	0	0	1	1	1	17	0	1	0
7	0	0	0	0	0	0	1	17	2	0
8	0	0	0	0	0	1	0	3	13	4
9	0	0	0	0	0	0	1	3	5	9

Classification Report

	precision	recall	f1-score	support
0	0.80	0.44	0.57	18
1	0.67	0.86	0.75	28
2	0.36	0.53	0.43	19
3	0.50	0.44	0.47	18
4	0.50	0.47	0.48	15
5	0.83	0.45	0.59	22
6	0.77	0.81	0.79	21
7	0.65	0.85	0.74	20
8	0.57	0.62	0.59	21
9	0.69	0.50	0.58	18
accuracy	-	-	0.61	200
macro avg	0.63	0.60	0.60	200
weighted avg	0.64	0.61	0.61	200

6.1.3 Conclusion

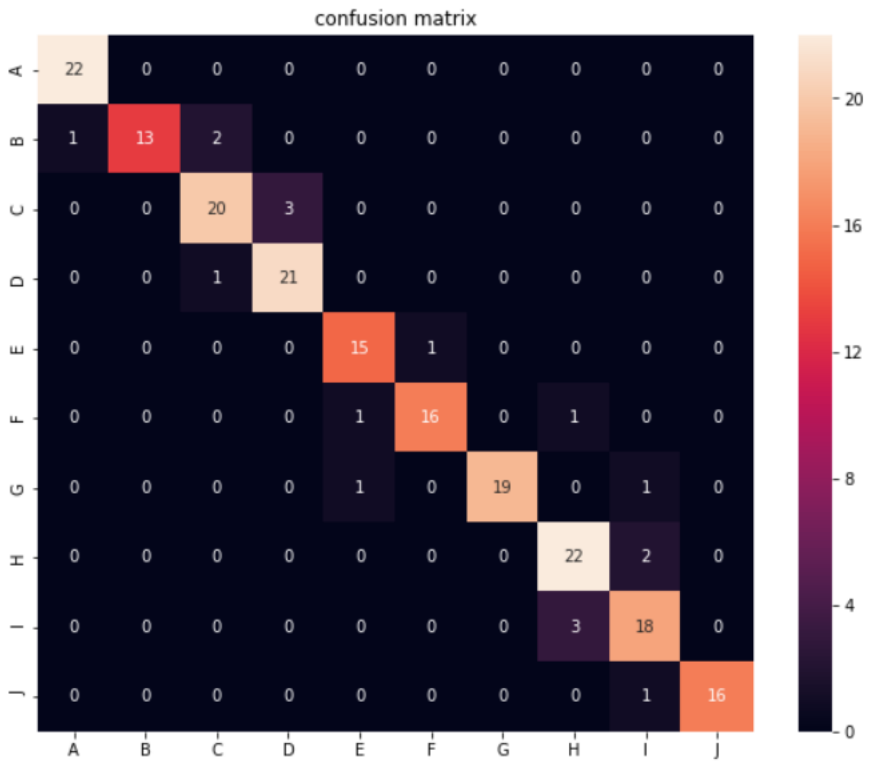
By using different values of K, the above classification report gave the maximum possible accuracy using K Nearest neighbours of **0.61**.

6.2 Naive Bayes

6.2.1 Description

Naive Bayes algorithms are a part of the supervised learning algorithms.The word “Naive” refers to the fact that we considered each of the features, namely, zero crossing rate, chroma frequency, spectral centroid, spectral roll off and mel frequency cepstral coefficient to be independent of each other. We used Gaussian Naive Bayes in particular because the input values were continuous or real valued. Naive Bayes allows multiclass prediction and performs well when the number of features are less and the dataset is small which made it a good choice for our problem statement. We first generated the model using gnb=GaussianNB() function. After that we trained the model using gnb.fit() and finally predicted the response with the help of gnb.predict().

6.2.2 Graphical Representation Of The Results



Classification Report

	precision	recall	f1 - score	support
0	0.96	1.00	0.98	22
1	1.00	0.81	0.90	16
2	0.87	0.87	0.87	23
3	0.88	0.95	0.91	22
4	0.88	0.94	0.91	16
5	0.94	0.89	0.91	18
6	1.00	0.90	0.95	21
7	0.85	0.92	0.88	24
8	0.82	0.86	0.84	21
9	1.00	0.94	0.97	17
accuracy	-	-	0.91	200
macro avg	0.92	0.91	0.91	200
weighted avg	0.91	0.91	0.91	200

6.2.3 Conclusion

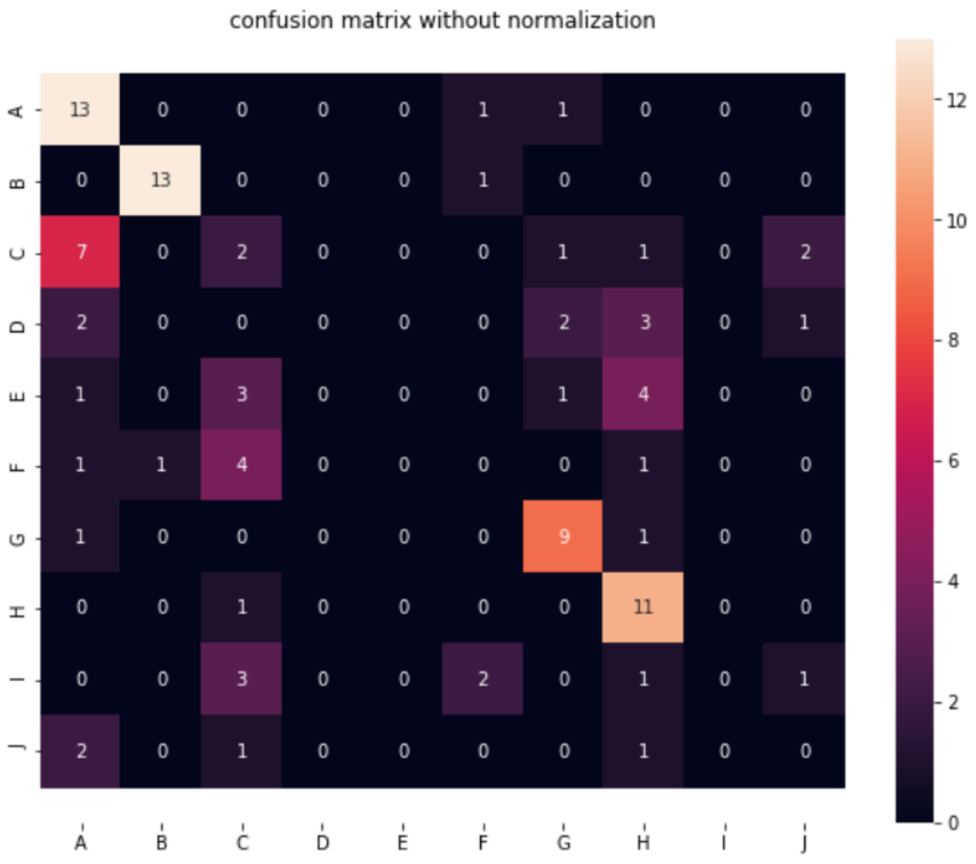
From the above classification report ,it was concluded that the accuracy was **0.91**.

6.3 Neural Network

6.3.1 Description

We used the Dense Neural Network in our analysis. Dense Neural Network is a type of Neural Network in which all layers are fully connected i.e all the neurons are connected to the neurons of the next layer. We used Dense Neural Network over the linear classification because the densely connected layers provide every learning feature from the previous layer rather than providing limited features like linear classifiers. We implemented a Dense Neural Network using Keras and TensorFlow library. The dimensions of input, hidden and output layer according to our model are 26, 16 and 10 respectively. We used the ReLu activation function with cross entropy and added the layers using the model.add() function.

6.3.2 Graphical Representation Of The Results



Classification Report

	precision	recall	f1-score	support
0	0.18	0.22	0.20	9
1	0.67	0.91	0.77	11
2	1.00	0.09	0.17	11
3	0.26	0.50	0.34	10
4	0.00	0.00	0.00	11
5	0.00	0.00	0.00	8
6	0.56	0.83	0.67	6
7	0.43	1.00	0.60	13
8	0.40	0.67	0.50	9
9	0.00	0.00	0.00	12
accuracy	-	-	0.42	100
macro avg	0.35	0.42	0.33	100
weighted avg	0.35	0.42	0.32	100

6.3.3 Conclusion

It can be concluded from the classification report shown above that the accuracy using Neural Network came out to be **0.42**.

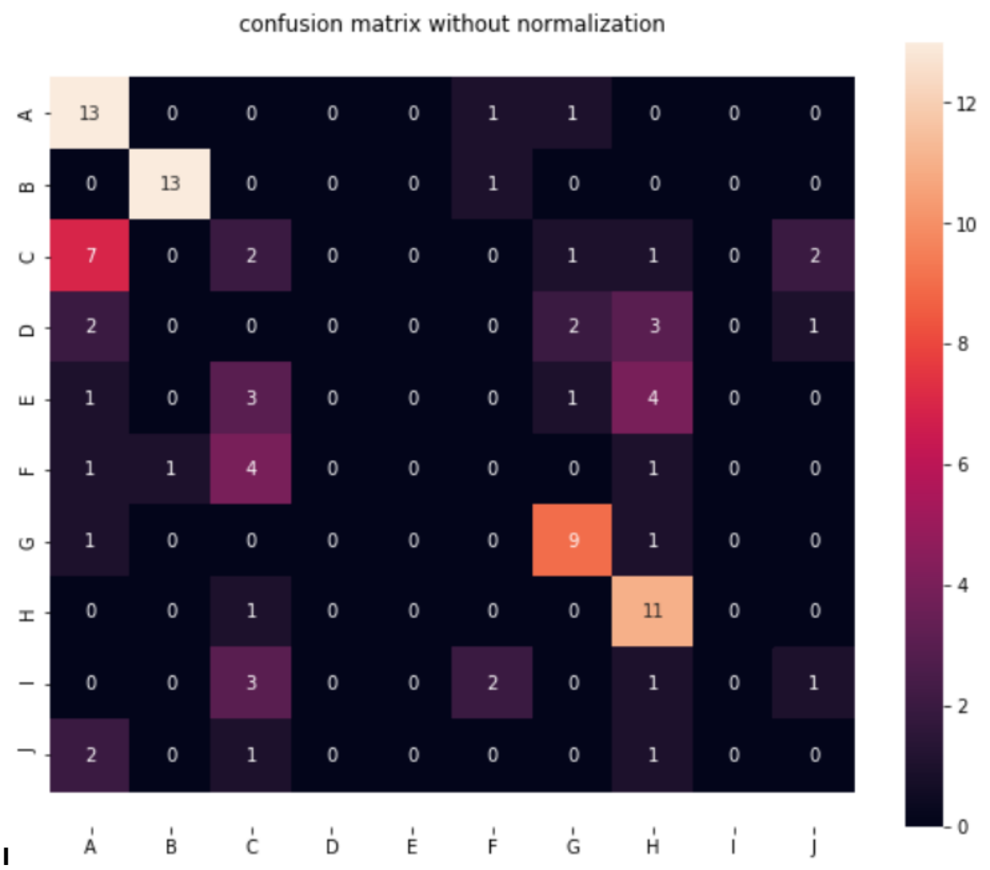
ACCURACY AFTER REMOVAL OF CERTAIN FEATURES

Accuracy after removing the following parameters are :-

FEATURE	ACCURACY
Spectral_centroid	0.21
Spectral_bandwidth	0.16
Rolloff	0.26
zero_crossing_rate	0.2
Mfcc	0.15
Chroma_stft	0.1
rmse	0.28

- This shows that the removal of some features have more impact than removal of the other features
- Chroma_stft has the maximum impact on the accuracy of the algorithm as its removal leads to the least accuracy.
- This shows that Chroma_stft is an important feature in this dataset.
- A Combination of all the important features (chroma_stft, spectra_bandwidth, spectral_centroid and all the mfcc) based on the above table increased the accuracy from 0.42 to 0.48

6.3.4 Graphical Representation Of The Results



	A	B	C	D	E	F	G	H	I	J
A	13	0	0	0	0	1	1	0	0	0
B	0	13	0	0	0	1	0	0	0	0
C	7	0	2	0	0	0	1	1	0	2
D	2	0	0	0	0	0	2	3	0	1
E	1	0	3	0	0	0	1	4	0	0
F	1	1	4	0	0	0	0	1	0	0
G	1	0	0	0	0	0	9	1	0	0
H	0	0	1	0	0	0	0	11	0	0
I	0	0	3	0	0	2	0	1	0	1
J	2	0	1	0	0	0	0	1	0	0

Classification Report

	precision	recall	f1-score	support
0	0.29	0.50	0.37	10
1	0.45	1.00	0.62	9
2	0.00	0.00	0.00	5
3	0.75	0.30	0.43	10
4	0.28	0.78	0.41	9
5	0.33	0.07	0.12	14
6	0.86	0.80	0.83	15
7	0.65	0.79	0.71	14
8	0.00	0.00	0.00	5
9	0.00	0.00	0.00	9
accuracy	-	-	0.48	100
macro avg	0.36	0.42	0.35	100
weighted avg	0.44	0.48	0.41	100

6.3.5 Conclusion

The accuracy after removing rolloff,rmse,zero_crossing_rate came out be **0.48**

6.4 Logistic Regression

6.4.1 Description

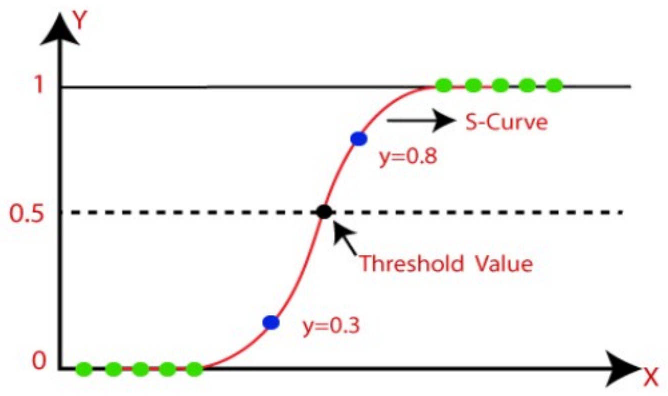
Logistic regression is a classification algorithm that underlies supervised learning and helps to estimate the probability of selected variants. In this case the results are binary (dichotomous) in nature, ie. either 0 indicating failure / no or 1 indicating success / yes.
We used this method because it predicts the effect (target variable) as discrete values.
We used multiclass logistic regression as we have 10 classes. When it comes to the classification function, the algorithm is most preferred because:

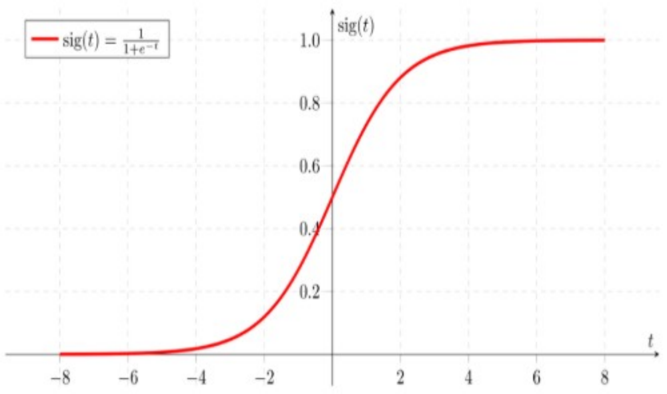
- Easy to interpret the result
- It does not require many computational resources
- Easy to regularize

The algorithm was implemented using the sklearn library in python. First, the model was created using the log = LogisticRegression () function.After that model was trained using log model.fit () and finally the result was predicted with the help of log model.predict () and accuracy was calculated. Parameters evaluated using function parameters = logmodel.coef_.

We also calculated the accuracy by removing various features and saw that the accuracy increased by **1%** by removing the chroma_stft and rmse features together.

In the figure below, it is the S-curve or Sigmoid curve that simply converts any value between -∞ to ∞ into discrete values.





Logistic Regression hypothesis is defined as:

$h_\Theta(x) = g(\Theta^T x)$

where function g is the sigmoid function, which is defined as below:

$g(z) = \frac{1}{1+e^{-z}}$

For multiclass we have,

$y \epsilon (0, 1, ...n)$

$h_\Theta^{(0)} = P(y = 0|x; \Theta)$

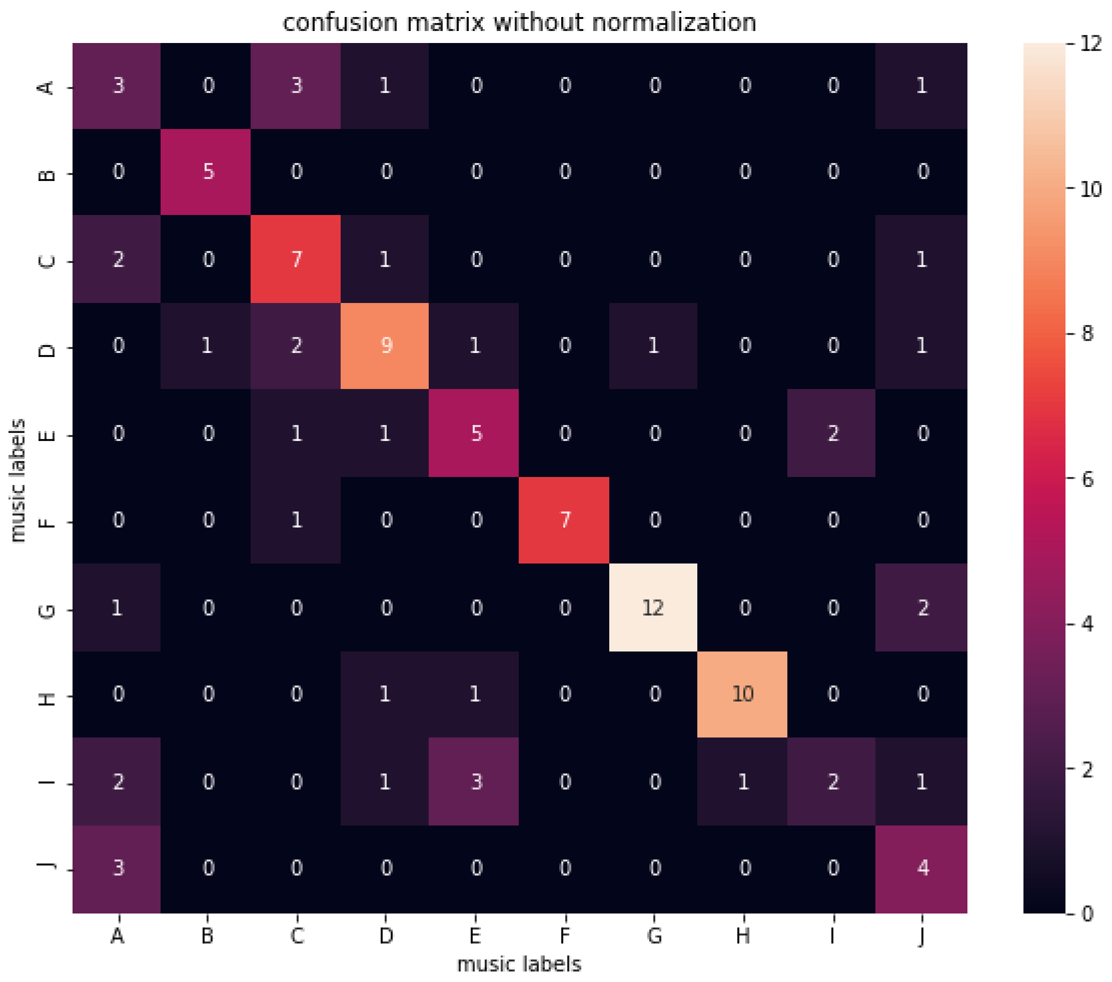
.

.

.

$h_\Theta^{(n)} = P(y = n|x; \Theta)$

6.4.2 Graphical Representation Of The Results



Classification Report

	precision	recall	f1 - score	support
0	0.27	0.38	0.32	8
1	0.83	1.00	0.91	5
2	0.50	0.64	0.56	11
3	0.64	0.60	0.62	15
4	0.50	0.56	0.53	9
5	1.00	0.88	0.93	8
6	0.92	0.80	0.86	15
7	0.91	0.83	0.87	12
8	0.50	0.20	0.29	10
9	0.40	0.57	0.47	7
accuracy	-	-	0.64	100
macro avg	0.65	0.64	0.63	100
weighted avg	0.67	0.64	0.64	100

6.4.3 Conclusion

From the above confusion matrix and the table , we can see that the accuracy using Logistic Regression comes out to be **0.64**.

6.5 Support Vector Machine

6.5.1 Description

Being a supervised machine learning algorithm, the Support Vector Machine was used for classification of the audio files. It performed the task by constructing a decision boundary using the data points presented in the dataset.

The decision boundary was such that the data points belonging to different labels were separated. But there were several possible boundaries which did the same, hence, SVM was used to choose the one in which nearest points(Support Vectors) were at maximum distance from the boundary.

When it comes to the task of classification, the algorithm was highly preferred due to following reasons:
1. Significant accuracy
2. Less computation power

In order to fit the decision boundary of higher dimensional data in an optimised way, the **kernel trick** was performed too.

This algorithm was implemented via **sklearn**.

We implemented both **linear SVM** and **non-linear SVM** in our analysis.

Linear SVM
In linear SVM, linear decision boundary was analysed.

Non-linear SVM
Since we had more than two class labels, non-linear decision boundary was analysed too using non-linear SVM. This was done using the Kernel trick which aimed to find a hyperplane in the multidimensional space to correctly separate the data points from each other.

There were different types of kernels which we used:

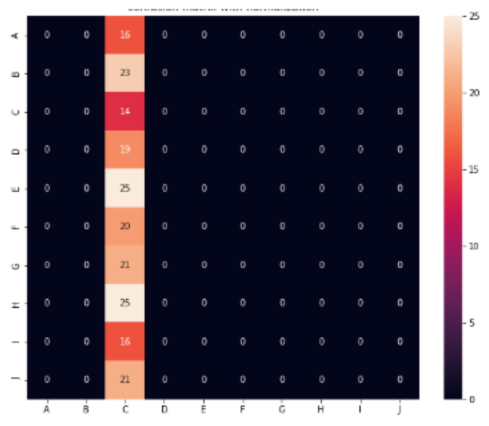
1.Polynomial

2.RBF

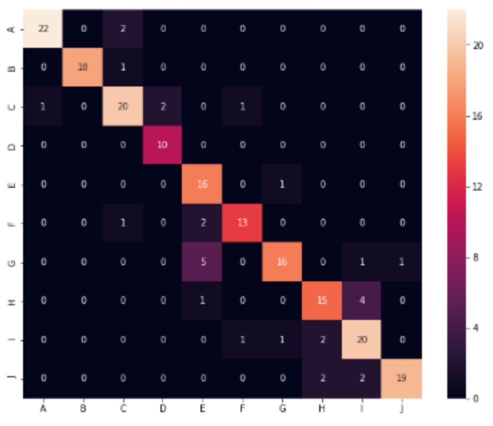
3.Sigmoid

6.5.2 Graphical Representation Of The Results

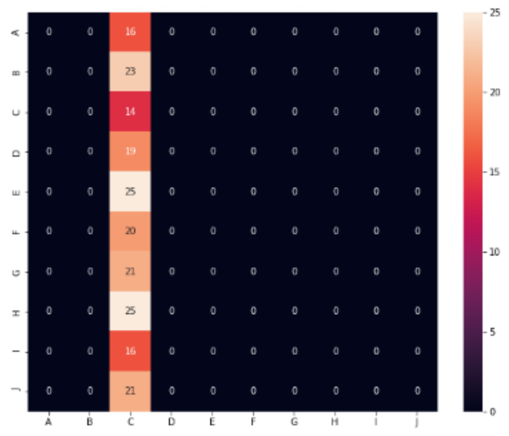
Sigmoid



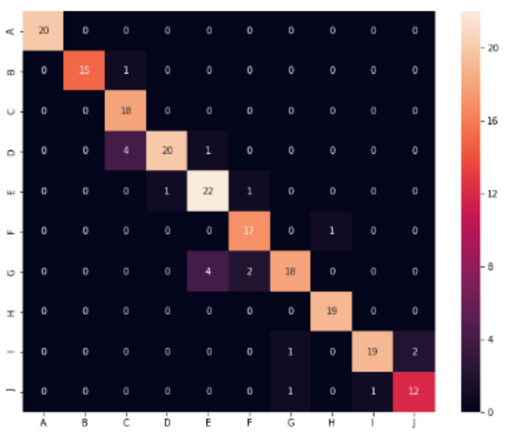
Polynomial



RBF



Linear



Linear SVM

	precision	recall	f1-score	support
0	1.00	1.00	1.00	20
1	1.00	0.94	0.97	16
2	0.78	1.00	0.88	18
3	0.95	0.80	0.87	25
4	0.81	0.92	0.86	24
5	0.85	0.94	0.89	18
6	0.90	0.75	0.82	24
7	0.95	1.00	0.97	19
8	0.95	0.86	0.90	22
9	0.86	0.86	0.86	14
accuracy	-	-	0.90	200
macro avg	0.91	0.91	0.90	200
weighted avg	0.91	0.90	0.90	200

Polynomial SVM

	precision	recall	f1-score	support
0	1.00	1.00	1.00	20
1	1.00	0.94	0.97	16
2	0.78	1.00	0.88	18
3	0.95	0.80	0.87	25
4	0.81	0.92	0.86	24
5	0.85	0.94	0.89	18
6	0.90	0.75	0.82	24
7	0.95	1.00	0.97	19
8	0.95	0.86	0.90	22
9	0.86	0.86	0.86	14
accuracy	-	-	0.86	200
macro avg	0.86	0.86	0.85	200
weighted avg	0.86	0.85	0.85	200

RBF SVM

	precision	recall	f1-score	support
0	0.00	0.00	0.00	17
1	0.00	0.00	0.00	24
2	0.08	1.00	0.14	15
3	0.00	0.00	0.00	15
4	0.00	0.00	0.00	15
5	0.00	0.00	0.00	24
6	1.00	0.12	0.21	26
7	1.00	0.10	0.18	20
8	0.00	0.00	0.00	24
9	0.00	0.00	0.00	20
accuracy	-	-	0.10	200
macro avg	0.21	0.12	0.05	200
weighted avg	0.24	0.10	0.06	200

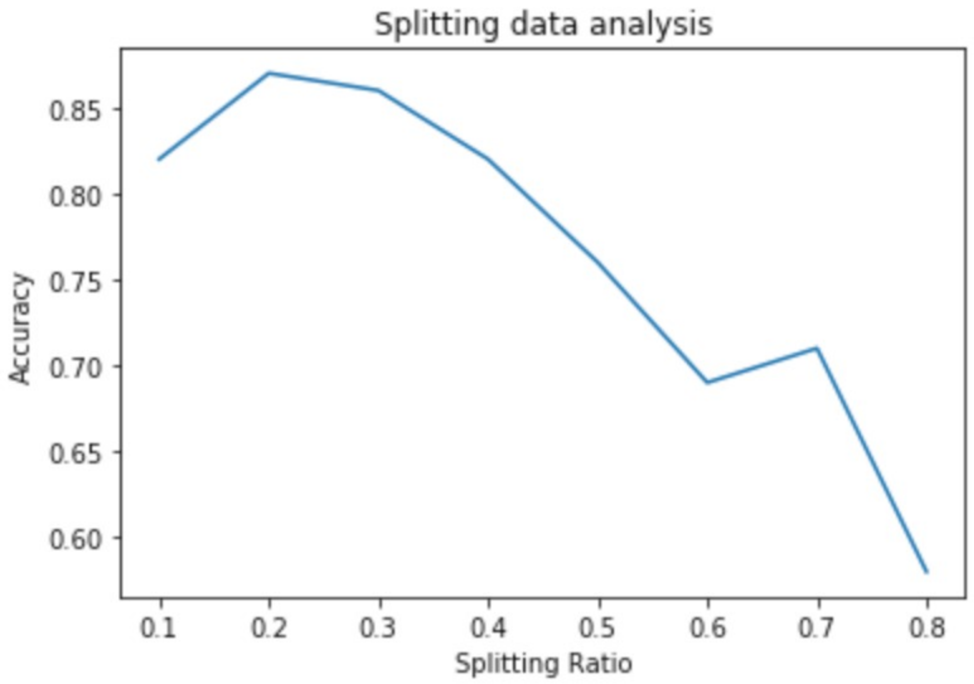
Sigmoid SVM

	precision	recall	f1-score	support
0	0.00	0.00	0.00	16
1	0.00	0.00	0.00	23
2	0.07	1.00	0.13	14
3	0.00	0.00	0.00	19
4	0.00	0.00	0.00	25
5	0.00	0.00	0.00	20
6	0.00	0.00	0.00	21
7	0.00	0.00	0.00	25
8	0.00	0.00	0.00	16
9	0.00	0.00	0.00	21
accuracy	-	-	0.07	200
macro avg	0.01	0.10	0.01	200
weighted avg	0.00	0.07	0.01	200

ANALYSIS AFTER VARYING TRAINING AND TESTING DATA

Training split percentage	Testing split percentage	Accuracy(%)
90	10	82
80	20	87
70	30	86
60	40	82
50	50	76
40	60	69
30	70	71
20	80	58

GRAPHICAL REPRESENTATION



Since, there was no sudden increase/decrease in the accuracy of the model when splitted in different ratios. Thus,it can be concluded from above that the quality of features were good and data fitted the model well.

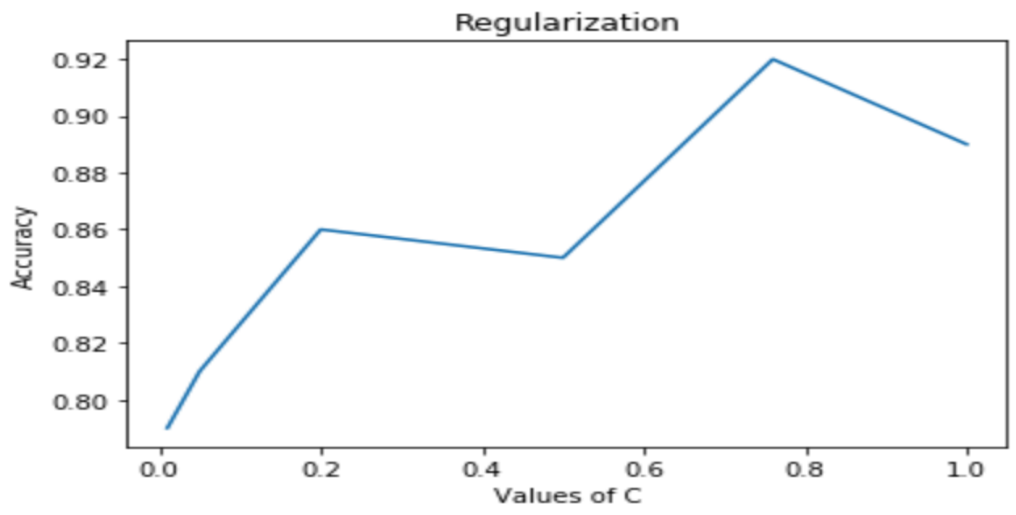
6.5.3 Regularization

The regularization is a technique to deal with the problem of overfitting by fine tuning parameters which are given as an input to the SVM classifier. It is of great importance as its purpose is to reduce miss-classifications as much as possible.

As from above, a conclusion can be drawn that maximum accuracy was that of Linear SVM and hence, to achieve more accurate results regularization was performed on the model. The lambda (C parameter) has by default 1.0 value.

The different values of C were chosen and the accuracy were analysed as shown below:

LAMBDA (C)	ACCURACY
0.01	0.79
0.05	0.81
0.30	0.86
0.50	0.85
0.75	0.92
1.0	0.89



After performing regularization, it was observed that an accuracy of **0.92** was achieved which was earlier **0.90**.

6.5.4 Conclusion

The accuracy for different models of SVM method are:

(1) Linear SVM(without regularization): 0.90

(2) Linear SVM(with regularization C = 0.75): 0.92

(3) Polynomial SVM: 0.86

(4) RBF SVM: 0.10

(5) Sigmoid: 0.07

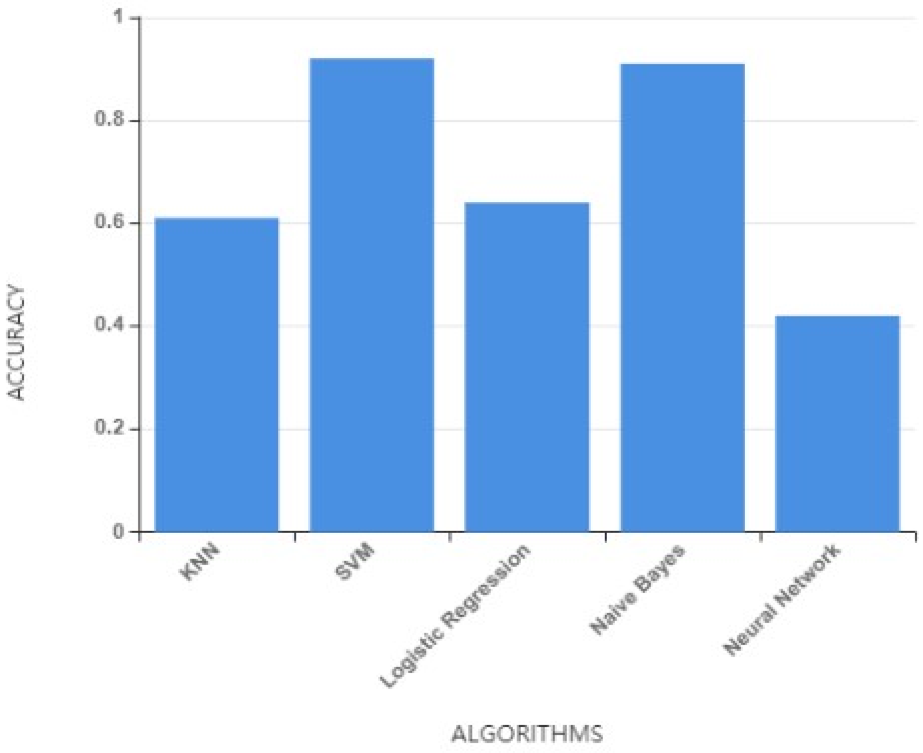
Hence, it was concluded that **LINEAR SVM with C = 0.75** gives the best accuracy.

7

Conclusion

In this project, we aimed to classify a given audio sample into one of 10 genres, by training the model on a dataset of 1000 samples. First, we identified and extracted the features that could be used to differentiate among the genres and using these, we fed the data into our machine learning algorithms.

A variety of algorithms were implemented and they gave us varying degrees of accuracy. The highest accuracy was achieved by **Linear SVM**. An accuracy of 92% was obtained by applying L1-regularization and using the entire set of features in the Linear SVM model.



COMPARISON OF IMPLEMENTED ALGORTIHMS

Algorithm used	**Accuracy**
KNN	0.61
SVM	0.92
Logistic Regression	0.64
Naive Bayes	0.91
Neural Networks	0.42

8

Future Work

While we were able to achieve a reasonable accuracy from the above work, there are still some areas where more work can be done. These are:

- **Broaden our training data**
  We used the GTZAN dataset in our analysis. This dataset gave us useful insights and results. However, to improve the practicality of our work, the next step would be to work on more extensive data. This can be done by exploring more genres. There is a large variety of genres and subgenres in the market, apart from the main ones we used. Using these additional genres, the model would become more thorough.
- **Feature Refining**
  The results can be obtained more accurately by identifying other musically relevant features and incorporating them in the models.
- **Practical Application**
  This project can be applied to real life scenarios. It can be used by music streaming apps such as Spotify to improve their recommendation systems based on a user's existing preferences. We can find how closely the genres are related to each other, and use this to recommend music from either the same genre or from related genres.

9

References

(1) http://cs229.stanford.edu/proj2017/final-reports/5244969.pdf

(2) http://home.iitk.ac.in/ archit/cs365/project/report.pdf

(3) https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8

(4) https://medium.com/analytics-vidhya/music-genre-classification-with-python-51bff77adfd6

(5) http://cs229.stanford.edu/proj2017/final-reports/5244969.pdf

(6) http://marsyas.info/downloads/datasets.html