ISyE 6740 – Fall 2021
Final Report

Team Member (Solo):
Jaykumar Kakkad (GT ID: 903661208)

Project Title:

Exploring Machine Learning for Music Classification

Source: https://www.pngwing.com/en/free-png-zbbyg

Abstract

In this project, we investigate the problem of Music Genre classification with the aim of improving accuracy compared to the Paper[1] that used the KNN model. We used Short term fourier transform (STFT), a time-frequency analysis method, to create spectrograms of audio files. Using the Spectrogram, we studied and derived 36 key features for each audio signal. We then use PCA to visualize these features in 2D, use KDE & LOF to remove outliers and Mutual information to conduct feature selection. Based on the data analysis, we decided to fit five different models - KNN, SVM-Kernel, CART, Random Forest and Adaboost. Based on various experiments, we found that SVM-Kernel (71% accuracy) was able to outperform the KNN Model (63% accuracy).

Background

In the last 2 decades, there have been exciting innovations in the area of audio processing. Last year, Google came up with an extremely innovative product. Now, one can just hum a song and the google assistant will identify the song for you.

In this course, we have learned and used various algorithms on real numbers and images. However, we haven't used these algorithms on Audio data. This could be because the features in audio data are not very straight forward. I wonder how useful the ML algorithms that we learnt in this course would be in classifying and predicting the audio data.

Problem Statement

In the project, I plan to explore the use of audio data on the ML Algorithms we have learnt. I plan to answer below key questions:

a) What are various features one can extract from a Music file that can be considered relevant?

b) How can we extract these features from Music files? Can PCA or equivalent methods be used to visualize the data?

c) In this case, we plan to classify the audio tracks into various Genres. Is there a way to have better accuracy than Paper[1] classifying audio?

The paper[1] discusses various audio features and implements GMM based classifiers and KNN to classify audio into various Genres. This paper shows the classification accuracy of around 61%. Hency, my Goal is to use the same dataset and try to improve the accuracy of Genre Classification by using different techniques.

Data Source

I used the same dataset as used by the paper[1]. Marsyas data
This dataset consists of 1000 audio tracks of 10 different Genres (100 each). Each audio file is 16bit mono of 30 seconds and is in .wav format with sampling rate of 22,050 samples/second. The key problem now is to figure out appropriate features to extract from these samples.

Methodology

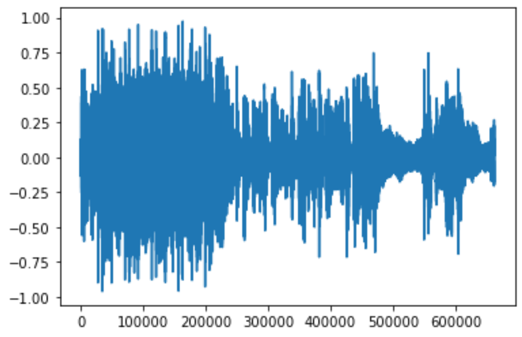
Our methodology can be divided into 3 parts: **a) Feature Extraction**: We started with extracting relevant features from the Audio files. Using a spectrogram, we extracted 36 features from each audio file. **b) Exploratory data analysis**: These features were visualized to find out which algorithms can be useful. As a part of EDA, we then detected outliers using KDE and LOF and decided to remove a few. Also, we used correlation matrix and Mutual information to perform feature selection. **c) Classification:** Finally, we fitted five classification algorithms and conducted several experiments.

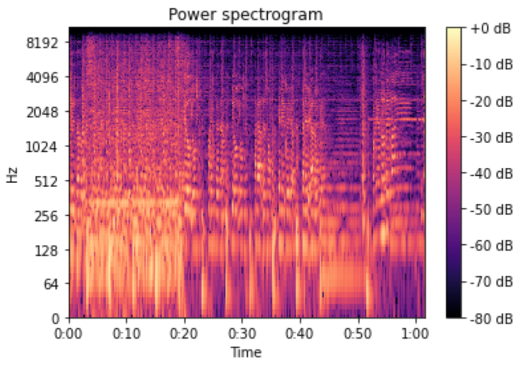
I) Extracting features

Making a Spectogram:

Every audio is a time series that shows change in amplitude and frequency. Just knowing amplitude isn't very useful, but knowing the changes in frequency can help us create many features to uniquely represent audio. Hence, the first step is to create a Spectrogram. The Spectrogram helps us find the Spectrum of frequencies with respect to time.

Audio signal **to →** **Spectrogram**





We referred to [3] and [4] to understand and implement various steps to extract a spectrogram from each audio signal. It involves many steps:

- Discrete Fourier transform (DFT): The main idea is that a signal is projected on various complex sinusoids (the dot product) with different frequencies. For the frequency present in the signal the value will be higher than others.

$$x[k] = \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi nk/N} \quad \text{for } k = 0,1,2,.. N-1$$

Where:
x[k] is the spectrum output
x(n) is the input sample
$e^{-j2\pi nk/N}$ is complex sinusoids from Euler's formula ($cos(\Theta) + j\,Sin(\Theta)$)
N is number of samples

- Fast Fourier transform (FFT): It is just an algorithm that takes advantage of the symmetry in the signal and executes the DFT faster. It only works if the size of the samples is of the size 2^N . FFT size is generally kept higher than the window size and zero padding is used which helps in improving frequency resolution.

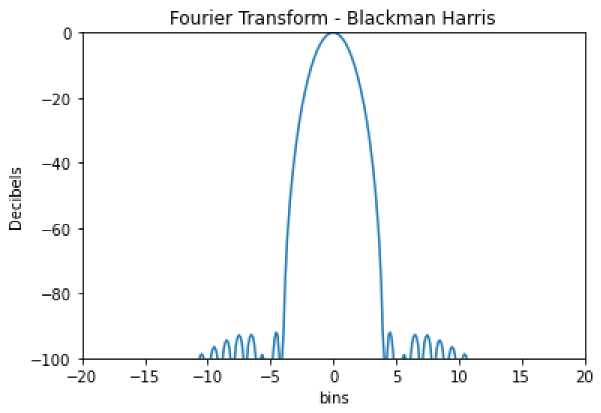
- Short-time Fourier Transform (STFT): STFT is just a time varying version of DFT. It is a process wherein we divide the signal into smaller frames and conduct a DFT on each frame to determine the frequency spectrum of the window.

$$X_l[k] = \sum_{n=-N/2}^{N/2-1} w[n]\, x[n + lH]\, e^{-j2\pi nk/N} \quad l = 0, 1....$$

Where,
l is a frame number
w[n] is analysis window
H is hop size

- Windowing : For Non integer values, the DFT does not work properly and frequency content spreads over the full range of the frequencies. This is called Spectral leakage and to mitigate this phenomenon windowing is used. Hence, every frame of STFT is multiplied by a Window (w). Since this is an audio signal, we need lower noise to signal ratio and hence Blackman Harris was used as a window function. The size of the window is selected based on the time- frequency trade-off. I have selected the size of the window as 1000 so as to get better frequency resolution and detect appropriate harmonics.

$$w[n] = 1/M \sum_{l=0}^{3} \alpha_l\, cos(2\pi ln/M) \quad n = -M/2,...\ M/2\,;$$
$$\alpha_0 = 0.3587,\ \alpha_1 = 0.488,\ \alpha_2 = 0.0116$$

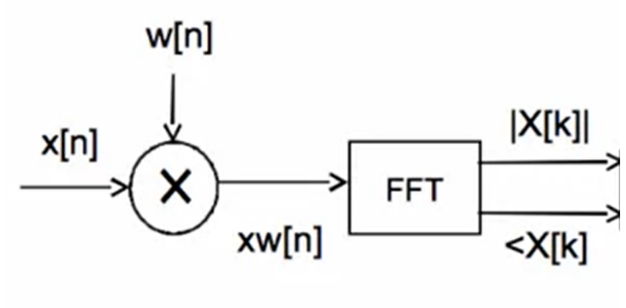


- Hop Size: The hop size is selected such that the window functions smooths over time and does not create any unnecessary variations. A rule of thumb is to keep hop size around 1/4th of the window size to have a smooth function.

Algorithmic parameters for STFT:
Based on the above discussion, following algorithmic parameters were set to extract a spectogram:

- Fast Fourier window size = 2048
- Hop Size =250
- Window Length = 1000
- Window Type = 'blackmanharris'

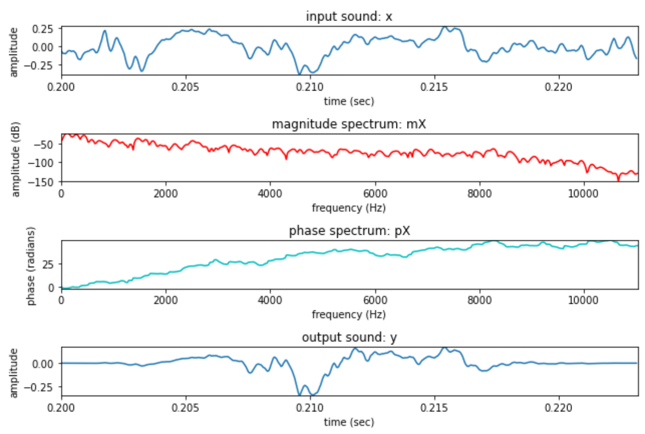
STFT Process to make a Spectrogram

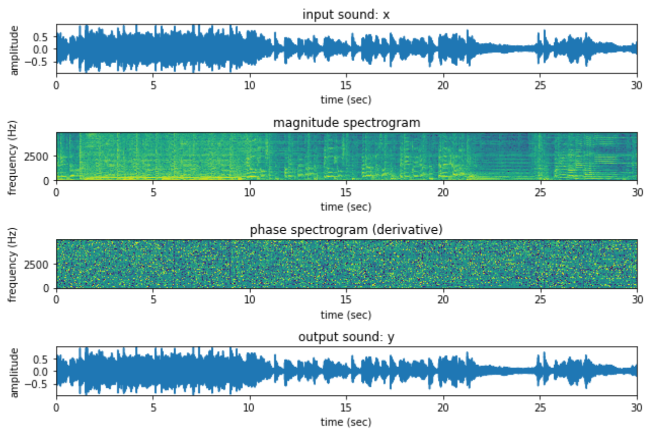


Source: Audio Signal processing for music [3]

DFT of a part of signal

STFT of a Signal (Spectrogram)





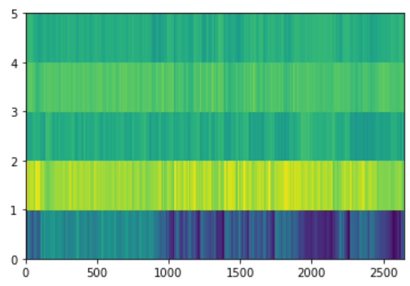
Source: Software used from Audio Signal processing for music [3]

Can Changing Window size and type impact accuracy of Algorithm?
This is an obvious question we had. However, the research paper [6] shows that there is not much difference in accuracy between 256,512 and 1024 window sizes. Also, the paper found that the window Parzen had the best accuracy which means these signals require lower noise decibels. Hence, our choice of Blackman-Harris (decibels = -90) is better than Parzen (decibels = -50). Based on this research, we think our choice of window size and window type is optimal for this problem.

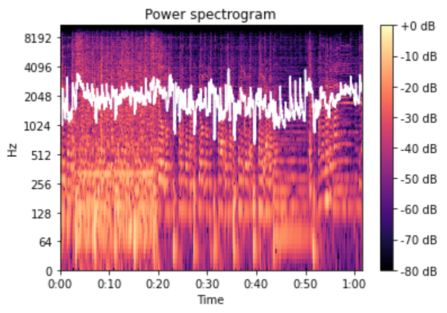
Extracting Features from Spectrogram

We extracted 36 features in total based on Spectrogram, Pitch classes and Tempogram.

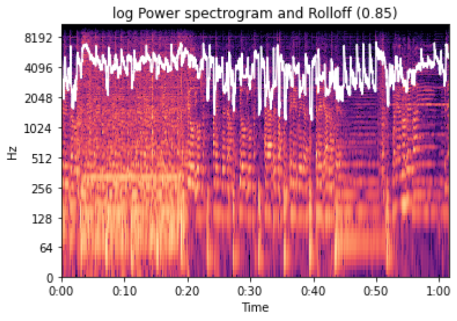
1. **MFFC's Coefficients**



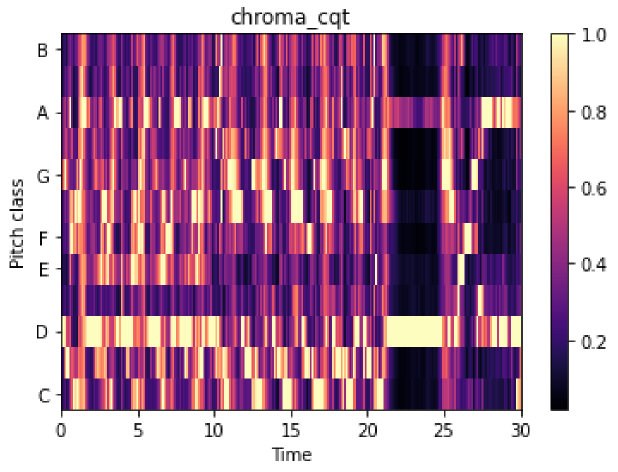
2. **Centroid**



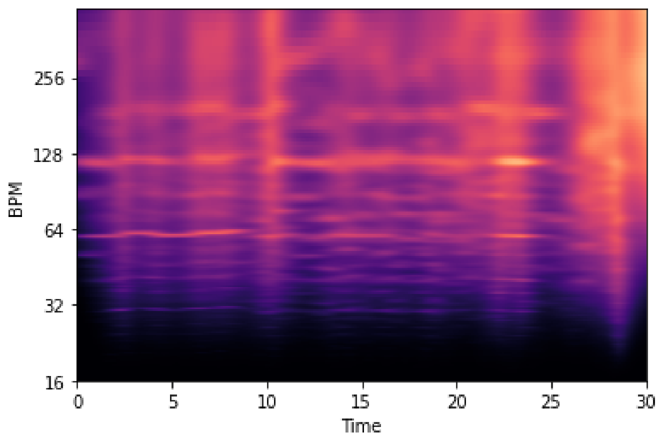
3. **Spectral Rolloff**



4. **Pitch class**



5. **Tempogram**



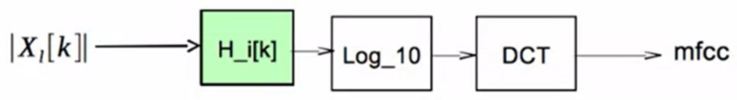
Source: Using Librosa library

1. **<u>Mel Frequency Cepstral Coefficients (MFCC):</u>**

MFCC is a cosine transform of log magnitude spectrum on mel scale. The Mel scale is used so as to make the feature closer to how humans hear. Only the top 5 MFCC coeff's are considered as they consist of high information. For each coeff, we have included mean, variance and skewness leading to 15 features in total

$$MFCC_l = Discrete\ Cosine\ Transform\left\{ log_{10}\left(\sum_{k=0}^{N/2} |X_l[k]|\ H_i[k] \right) \right\}$$

where,
L : frame, X[k] is spectrum and H[k] is mel scale filter bank for each filter i



Source: Audio Signal processing for music [3]

2. **Centroid :**

The spectral centroid is the weighted mean of frequencies present in the signal. The feature can be thought of as related to brightness of the signal. We have considered the mean and variance of centroid over the period adding two features to our feature set

$$Centroid_l = \frac{\sum_{k=0}^{N/2} k * \left|X_l[k]\right|}{\sum_{k=0}^{N/2} \left|X_l[k]\right|}$$

3. **Spectral Rolloff:**

The spectral rolloff is the frequency below which 85% of magnitude distribution is present. This measure is related to the overall shape of the spectral. We consider the mean and variance of Rolloff over the period adding two features.

4. **Low Energy feature:**

It is defined as % of analysis windows that have less RMS energy than the Average RMS energy. It is a measure of the loudness of the audio. This adds one feature to the feature set.

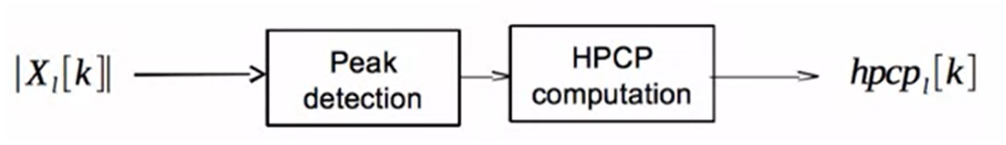
RMS energy is defined as : $\sqrt{\frac{1}{N^2}\sum_{k=0}^{N-1}\left|X_l[k]\right|}^{\,2}$

5. **Zero Crossings in time domain:**

The number of zero crossings provide a measure of noisiness of an audio signal

6. **Chroma / Pitch classes:**

The spectrogram is used to create an harmonic pitch class profile also known as Chroma. This profiles 12 pitch classes. We have considered the mean of each class adding 12 features to our feature set.



Source: Audio Signal processing for music [3]

7. **Tempogram:**

We use tempo, sum of tempogram mean and max of tempogram mean as features

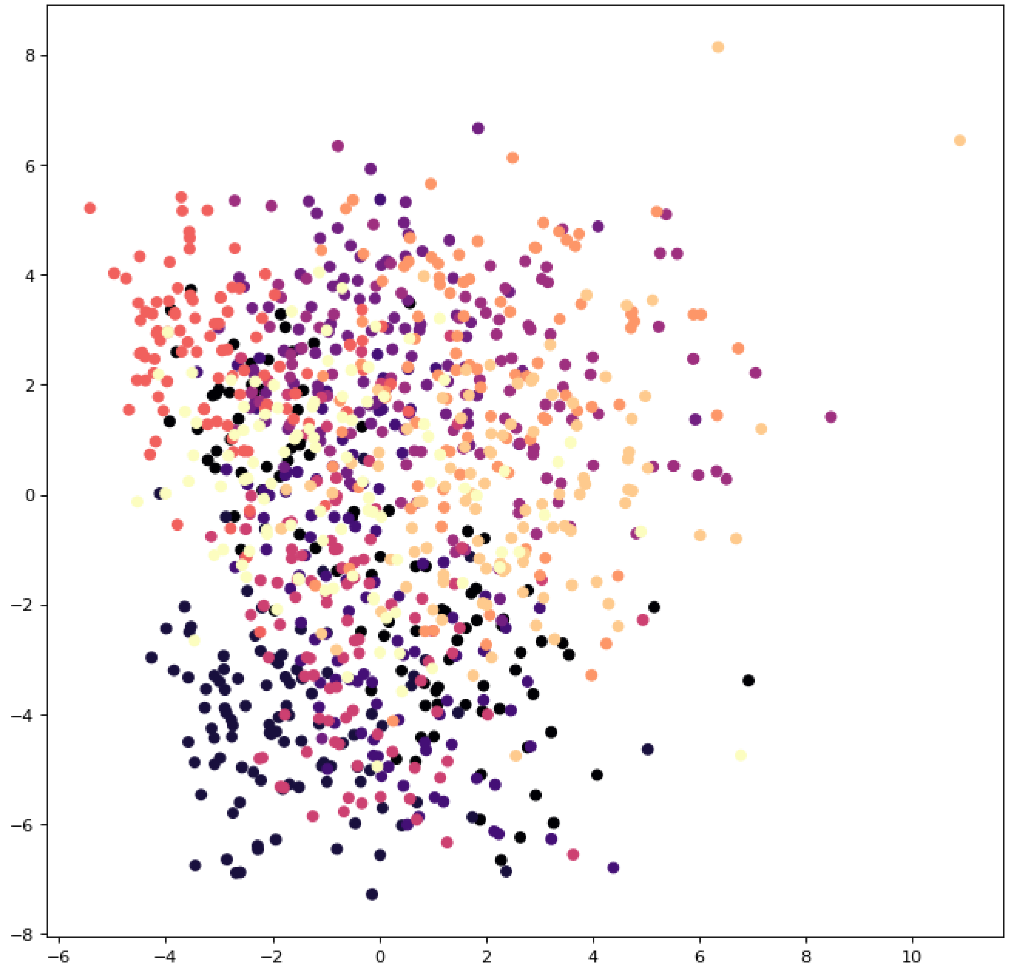
After evaluating various libraries from Paper [9], we have used a python library Librosa to generate all the above features.

II) Exploratory Data Analysis

1) Visualizing data on 2-D space

We first projected the data on 2-D space using principal component analysis to visualize how the various classes are distributed. From the below plot it seems that classes do not have a specific structure and are non-linearly separable. It seems that using algorithms that can classify non-linearly separable data could be a good starting point. Hence, using Kernel SVM, decision tree, Random Forest and Adaboost seems appropriate to me.

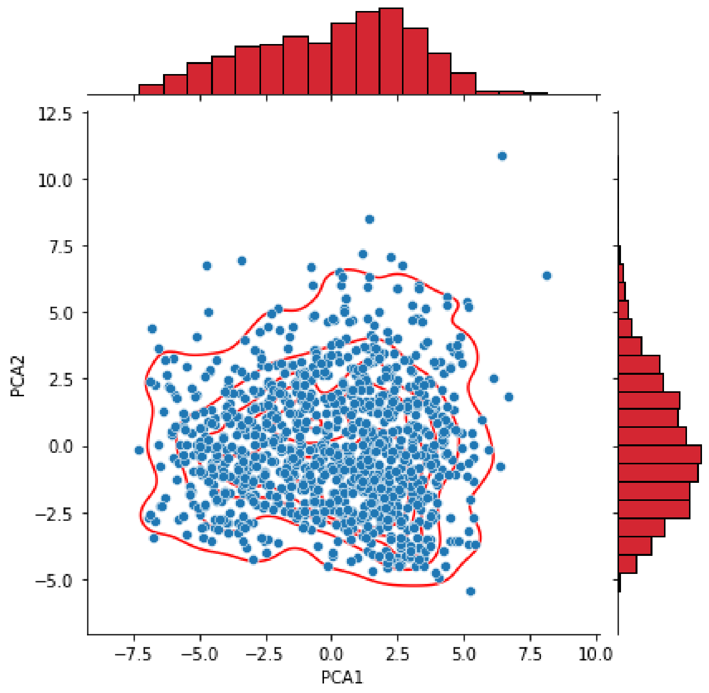
Projection of data points on Two PCA components



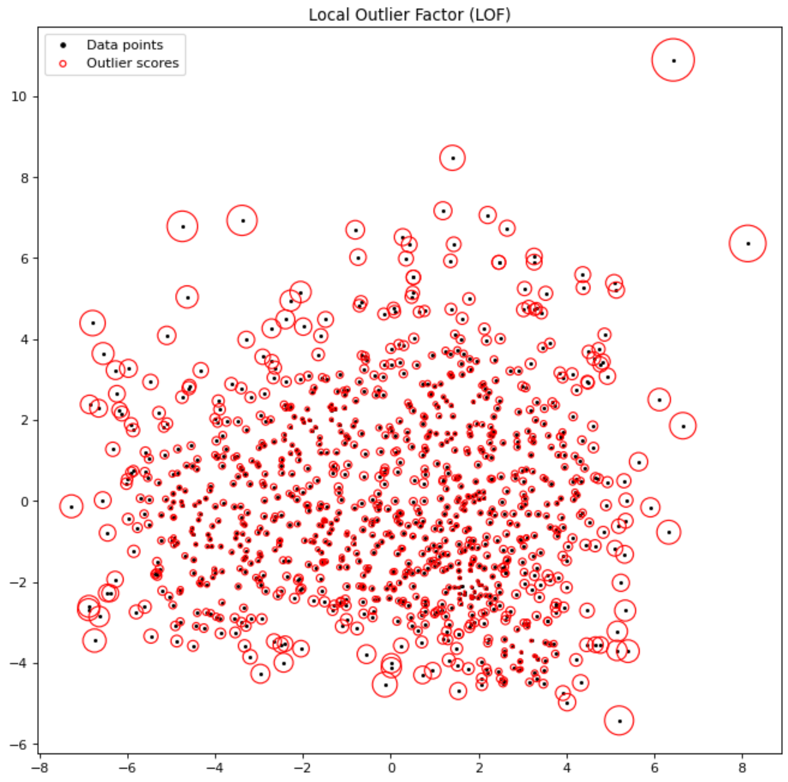
2) Outlier detection - KDE and LOF

From the PCA, it seems there are a few outliers in the data. To further visualize outliers, we decided to use Kernel density estimate (KDE) on the two principal components. From the KDE visualization below, it seems there are around 17-20 outliers which comprises 2% of the data points. The Key question was whether to remove or keep these outliers. Since the Outliers were from different classes (not confined to a specific class), we decided to remove some of them. However, since we were planning to use non-parametric approaches, it felt incorrect to remove outliers based on the global approach of KDE. Hence, we decided to use the **Local outlier factor(LOF)** approach.

Kernel Density Estimate (KDE)



Local Outlier Factor (LOF)



LOF approach is based on finding the density of a data point with respect to the density of its neighbour.

$$\mathrm{LOF}_k(A) := \frac{\sum_{B \in N_k(A)} \frac{\mathrm{lrd}_k(B)}{\mathrm{lrd}_k(A)}}{|N_k(A)|} = \frac{\sum_{B \in N_k(A)} \mathrm{lrd}_k(B)}{|N_k(A)| \cdot \mathrm{lrd}_k(A)}$$

where lrd is local reachability density and is defined as

$$\mathrm{lrd}_k(A) := 1 / \left(\frac{\sum_{B \in N_k(A)} \text{reachability-distance}_k(A, B)}{|N_k(A)|} \right)$$

Source: Paper [7] and wikipedia

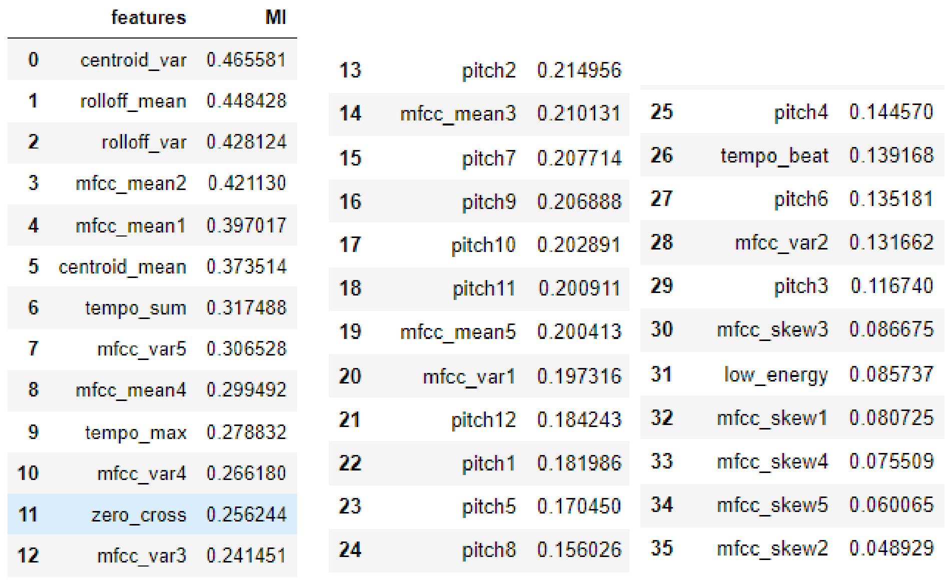
If the LOF score is less than 1, it is definitely considered as inline and LOF score of more than 1 may be considered as outlier. The main disadvantage of this approach is that deciding a LOF score threshold to find outliers isn't very well defined. Hence, we used visual inspection and decided to use a LOF score of 1.5 to remove outliers. Based on this, we removed 17 outliers from the data.

3) Feature selection

We used Librosa library to extract all the features explained in the earlier section. Since all these features are from a time series, our main concern was to check if the features were too correlated. So, we created a correlation matrix.

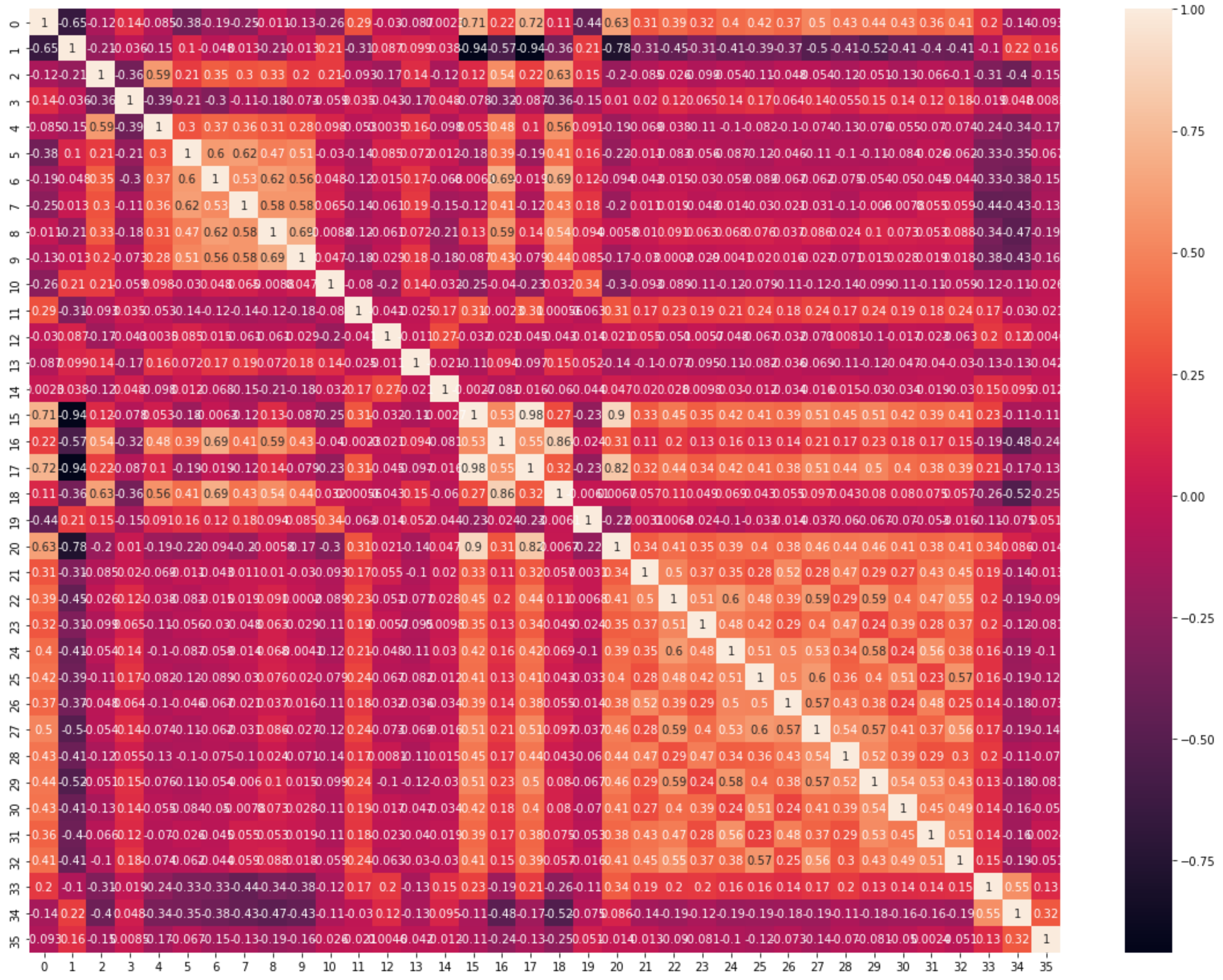
There is slightly higher correlation between features 20-32 which are the features related to pitch classes and features 5-9 which correspond to MFCC variance. While the correlation is not extreme, we still wanted to evaluate whether to remove some of them. To decide whether to remove any of these features, we calculated Mutual information (MI) score for each of them with respect to the dependent variable. These variables seem to have high MI score so we decided to keep all of them.

Mutual information score



	features	MI
0	centroid_var	0.465581
1	rolloff_mean	0.448428
2	rolloff_var	0.428124
3	mfcc_mean2	0.421130
4	mfcc_mean1	0.397017
5	centroid_mean	0.373514
6	tempo_sum	0.317488
7	mfcc_var5	0.306528
8	mfcc_mean4	0.299492
9	tempo_max	0.278832
10	mfcc_var4	0.266180
11	zero_cross	0.256244
12	mfcc_var3	0.241451
13	pitch2	0.214956
14	mfcc_mean3	0.210131
15	pitch7	0.207714
16	pitch9	0.206888
17	pitch10	0.202891
18	pitch11	0.200911
19	mfcc_mean5	0.200413
20	mfcc_var1	0.197316
21	pitch12	0.184243
22	pitch1	0.181986
23	pitch5	0.170450
24	pitch8	0.156026
25	pitch4	0.144570
26	tempo_beat	0.139168
27	pitch6	0.135181
28	mfcc_var2	0.131662
29	pitch3	0.116740
30	mfcc_skew3	0.086675
31	low_energy	0.085737
32	mfcc_skew1	0.080725
33	mfcc_skew4	0.075509
34	mfcc_skew5	0.060065
35	mfcc_skew2	0.048929

Correlation Matrix



Features are: 0:mfcc_mean1, 1:mfcc_mean2, 2:mfcc_mean3, 3:mfcc_mean4, 5:mfcc_mean5, 6:mfcc_var1, 7:mfcc_var2, 8:mfcc_var3, 9:mfcc_var4, 10:mfcc_var5, 11:mfcc_skew1, 12:mfcc_skew2 ,13:mfcc_skew3, 14:mfcc_skew4, 15:mfcc_skew5, 16:centroid_mean, 17:centroid_var, 18:rolloff_mean, 19:rolloff_var , 20:low_energy, 20:zero_cross, 21:pitch 1, 22:pitch 2, 23:pitch 3 24:pitch 4, 25:pitch5,26:pitch6, 27:pitch 7, 28:pitch8, 29:pitch9, 30:pitch10, 31:pitch11, 32:pitch 12, 33:tempo_max, 34:tempo_sum, 35:tempo_beat

III) Classification Models

1. KNN

At the start, KNN was implemented to check whether the accuracy is closer to the accuracy of the paper[1]. KNN assigns x the same label as the closest training samples

$$f(x) = sign\left(\sum_{i \epsilon I_k(x)} y^i\right)$$

where,

I(x) - indices of k training points closest to x

y^i *is labels* ± 1

As the measure of nearness/distance, we used 'minkowski','mahalanobis','manhattan'. Also, various nearest neighbours were used.

2. SVM - Kernel:

During data exploration, we decided to use models that can classify non-linearly separable data as there didn't seem to be any linear boundaries. Hence, the first model we decided to use was Kernelized SVM and tried various kernels like RBF and Polynomial. Instead of constructing a higher dimensional feature map, the kernel functions take inner products to achieve the same results.

$$Max_{\alpha} L(w, b, \alpha) = \sum_{i}^{m} \alpha_i - \frac{1}{2}\sum_{i,j}^{m} \alpha_i \alpha_j y^i y^j \; k(x^i, x^j)$$

Subject to:

$\alpha_i \geq 0, \quad i = 1,.., m$

$\sum_{i}^{m} \alpha_i y^i = 0$

α_i *is the lagrangian multiplier*

$$polynomial\ kernel\ k(x, y) = (x^T y)^d \; ; \; Gaussian\ RBF\ K(x, y) = exp\left(- \frac{||x-y||^2}{2\sigma^2}\right)$$

Hyper-Parameters: We varied the degree of polynomial and variance of RBF to find the optimal kernel function. We also varied the Langrarian multiplier. We used a 10-fold CV to find the best value of degree, gamma(variance of RBF) and C (lagrangian multiplier).

3. Decision Tree Classifier (CART):

CART is a non-parametric algorithm and can provide better performance whenever data does not have a specific distribution. Our data here looked as if there is no specific distribution for each class. Hence, I decided to train a classification tree on this data. Moreover, this classification tree could also act as a weak classifier for Bagging and Boosting approaches later.

$For\ Classification$: $\widehat{p_{jk}} = \frac{1}{N_j} \sum_{x \in R_j} \prod \left(y_i = k\right)$ $where\ k(j) = arg\ max_k \widehat{p_{jk}}$

$For\ Pruning$: $C_\alpha(J) = \sum_{j=1}^{|J|} N_j Q_j + \alpha|J|$

$N_j Q_j$ is a measure of Error. We are using Gini Index
α is the $lagrangian\ multiplier$
j - Region or splitting variable and k - class

We select maximum depth and Alpha($lagrangian\ multiplier$) using 10 fold CV score

4. Random Forest (Bagging):

Since the decision tree showed signs of high variance (high difference between test score and CV score) hence, we decided to use Random Forest on classification tree (CART). Random forest helps in reducing the variance of the classifier. Trees are grown on bootstrapped sample of the data i.e.

- v variables are selected out of total p variables such that v<p
- Only partial data points are used with replacement (i.e some points can be used more than once)

Majority vote of the leaf node is taken to classify the data points. Bagging helps reduce the overall variance of the classifier.

If there are B random variables, each with $\sigma^2\ variance$, and ρ represents correlation between them, then

overall variance $= \rho\,\sigma^2 + \frac{(1-\rho)}{B}\sigma^2$

We kept increasing the number of trees till the accuracy stabilized.

5. Adaboost (using CART):

The decision tree was already overfitting and hence had high bias. So, using Boosting may not have been such a good approach. However, out of curiosity we still tried fitting Adaboost on the trained decision tree. We kept increasing the number of iterations and plotted the CV score and test score.

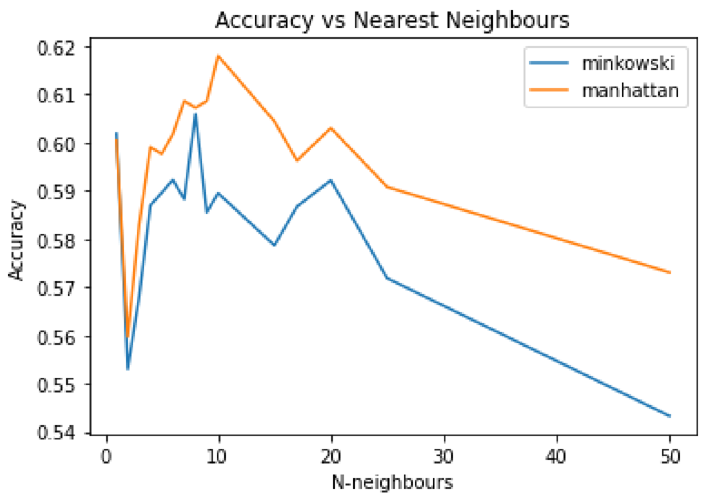
Experiments and Results

For experiments, the data is split into 75% training and 25% test set. The hyper-parameters of all models are tuned using 10-fold cross validation using training data. The accuracy of the model, confusion matrix and classification report is reported using the test data.

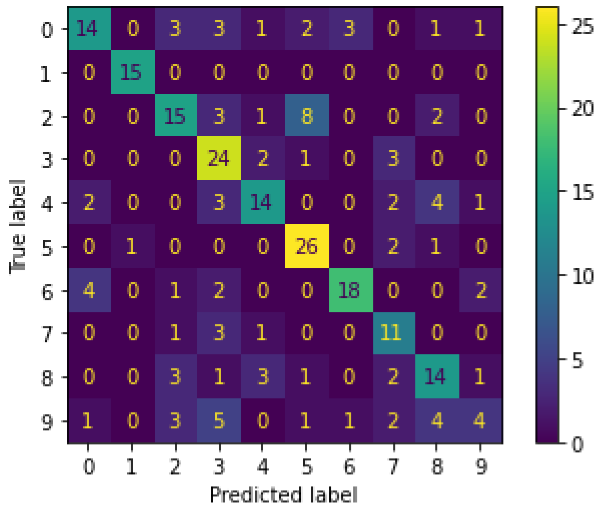
1. KNN

Based on the 10- fold cross validation, we found the distance metric 'manhattan' and n_neighbors as 10 to be the most accurate. The accuracy of KNN on the test dataset is 63%. The accuracy has been mainly impacted by higher misclassification in 'rock' music. This accuracy is in-line with 61% found in the case of the Paper[1]. Reason for our accuracy to be slightly better than Paper[1] could be a) removal of outliers and b) possibly randomness.

Hyper-Parameter tuning



Confusion matrix



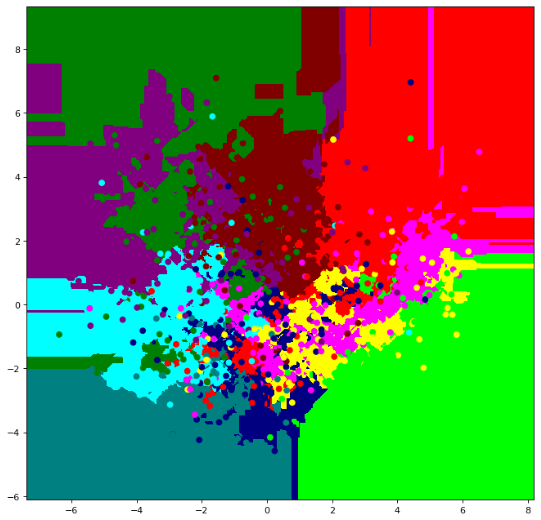
Classification report

```
              precision    recall  f1-score   support

         1.0       0.67      0.50      0.57        28
         2.0       0.94      1.00      0.97        15
         3.0       0.58      0.52      0.55        29
         4.0       0.55      0.80      0.65        30
         5.0       0.64      0.54      0.58        26
         6.0       0.67      0.87      0.75        30
         7.0       0.82      0.67      0.73        27
         8.0       0.50      0.69      0.58        16
         9.0       0.54      0.56      0.55        25
        10.0       0.44      0.19      0.27        21

    accuracy                           0.63       247
   macro avg       0.63      0.63      0.62       247
weighted avg       0.63      0.63      0.62       247
```

KNN decision boundary in 2D



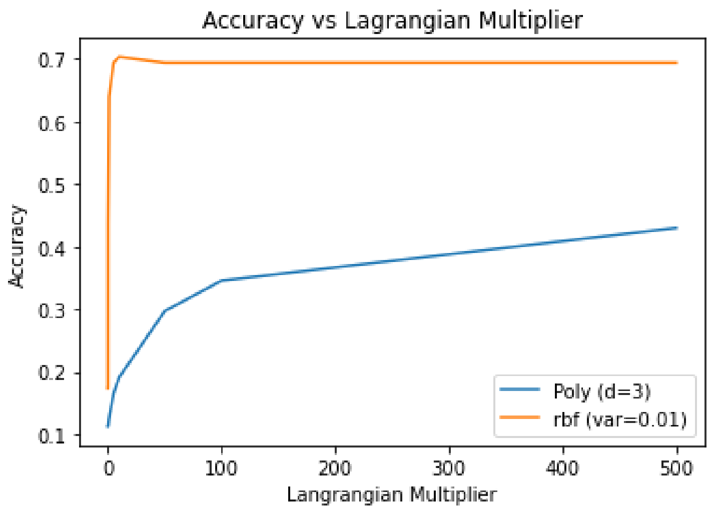
Classification Classes - 1:Blue, 2:classical, 3:country, 4:disco, 5:hip hop, 6:jazz, 7:metal, 8:pop, 9:reggae, 10:rock

As expected, the projection of the decision boundary of KNN in 2D looks somewhat like the Voronoi structure.

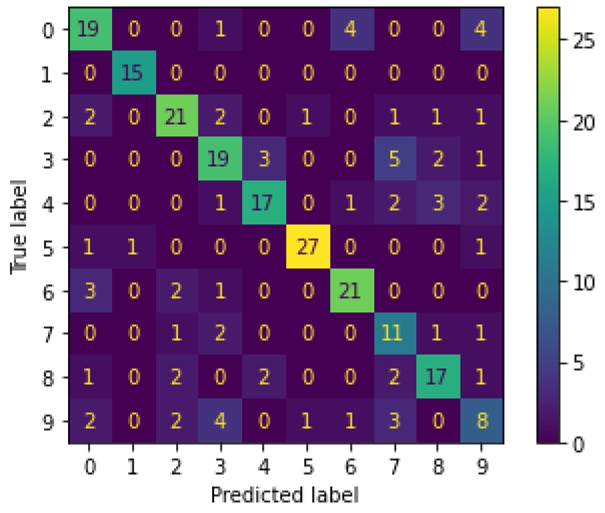
2. SVM - Kernel:

The 10-fold CV score of SVM-rbf shows better accuracy than SVM-poly (degree=3). We tried various values of Langrangian multiplier and gamma(variance of rbf) and concluded that SVM-rbf with gamma=0.01 and C=10 is the best model. The accuracy of the model on test data was 71% which is way better than KNN. Also, the CV score and test score are nearly equal suggesting that there is no overfitting of the model. SVM-rf model improved f1-score of each class and mainly class 10 (rock). The improved accuracy in SVM-rf suggests that data is indeed non-linearly separable. Additionally, one reason why SVM may have delivered a good score is removal of the outliers from data as SVM is very sensitive to outliers

Hyper-Parameter tuning



Confusion matrix



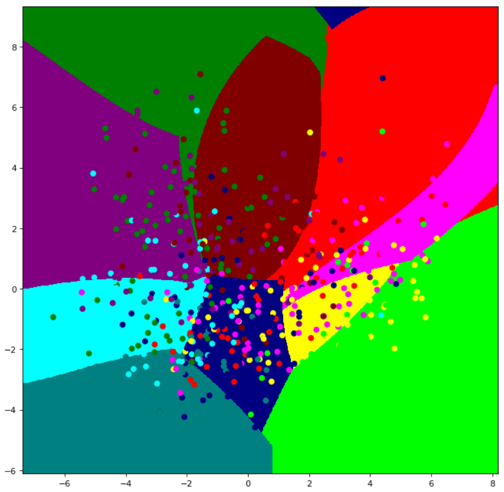
Classification report

```
              precision    recall  f1-score   support

         1.0       0.68      0.68      0.68        28
         2.0       0.94      1.00      0.97        15
         3.0       0.75      0.72      0.74        29
         4.0       0.63      0.63      0.63        30
         5.0       0.77      0.65      0.71        26
         6.0       0.93      0.90      0.92        30
         7.0       0.78      0.78      0.78        27
         8.0       0.46      0.69      0.55        16
         9.0       0.71      0.68      0.69        25
        10.0       0.42      0.38      0.40        21

    accuracy                           0.71       247
   macro avg       0.71      0.71      0.71       247
weighted avg       0.72      0.71      0.71       247
```

SVM-rbf decision boundary in 2D

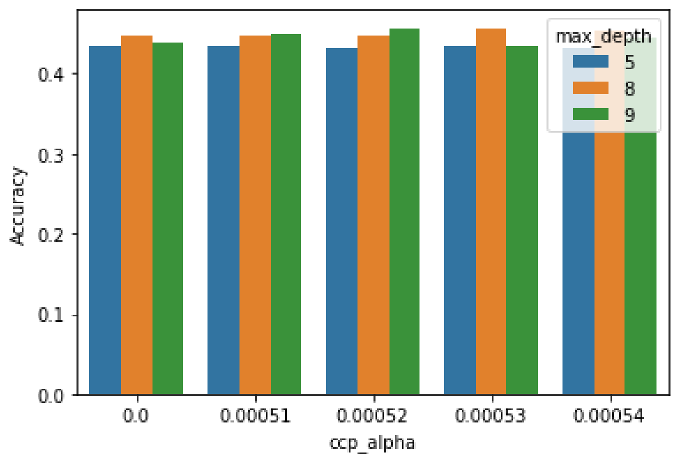


Classification Classes - 1:Blue, 2:classical, 3:country, 4:disco, 5:hip hop, 6:jazz, 7:metal, 8:pop, 9:reggae, 10:rock

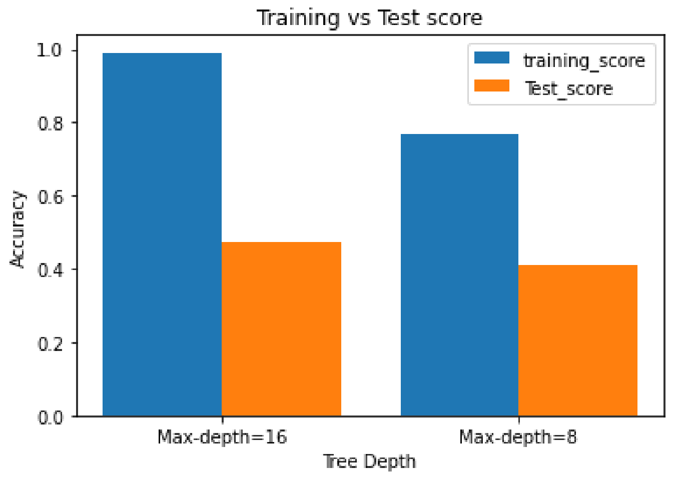
3. Decision tree (CART)

Given that data seems to be non-linearly separable and it may not have any clear distribution, it seemed logical to use the classification tree model. At max depth of 16, the training score was 99% and test score was 45% which clearly shows that there is overfitting of model on the data. Hence, we started tuning the regularization parameter (alpha) and maximum depth using the 10 fold CV. We found that at max-depth of 8 and alpha of 0.00053 the model produced the best CV score. At this point the training score came down to 77% and test score declined to 42% suggesting that the model still significantly overfits. To overcome this variance (overfitting), the next logical step was to try and use the bagging approach.

Hyper-Parameter tuning



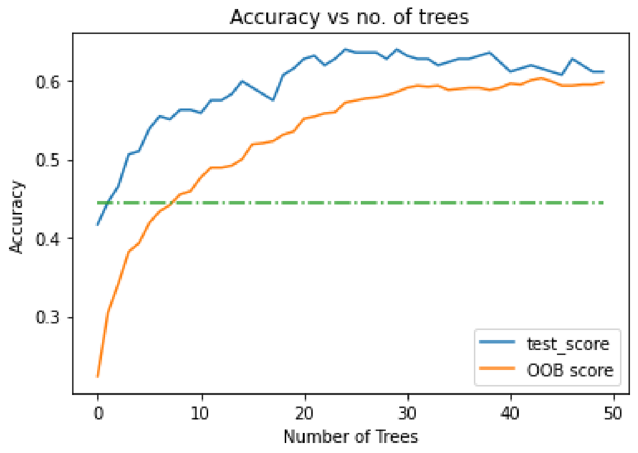
Clear signs of Overfitting



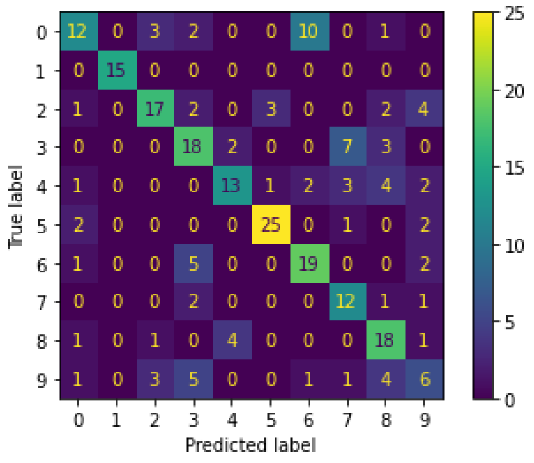
4. Bagging - Random Forest

As the CART model was overfitting the data which meant that the model had high variance. Hence, we decided to use a random forest approach. Random forest approach helps reduce the variance of a classifier and improves its test accuracy.

Hyper-Parameter tuning



Confusion matrix



We decided to use out-of-bag (OOB) score as the measure of cross validation score. The OBB score started stabilizing after the number of trees increased to more than 40. Moreover, the OBB score came closer to the test score indicating that there is no overfitting of the model on the data. The Accuracy of the Random forest model stabilized at 63% on test data.

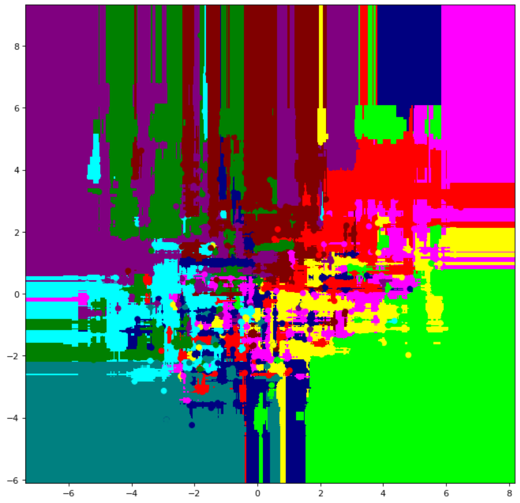
Classification report

```
              precision    recall  f1-score   support

         1.0       0.63      0.43      0.51        28
         2.0       1.00      1.00      1.00        15
         3.0       0.71      0.59      0.64        29
         4.0       0.53      0.60      0.56        30
         5.0       0.68      0.50      0.58        26
         6.0       0.86      0.83      0.85        30
         7.0       0.59      0.70      0.64        27
         8.0       0.50      0.75      0.60        16
         9.0       0.55      0.72      0.62        25
        10.0       0.33      0.29      0.31        21

    accuracy                           0.63       247
   macro avg       0.64      0.64      0.63       247
weighted avg       0.64      0.63      0.62       247
```

Random Forest decision boundary in 2D

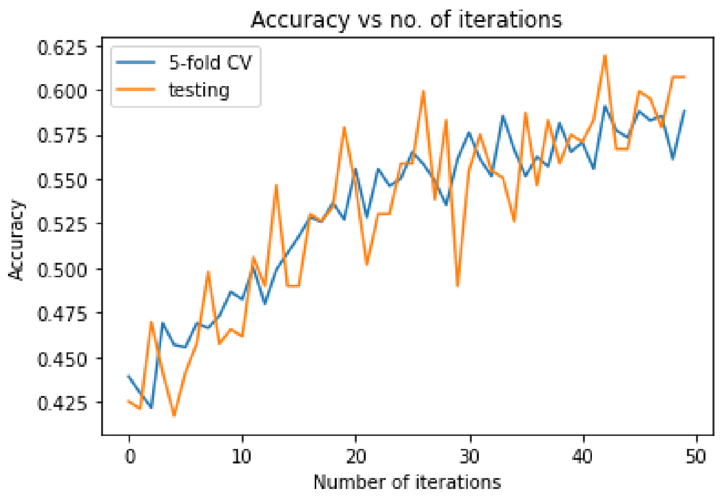


Classification Classes - 1:Blue, 2:classical, 3:country, 4:disco, 5:hip hop, 6:jazz, 7:metal, 8:pop, 9:reggae, 10:rock

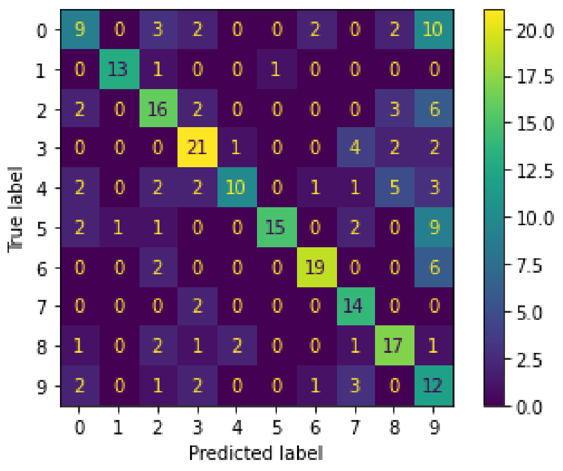
5. Boosting- Adaboost

The CART model that we trained above had high variance, so using a boosting method may not be the best approach. However, out of curiosity we tried to use boosting. We tried boosting performance at various iterations. Even though the performance improved from 42% level to around 60%, there were wide fluctuations between CV score and test score. This could be an indication of overfitting of the model.

Hyper-Parameter tuning



Confusion matrix



	0	1	2	3	4	5	6	7	8	9
0	9	0	3	2	0	0	2	0	2	10
1	0	13	1	0	0	1	0	0	0	0
2	2	0	16	2	0	0	0	0	3	6
3	0	0	0	21	1	0	0	4	2	2
4	2	0	2	2	10	0	1	1	5	3
5	2	1	1	0	0	15	0	2	0	9
6	0	0	2	0	0	0	19	0	0	6
7	0	0	0	2	0	0	0	14	0	0
8	1	0	2	1	2	0	0	1	17	1
9	2	0	1	2	0	0	1	3	0	12

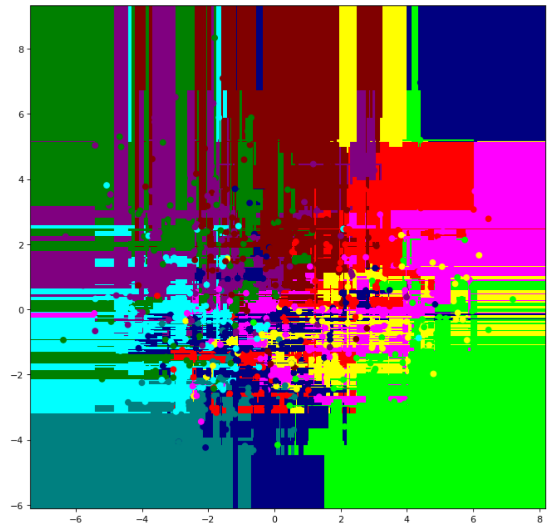
Classification report

```
              precision    recall  f1-score   support

         1.0       0.50      0.32      0.39        28
         2.0       0.93      0.87      0.90        15
         3.0       0.57      0.55      0.56        29
         4.0       0.66      0.70      0.68        30
         5.0       0.77      0.38      0.51        26
         6.0       0.94      0.50      0.65        30
         7.0       0.83      0.70      0.76        27
         8.0       0.56      0.88      0.68        16
         9.0       0.59      0.68      0.63        25
        10.0       0.24      0.57      0.34        21

    accuracy                           0.59       247
   macro avg       0.66      0.62      0.61       247
weighted avg       0.66      0.59      0.60       247
```

Adaboost decision boundary in 2D

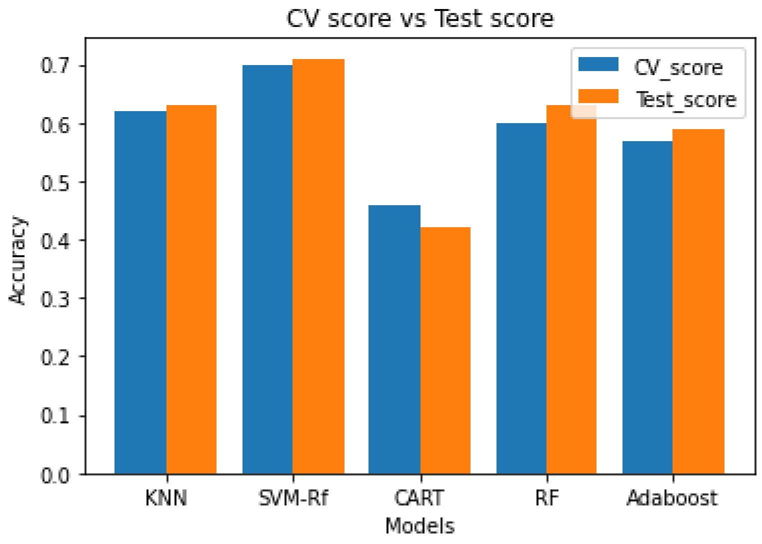


Classification Classes - 1:Blue, 2:classical, 3:country, 4:disco, 5:hip hop, 6:jazz, 7:metal, 8:pop, 9:reggae, 10:rock

Conclusion

In conclusion, we were able to figure out answers to our initial set of questions in problem statement:

1. The Audio features related to frequency domain are more relevant for the genre classification problem. We were able to extract features from the Audio data by using Short time fourier transform to create spectrogram and related features.
2. We were able to use PCA to visualize the data and it gave us a starting point to explore models. Moreover, we detected and removed outliers using LOF and carried out feature selection using Mutual Information.



3. Overall - SVM kernel model performs the best. Out of all the models trained, SVM-Kernel delivered the best performance. The performance was better than the Paper[1] KNN model that we set out to outperform. Two key reasons why we think SVM worked well was a) The data was non-linearly separable and the kernel trick helped and b) The removal of outliers may have significantly helped as SVM is in general very sensitive to outliers.

Future work

We highlight below various points that could be explored (given more time):

1) Experiment with feature length: In the Paper[8], the authors found that changing the length of audio files extracts produces different levels of accuracy. In our dataset, the audio files are 30seconds long. It could be interesting to see if the accuracy of the models could be improved if we select different segment lengths like say 10 sec, 20 sec, etc.
2) Using additional features: In all the models, one common source of error was misclassification of rock music. We could explore some more features that could help classify rock music more accurately. This could help improve accuracy of models overall.
3) Using PCA on Spectrogram images: Similar to a face detection, we could try to make an Eigenvector for each class using the spectrogram images of audio files. For every test image, we can then calculate a projection residual using the formulae
   Projection Residuals = test image - Projection of test image on space spanned by eigenvectors of the class.
   When we project the test image on eigenvectors of different classes, we can say that the test image is of the class with lowest projection residuals.
4) Exploring Neural networks Models: It would be interesting to see how Neural network based algorithms will perform. We could feed the Spectrogram images directly to the algorithms which could potentially improve their accuracy.

Reference

[1] Musical Genre Classification of Audio Signals by George Tzanetakis
[2] An Industrial-Strength Audio Search Algorithm by Avery Li-Chun Wang

[3] Coursera: Audio Signal Processing for Music Applications by Prof Xavier Serra and Prof Julius Smith (Link)
[4] Center for Computer Research in Music and Acoustic by Prof Julis Smith (Stanford) (Link)
[5] Spectral leakage and windowing by MIL University of Florida (Link)

[6] Short Time Fourier Transform Based Music Genre Classification by Ahmet Elbir, Hamza Osman Ihan
[7] LOF: Identifying Density-Based Local Outliers by Markus M. Breunig et all (Link)

[8] Aggregate features and ADABOOST for music classification James Bergstra, Norman Casagrande ,Dumitru Erhan, Douglas Eck and Balazs Kegl (2006)

[9] AN EVALUATION OF AUDIO FEATURE EXTRACTION TOOLBOXES by David Moffat, David Ronan, Joshua D. Reiss (2015)