Machine Learning Engineer Nanodegree

Capstone Project

Music Genre Classification

Prakhar Kumar

20th April, 2019

Project Overview:

A Music Genre is a conventional category that identifies some pieces of music. Music can be divided into genres in many different ways. The artistic nature of music means that the classifications are subjective and some genres may overlap with one another. Some of the popular music genres are: hip-hop, rock, country etc.

It is very difficult to put a music into one category of music as it may have parts which belong to another category (genres) of music. There are differences between genres of music like hip-hop and heavy metal. Heavy metal music generally uses more bass and high-pitched sounds of musical instruments while hip-hop mainly focuses on the vocals.

Grouping the music into one genre is important because each individual is different as has a different taste in music. By grouping, we can allow users to listen to the music which they like.

Link to dataset: http://marsyas.info/downloads/datasets.html

Previous research: https://cse.iitk.ac.in/users/cs365/2015/_submissions/archit/report.pdf

Problem Statement:

The problem is to develop an ML application which can classify a music into different genres.

The audio files obtained cannot be directly used with the Machine Learning techniques, we will first have to convert them. The techniques used to convert are FFT (Fast Fourier Transformations) and MFCC (Mel-Frequency Cepstral Coefficients). There are 10 genres of music, each genre will be given a number from 0 through 9. After having the data in the required format, we will train 3 machine learning models namely – Logistic Regression, Random Forest and XGBClassifier on both type of features (FFT and MFCC) using the train dataset. We will then evaluate the model on test dataset using confusion matrices and roc auc score.

The result will then be put up in a table for comparison.

Metrics:

The metrics used for the projects are – Roc Auc Score.

Since our dataset contains multiple classes auc is the best metric we could use. It is better than plain accuracy because; say our dataset has 2 classes for now and if class 1 is 90% and class 2 constitutes 10%, if my classifier predicts everything to be class 1 then I would achieve an accuracy of 90% but 90% doesn't tell the whole story in-fact our classifier is a dumb classifier but just looking at the accuracy obtained we would think that it did a good job but it did not. In cases like these AUC provides a better understanding of how well we are doing on each class whereas the accuracy tells the overall story, AUC tells the score on each class and in this way we can decide if our classifier is doing well or not.

To read more on roc auc scores:

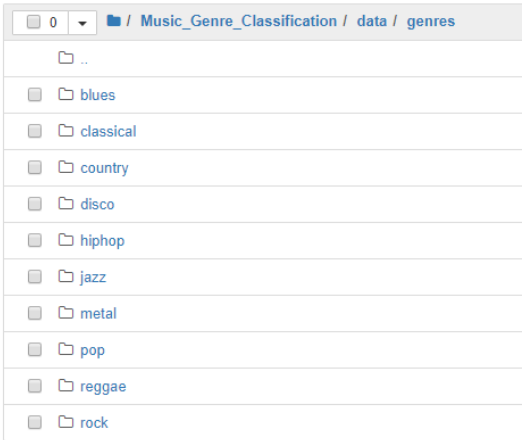
https://medium.com/greyatom/lets-learn-about-auc-roc-curve-4a94b4d88152

Data Exploration:

Since we are using audio data there isn't much that we can visualize except the frequencies and drawing a spectrogram.

Our Dataset has 1000 songs and 10 classes/genres. Each genre has 100 songs of 30 seconds each.

The following image shows all the genres available to us:



Each of the genres contain 100 songs from 1-100, these songs are in the form of .au file which will be converted to .wav format using the ffmpeg converter. The audio files in .wav format will then be used to generate FFT and MFCC features which will be used as training data.

Number of music files in each genre:

```
Genre  reggae  Contains : 100  Music Files
Genre  pop  Contains : 100  Music Files
Genre  classical  Contains : 100  Music Files
Genre  disco  Contains : 100  Music Files
Genre  jazz  Contains : 100  Music Files
Genre  metal  Contains : 100  Music Files
Genre  hiphop  Contains : 100  Music Files
Genre  country  Contains : 100  Music Files
Genre  rock  Contains : 100  Music Files
Genre  blues  Contains : 100  Music Files
```

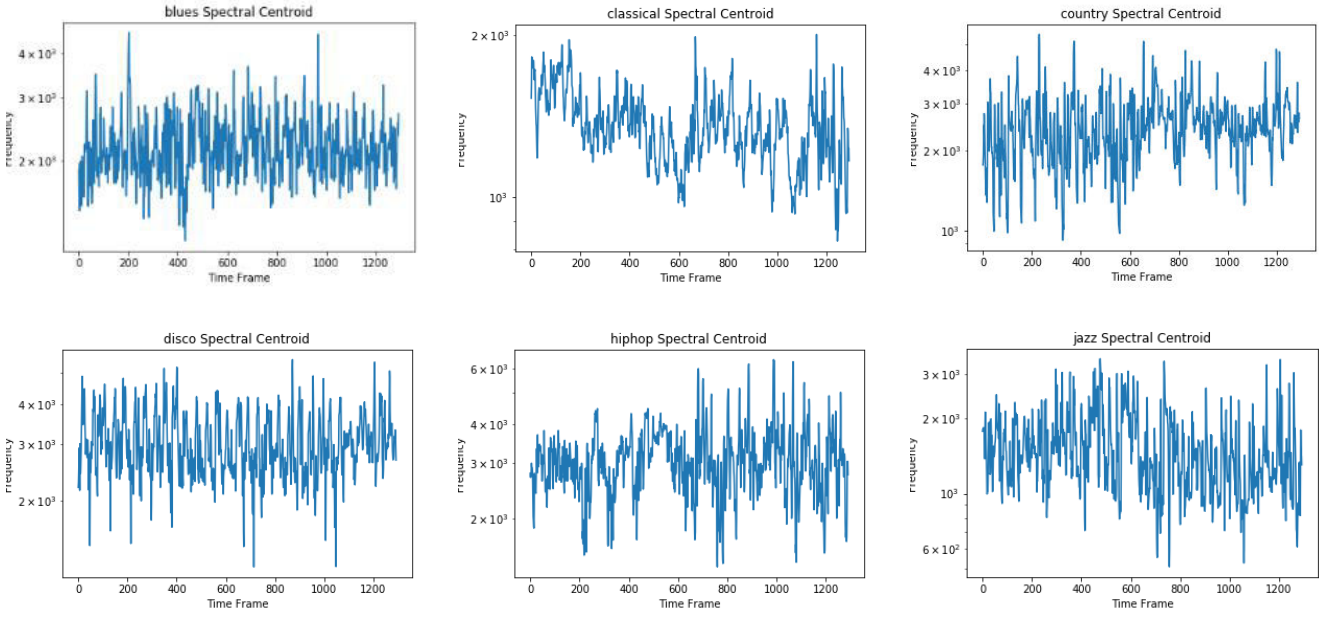
Sampling Rate of each genre:

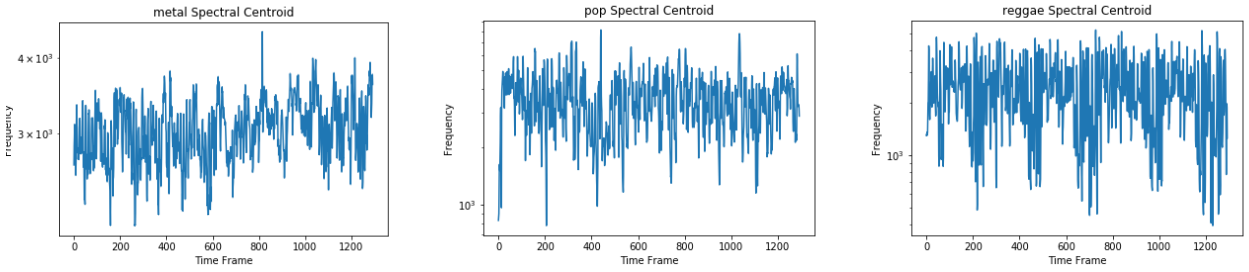
```
Sampling Rate of  reggae  Genre : 22050
Sampling Rate of  pop  Genre : 22050
Sampling Rate of  classical  Genre : 22050
Sampling Rate of  disco  Genre : 22050
Sampling Rate of  jazz  Genre : 22050
Sampling Rate of  metal  Genre : 22050
Sampling Rate of  hiphop  Genre : 22050
Sampling Rate of  country  Genre : 22050
Sampling Rate of  rock  Genre : 22050
Sampling Rate of  blues  Genre : 22050
```

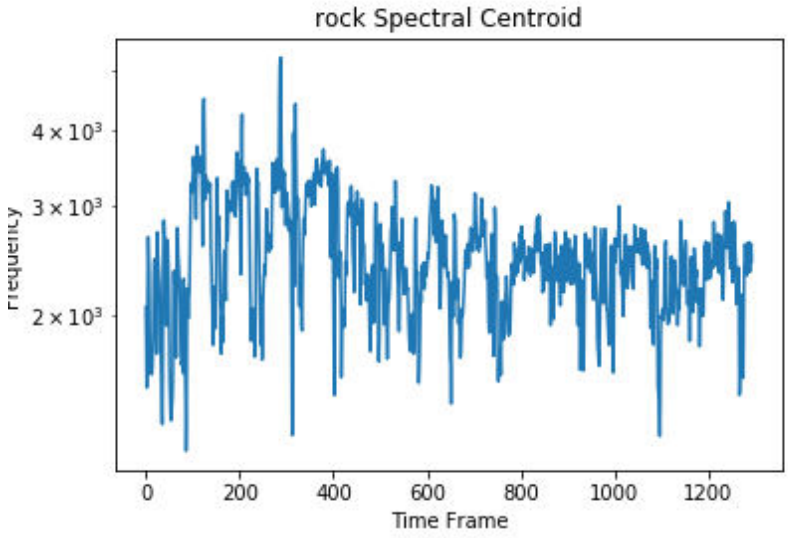
Spectral Centroid basically finds the average frequency in a bin/ time frame. It was calculated using the librosa library. We have used the default values of bins/hop-counts.

https://librosa.github.io/librosa/generated/librosa.feature.spectral_centroid.html

Spectral Centroid of each genre:





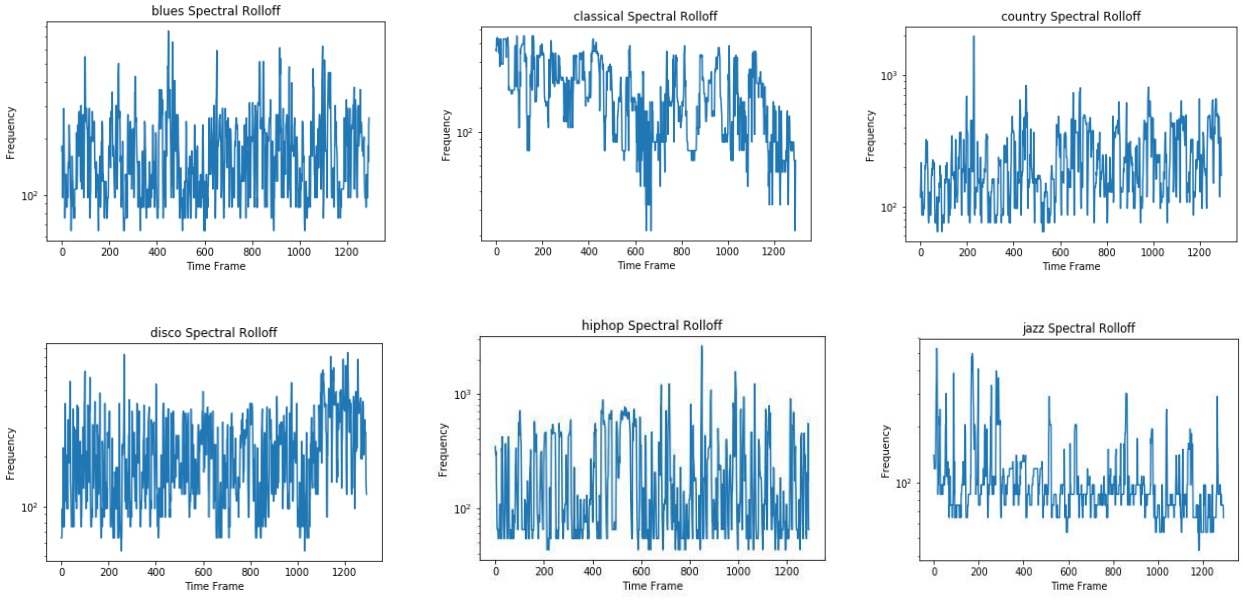


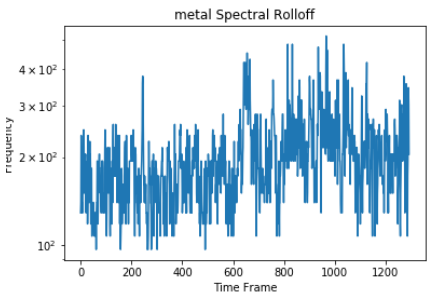
Hip hop seems to have higher average frequencies than other genres and classical and reggae genres seem to have lowest average frequencies.

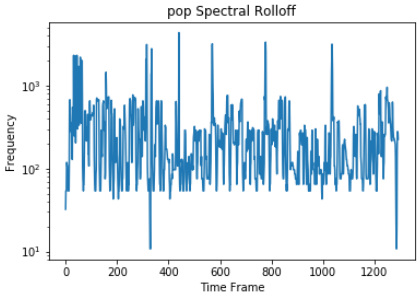
Spectral roll-off can be used to calculate the minimum frequencies in a timeframe. Again, we have used the librosa library to get the spectral roll off with roll percent parameter set to 0.10 to get lowest frequencies.

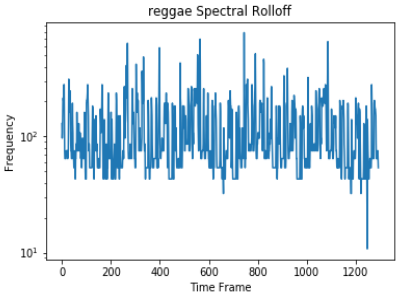
https://librosa.github.io/librosa/generated/librosa.feature.spectral_rolloff.html

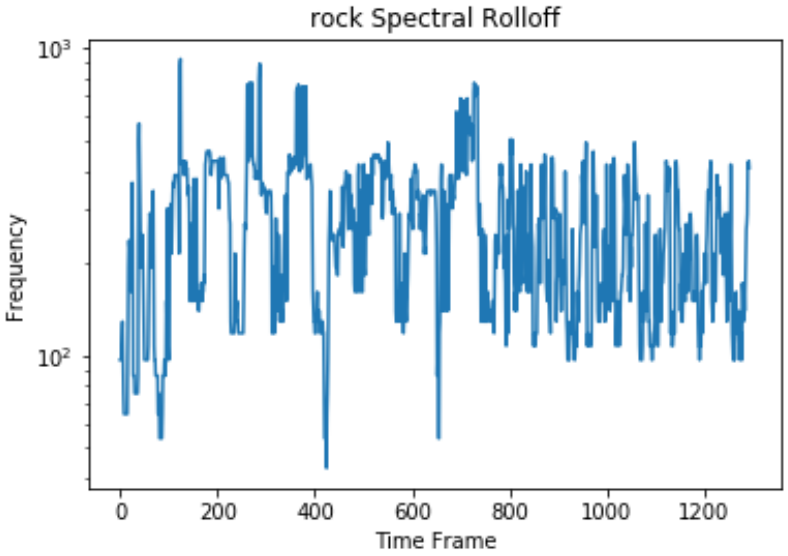
Spectral Roll off of each genre:











Metal and Pop Music seem to have higher minimum frequencies over some of the regions of the time frame.

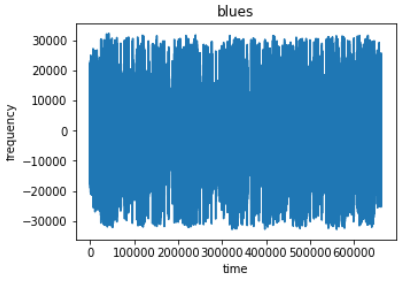
Data Visualization:

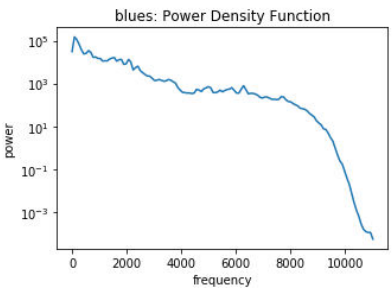
I have plotted spectrogram, frequency and power density for each of 10 classes:

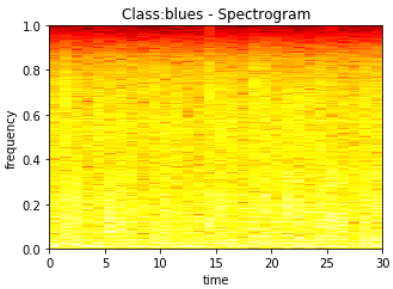
More on spectral power density: https://en.wikipedia.org/wiki/Spectral_density

More on Spectrogram: https://en.wikipedia.org/wiki/Spectrogram

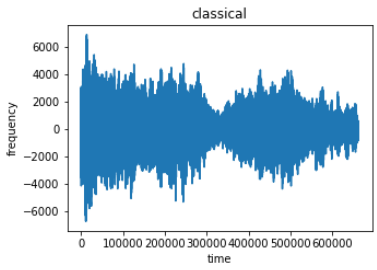
Blues Genres Images:

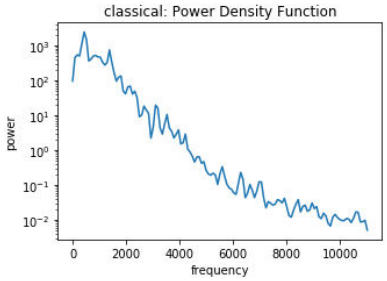


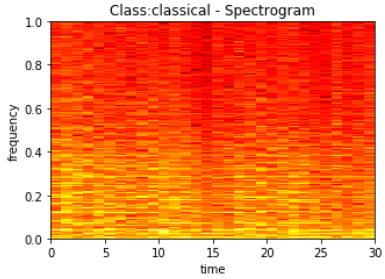




Classical Genre Images:

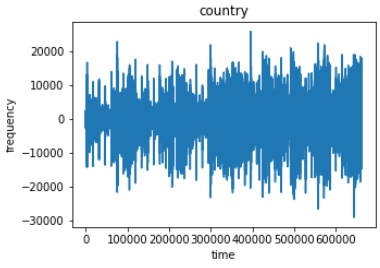


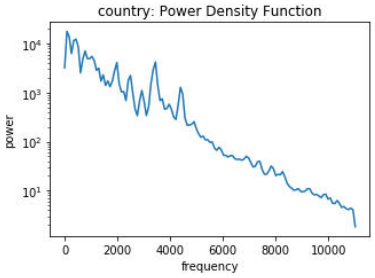


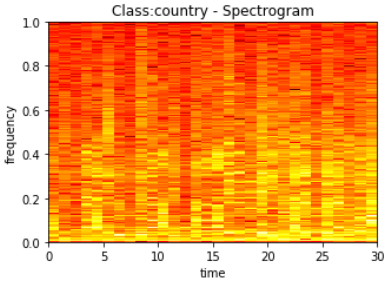


Just by looking at the 3 images above of the two genres we see that the two genres can be easily seen as dissimilar as frequency and spectral power density is a lot different. From this we can conclude that any ML model should be able to distinguish between these two easily, we can check this through confusion matrix later.

Country Genre:

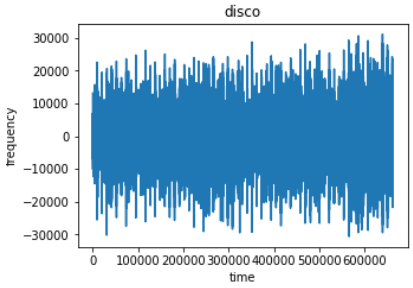


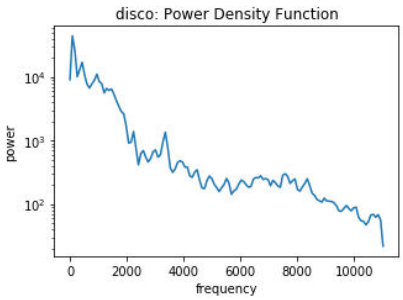


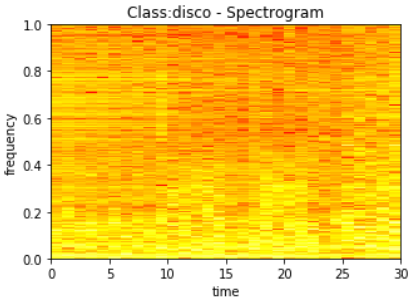


Country music seems to be a bit similar to our classical music so we might have some difficulties in classifying between these two.

Disco Genre:

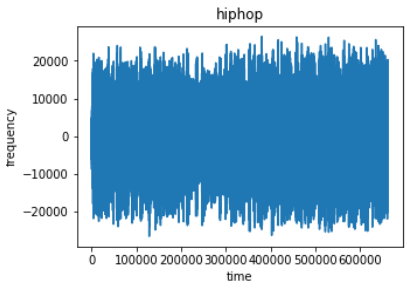


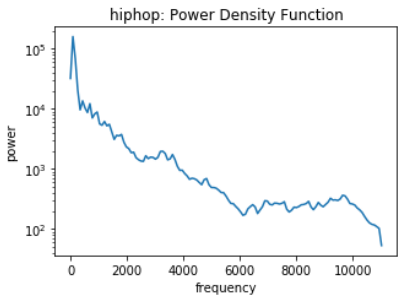


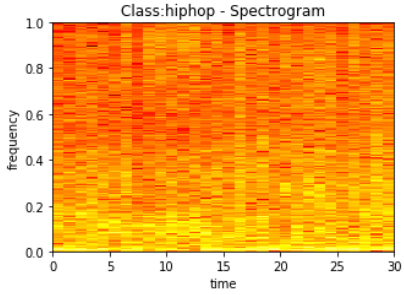


Disco Genre seems a bit like blues genre but the spectrogram shows some difference between the two.

Hip-hop Genre:

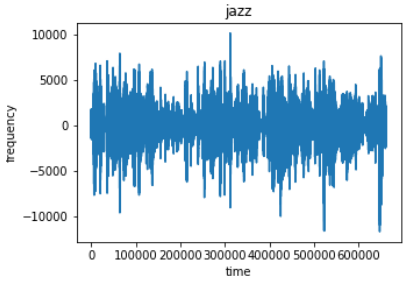


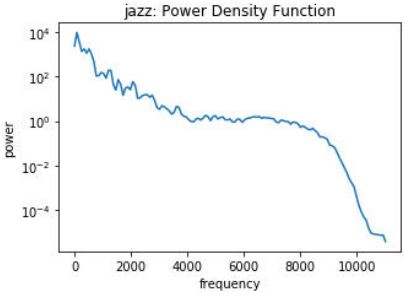


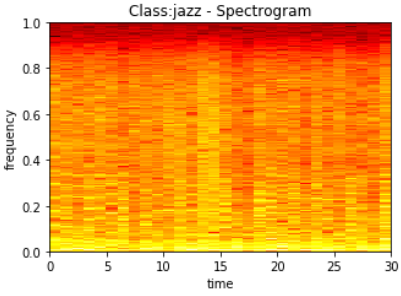


Hip-hop music has some similarities with all genres seen so far so this might be most difficult one to classify thus far.

Jazz Genre:

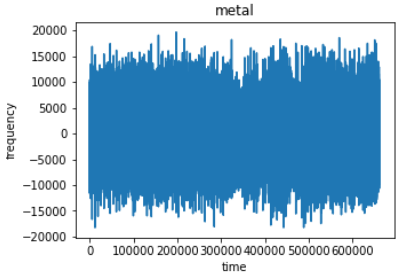


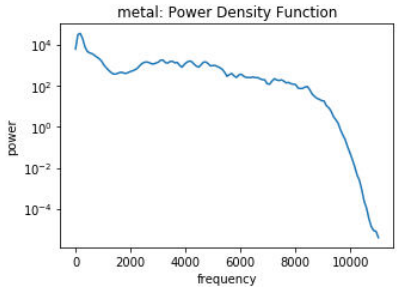


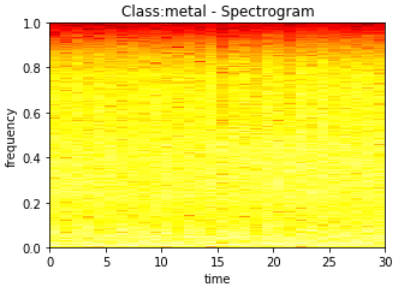


Jazz seems a lot like blues but the frequency is different so we can separate them.

Metal Genre:

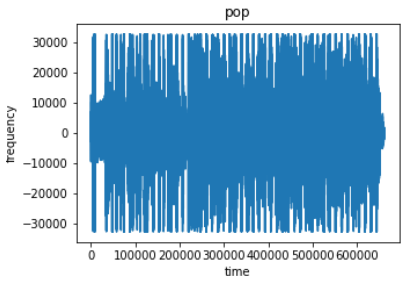


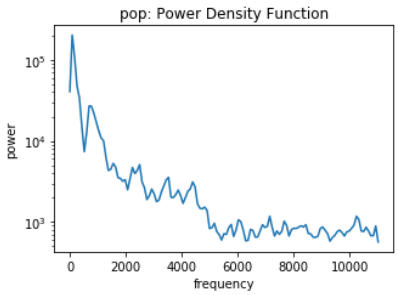


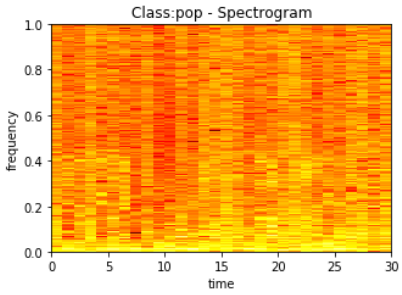


Metal also resembles blues highly; we can expect some difficulties in classifying between the two.

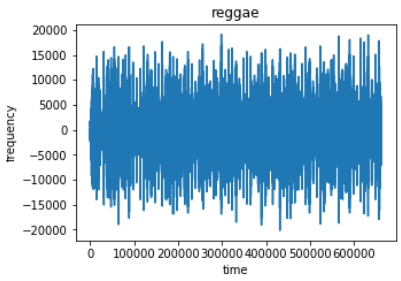
Pop Genre:

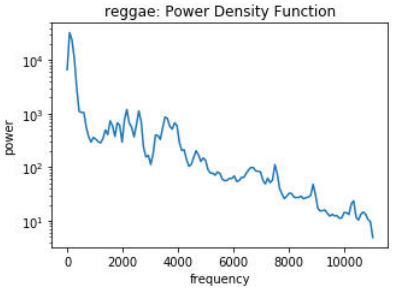


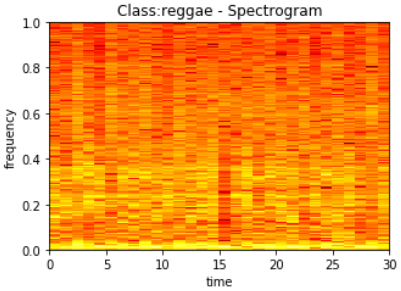




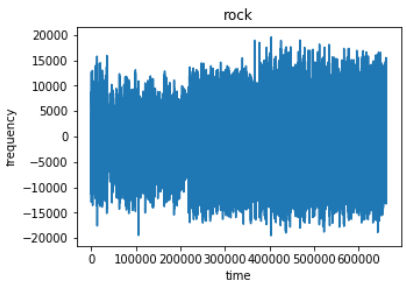
Reggae Genre:

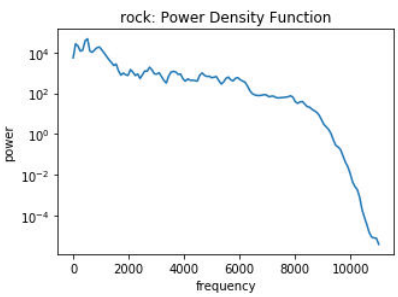


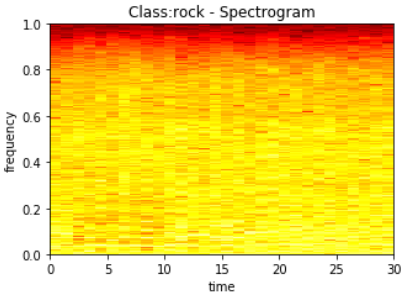




Rock Genre:







I have looked at all the genres and can say that:

1. Out of all the genres so far; classical, jazz and pop genres should be easily separable.
2. Hip-hop, country and disco Genres might be the most difficult ones to classify as these have many similarities with other genres present.

Algorithms/Techniques:

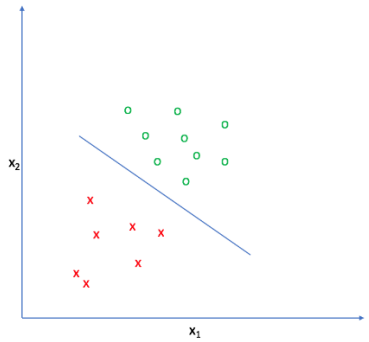
The Machine Learning Techniques used in the project are- Logistic Regression, Random Forest and XgBoost Classifier.

Logistic regression:

Logistic Regression is one of the simplest Classification techniques out there. Logistic Regression tries to find a linear decision boundary between the two classes.

Finding a linear decision boundary is easier when the dimensions of the data is high.

A simple Decision Boundary.



In above case everything above the blue line is class green and below it is class red. We can extend LR for multiple classes using the one-vs-rest strategy. The one-vs-rest strategy is discussed in detail here - https://utkuufuk.github.io/2018/06/03/one-vs-all-classification/

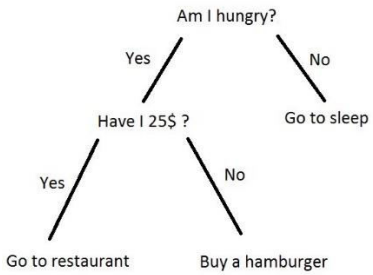
Pros: It is a very simple and intuitive model. It is very fast to train.

Cons: Being simple, it may capture all the complex relationships that exists within the data.

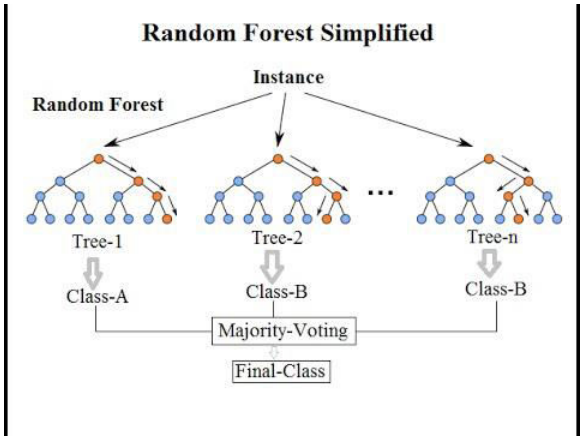
I have used it because LR generally forms a good baseline and works well with higher dimensional data.

Random Forest:

The random forest method combines multiple decision trees together. A decision tree represents the intelligence in the form decisions or from programming point of view if-else like structure. A sample decision tree is shown below.



Above denotes a very simple decision tree. Random Forests are just a combinates of such decision trees and the output of random forest is majority voting applied on the outputs of all decision trees that make up the forest. A very simplified version is:



Above two trees say that the sample/instance belongs to class B while one says it belongs to class A, so the result is that sample belongs to class B as most of the trees predict it to be class B.

Pros: Reduction in overfitting due to multiple trees and this kind of model also has lesser variance. In almost all the cases performs better than Decision Trees. It also does not require feature scaling.

Cons: Complexity, Time consuming and not intuitive as it is an ensemble model.

I chose random forest over decision trees due to its advantages over DT's. Also, the data is high dimensional so we can multiple trees trained on different sets of features and combine all the weak learners to get a better result. Also, being an ensemble model, it generally gives better results than others.

XgBoost Classifier

The XgBoost Classifier is also a tree-based method used for classification. It also builds multiple trees just like the random forest but unlike in random forest where each tree is built independent of one another; here the trees are built sequentially i.e. the next tree that follows tries to reduce the error of the previous tree.

For more details on this: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/

Pros: Being an ensemble model, it generally gives better results. We also have no need of doing feature scaling. It can be used to get feature importance's.

Cons: Harder to tune than other models as the number of parameters to tune are large. Not interpretable as it is an ensemble model. Time consuming.

I chose this because XgBoost model generally perform well than other models including Random Forests. Also, XgBoost works well when the dimensions are not very high like our dataset so we can have multiple trees learn on different sets of features and build upon previous trees to improve the performance.

Benchmark:

To get the benchmark result/threshold, FFT features (top 1000) were used. 3 Models were trained namely – Logistic Regression, Random Forest and XgBoost.

FFT features were obtained using the scipy library and default settings for scipy.fft()

Parameters to Tune:

All the models that I have trained are tuned. The parameters that I have used to tune for LR, RF and XgBoost are:

Model	**Parameters Tuned**
Logistic Regression (LR)	C, penalty
Random Forest (RF)	N_estimators, max_depth
XgBoost Classifier	N_estimators, max_depth

To get to know what these parameters mean: https://scikit-learn.org/stable/documentation.html

The table in Refinement Section shows the optimal value found during training for each of the models.

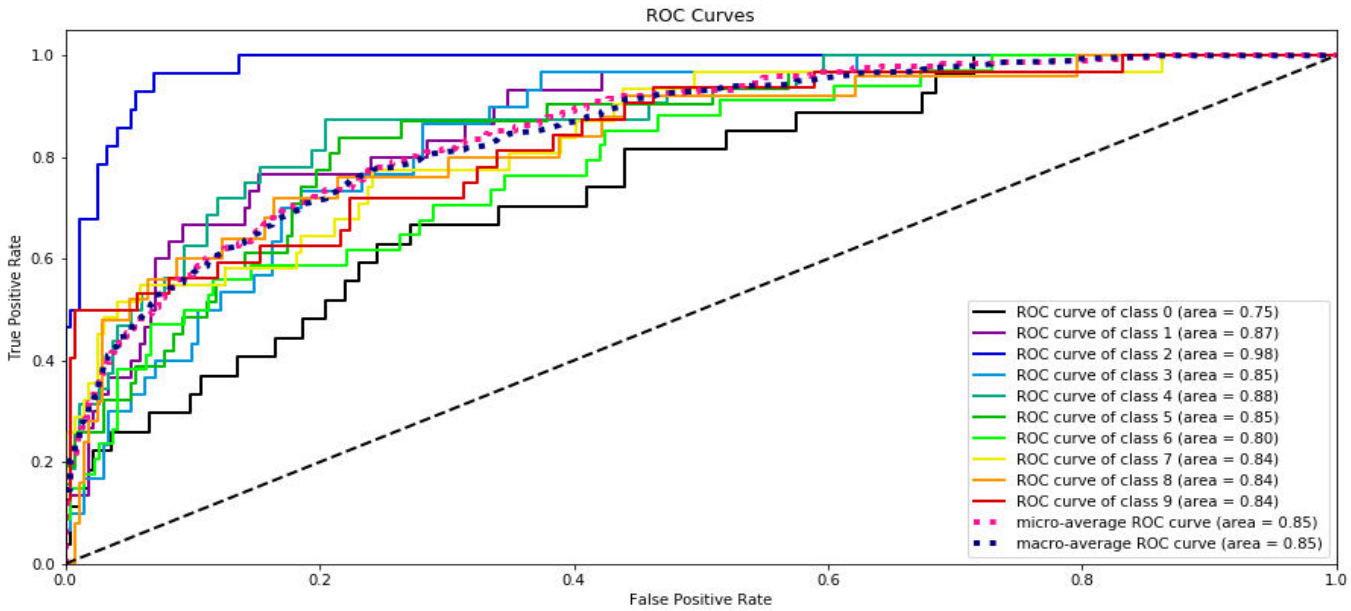
The roc auc score for 3 models are:

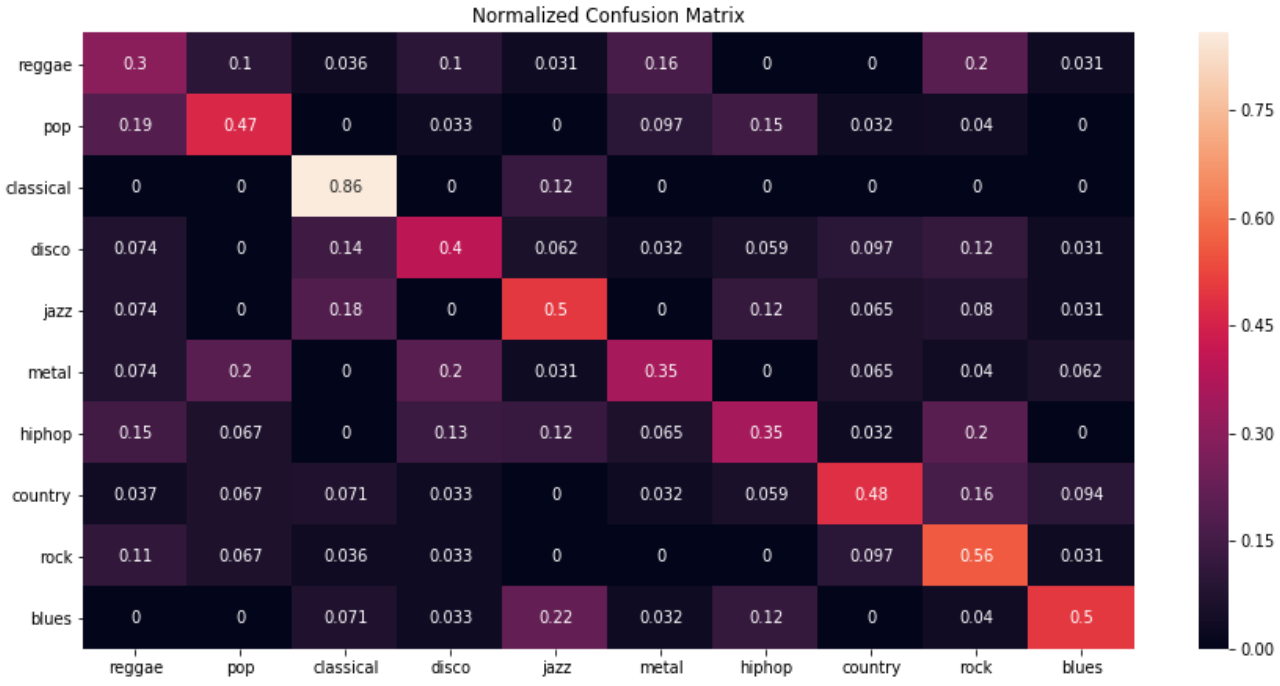
1. Logistic regression: 0.68 (micro-averaged), 0.66 (macro-averaged)
2. Random Forest: 0.82 (micro-averaged), 0.81 (macro-averaged)
3. XgBoost Classifier: 0.85 (micro and macro averaged).

So, the benchmark model taken is XgBoost and we have to get a better result than the ones obtained here.

XgBoost tuned parameter values: N_estimators: 300 and max_depth: 7, rest all are default.

For XgBoost benchmark the following images show the performance:





We see that the benchmark model is able to classify classical music very well classifying 86% of times correctly. This was expected as we saw in our visualization that the classical music is very different from the rest but it does not do a very good job on classifying metal/hip-hop/reggae music which was also expected.

Note: This result is on test data.

Methodology:

1. Data Pre-processing:

   The audio files in the data given is in a format which cannot be read. We first convert the audio files to .wav format using the ffmpeg audio converter. The file 1_Music genre Classification using the subprocess library and converts each audio file to .wav format. The ffmpeg audio converter has to installed if not already present and the source tar file is to be present inside the directory data/

2. Data Visualization:

   The 2nd ipython notebook is used to plot the spectral density, frequency and spectrogram for each of the 10 classes.

3. FFT Features/Benchmark model:

   The .wav files are used to generate FFT features. The features are then used to train 3 models LR, RF and XgBoost Classifier. Hyperparameter tuning was done to obtained the best benchmark model we could possibly get. The 3rd Notebook has all the

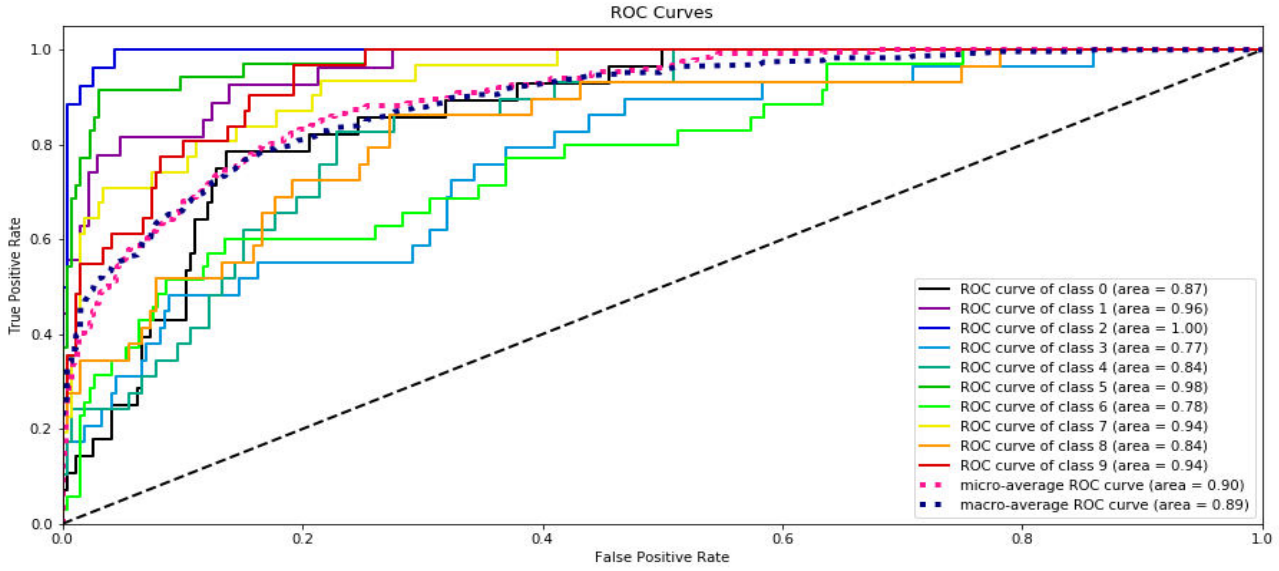
visualization related to this for comparing what was the best result obtained. XgBoost performed the best getting a roc auc score of 0.85.

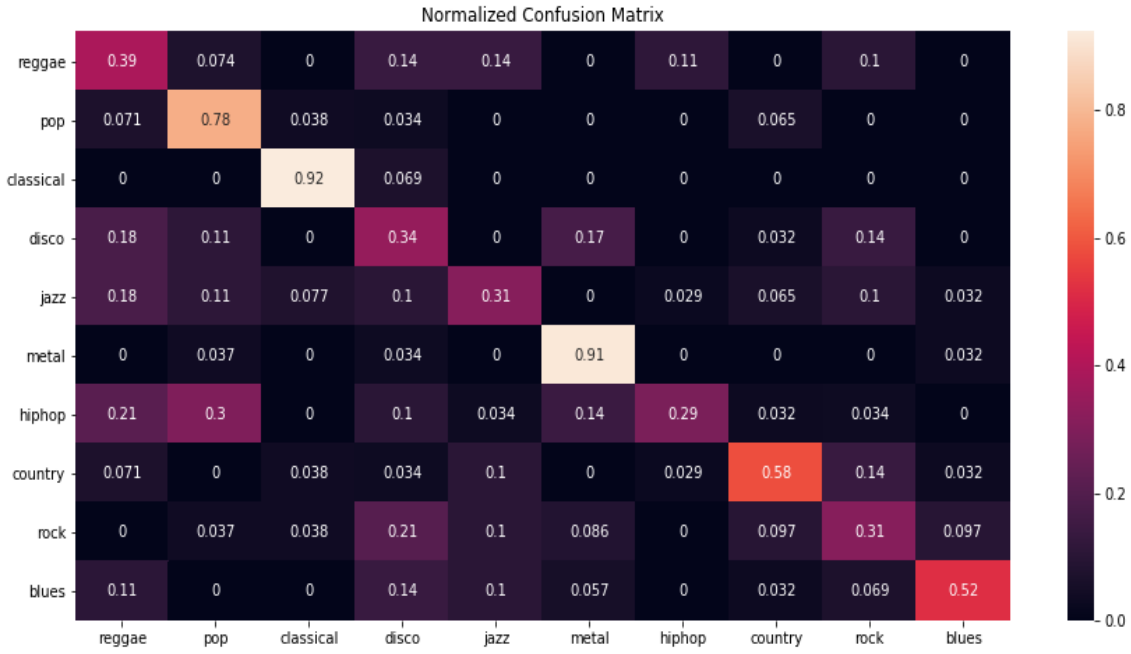
4. MFCC Features:

Using the features extraction techniques created by domain specialists, MFCC features were obtained. This was our first step to try and improve on the benchmark results. Here also 3 models were trained. Random forests were able to achieve a better score than other 2 tried and also all 3 beat the benchmark results.

MFCC Features were extracted using the python_speech_features library and default settings were used.

Following images show the result of our RF model trained on MFCC Features:

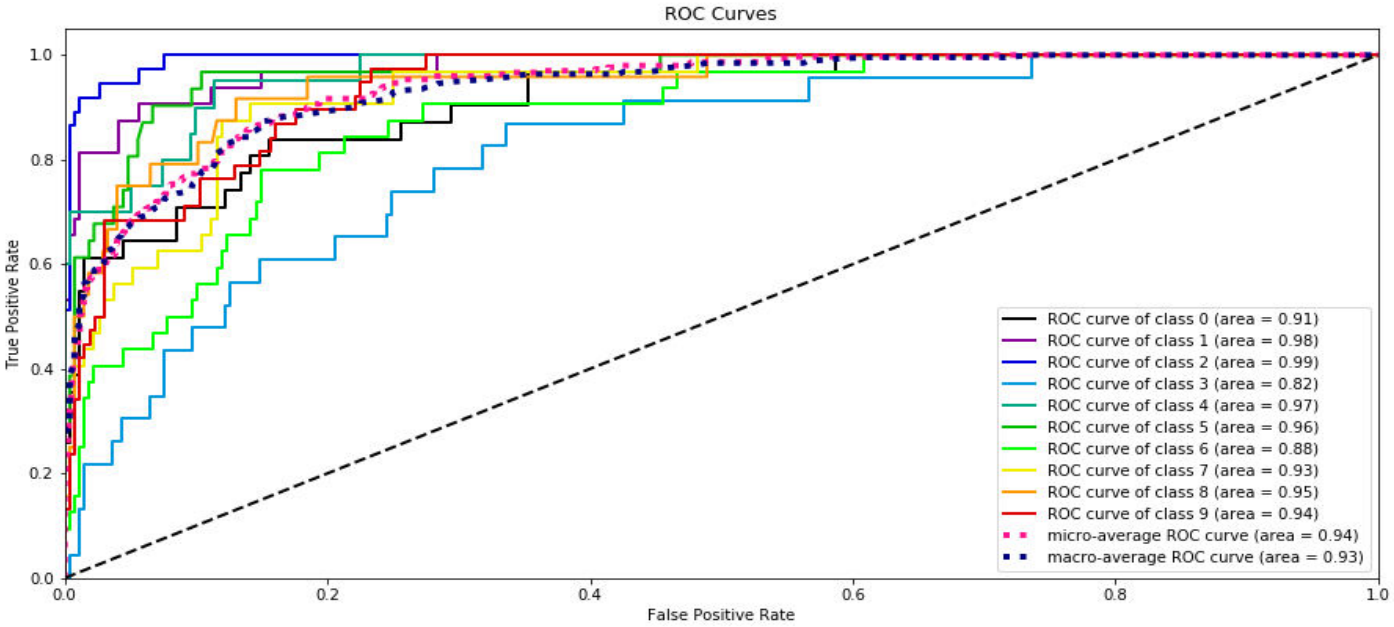


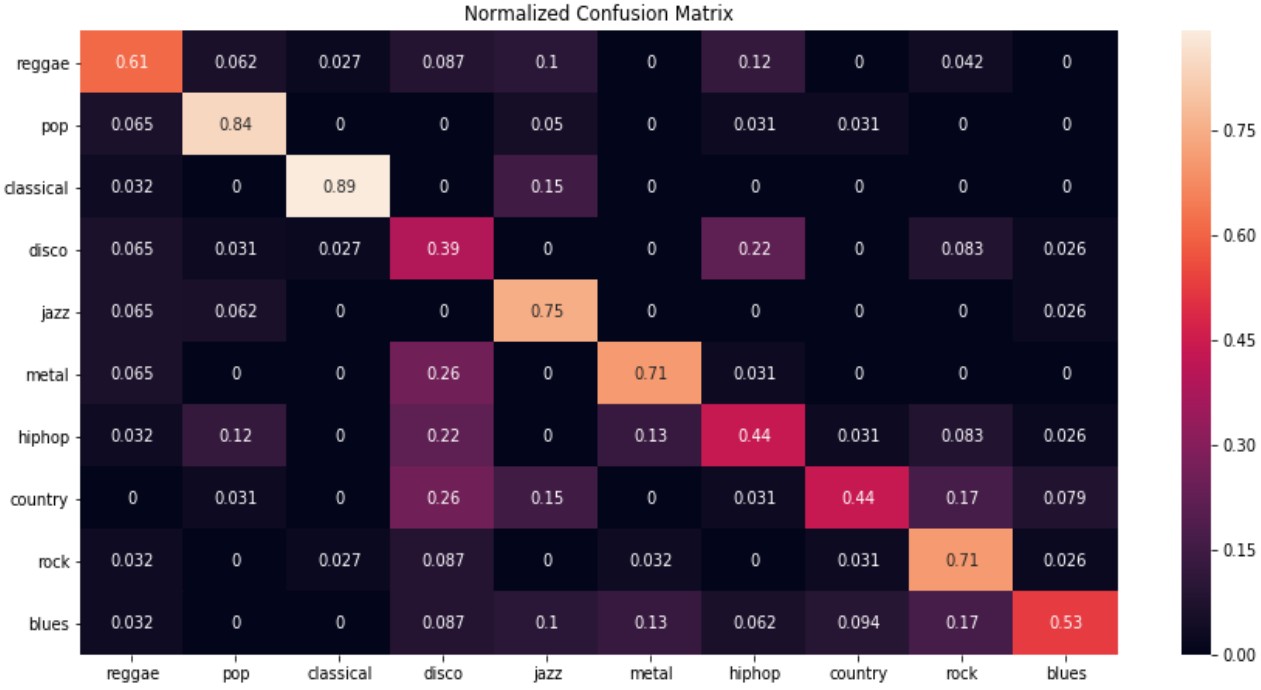


We see improvement over benchmark model that RF is able to more correctly classify classes like metal/blues/country but it's performance on hip-hop genres and reggae genres is low.

5. Combining both Features:

We have tried both types of featurization and now we will combine them. Combining both the features we hope to exploit the goodness of both featurization. All 3 models were trained on combined features as well. The XgBoost model performed the best out all the models trained so far. It had a roc auc score of 0.94 beating RF with MFCC feature and beating the benchmark XgBoost. The following images show its performance.





Refinement:

Each and every model trained was refined using hyperparameter tuning. The dataset was split into the ratio 70% train and 30% test set.

Train set was used to train the model and multiple parameter combinations were tried for each model and the ones which give the best result were chosen.

For validation/testing purposes the test data was used. We plot the confusion matrix for test data as well as draw the roc auc curve for each model.

For logistic Regression we chose alpha (learning rate) and penalty type (l1 or l2) as our hyperparameters to tune.

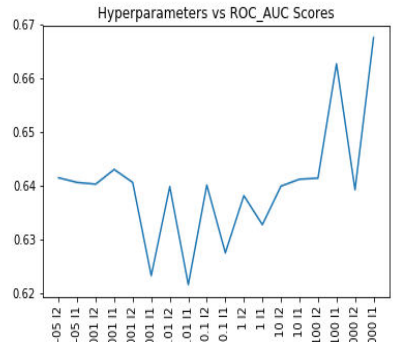
For Random Forest we chose no of estimators/trees and depth of trees as hyperparameters to tune.

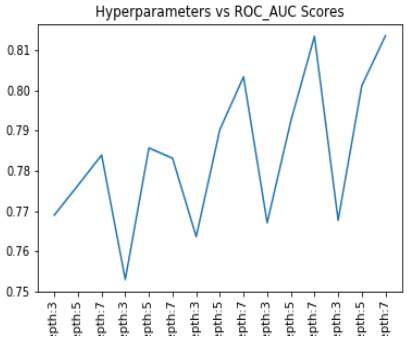
For XgBoost we choose no of estimators/trees and depth of trees as hyperparameters to tune.

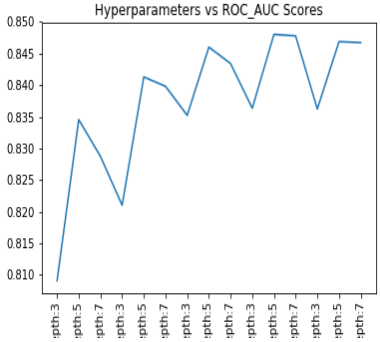
Model + Featurization	**Best Parameter Values**
Logistic Regression FFT Featurization	C = 1000, penalty = l1
Random Forest FFT Featurization	N_estimators = 500, max_depth = 7
XgBoost FFT Featurization	N_estimators = 300, max_depth = 7
Logistic Regression MFCC Featurization	C = 1000, penalty = l2
Random Forest MFCC Featurization	N_estimators = 300, max_depth = 7
XgBoost MFCC Featurization	N_estimators = 500, max_depth = 5
Logistic Regression Combined Features	C = 1, penalty = l1
Random Forest Combined Features	N_estimators = 300, max_depth = 7
XgBoost Combined Features	N_estimators = 500, max_depth = 3

For FFT Features Hyperparameter tuning results:

Three images for LR, RF and XgBoost respectively.

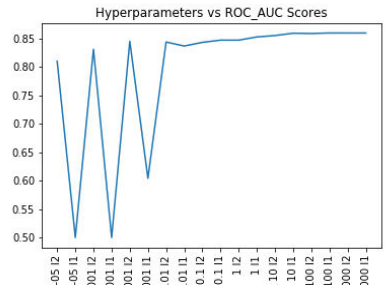


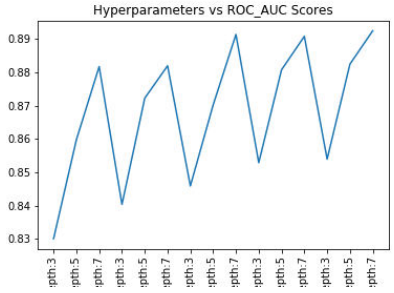


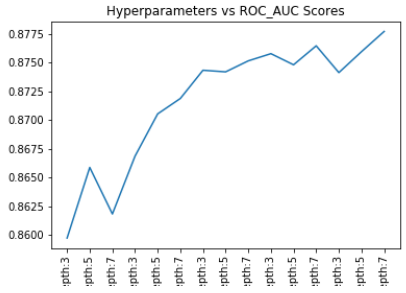


For MFCC Features Hyperparameter tuning results:

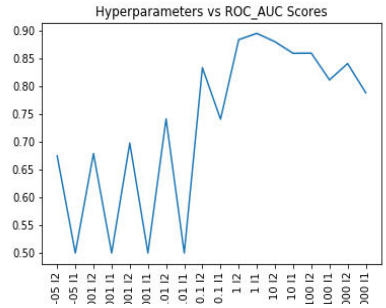
Three images for LR, RF and XgBoost respectively.

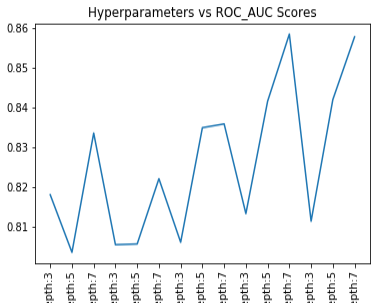


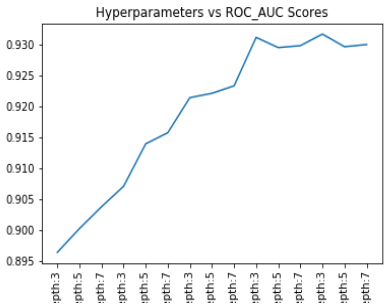




For Combined Features Hyperparameter tuning:







Results:

Our best model is the XgBoost Model trained on combined features (FFT+MFCC) which has all the default parameters excluding:

1. Number of Estimators: 500
2. Max Depth: 3

The above numbers are obtained through hyperparameter tuning as discussed in refinement section.

The roc auc score of the model is 0.93 (macro averaged) and 0.94 (micro averaged) which is similar to the score obtained on train dataset 0.936

Looking at the scores we can say that the model is not overfitting on the trainset as the scores are similar and it is also not underfitting as the score is quite good.

The model performs better than all the model tested on FFT, MFCC an combined and hence is chosen as the final model.

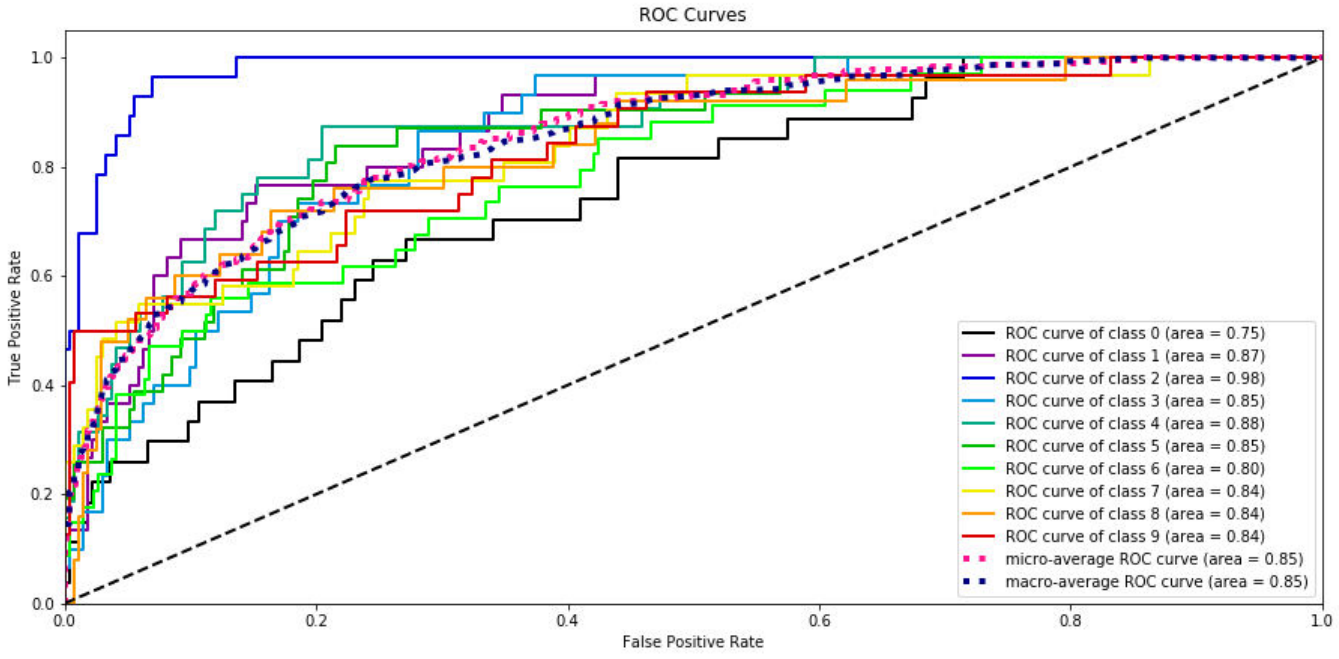
The benchmark model scored (roc auc score) 0.85 on test data while our best model got 0.94, an improvement of 0.09 on roc auc score and also looking at the confusion matrices of the two models we see that benchmark model did not do well of some genres but final model has improved performance on these genres and nearly same performance on other genres.

Name of Model + Featurization	**Micro averaged roc**	**Macro averaged roc**
XgBoost FFT Featurization (Benchmark)	0.85	0.85
XgBoost FFT + MFCC Featurization (Final model)	0.94	0.93

Conclusions:

Free Form Visualization:

An important quality of this project is that we have been able to achieve a very good roc auc score despite of the fact that we have no knowledge about how to handle audio/signal data. The FFT Features which we used as our benchmark features to get the benchmark model; the XgBoost Model was still able to achieve 0.85 roc auc score.



We used the features which have been known to work well with signal/audio data to get better results but these results aren't that bad of course we would want to do better and better always but what this kind of experimentation shows that even if we don't have domain expertise we can still make use of data in one-way or the other.

Reflection:

Summarizing the solution workflow: First we had to convert the audio data from .au format to .wav format which can be used to get FFT or MFCC features. Then we had to generate FFT and MFCC features from the .wav file so that we can train on them. Lastly, we build 3 ML Models namely Logistic regression, Random Forest Classifier and XgBoost Classifier and check their performance of the test set.

The part that I found interesting was the visualization part where I have plotted spectrogram, frequency and spectral power density plots. I found it interesting because I have little idea what the spectrogram and spectral power density plots do but even if I don't really know a lot, I can still look at them and use them to separate the different classes by just looking at them.

Improvements:

- One improvement can be to use more FFT features, I have only used top 1000, maybe using more of them give better results.
- Another improvement can be using Neural Networks. Neural Networks are known to find the best features from the available data. This can help us when we don't really have domain engineered feature. Neural Network of FFT Features or on MFCC Features may further improve the performance.
- A CNN Based Approach is also mentioned in this blog: https://medium.com/@CVxTz/audio-classification-a-convolutional-neural-network-approach-b0a4fce8f6c

References:

- Featurization Techniques: https://www.analyticsvidhya.com/blog/2017/08/audio-voice-processing-deep-learning/
- Music genre Info: https://medium.com/giglue/top-10-genres-of-music-industry-7f19cdb177cb
- Metrics for Multi-class classification: https://medium.com/usf-msds/choosing-the-right-metric-for-evaluating-machine-learning-models-part-2-86d5649a5428
- Hyperparameter Tuning: https://www.jeremyjordan.me/hyperparameter-tuning/