Capstone Project 1: Track genre classification based on its audio features

Final Report

Problem Statement

There are many types of genres in music ranging from pop to rock to classical. The objective of this project is to see if there are any features that define these genres of music.This would be useful to companies putting out new music, but are trying to define what genre this music would fall under. This would also be useful to customers trying to find new music in a certain genre.

Dataset

The dataset/s to be used are tracks (or songs) from differing genres that are part of Spotify's catalog. The different genres of tracks in this project will be classical, country, hiphop, pop and rock. Each track can can be accessed by using Spotify's Developer API. There are two types of information for each track that is provided by Spotify, Audio Analysis and Audio Features. The Analysis gives data on 'low-level audio analysis for all of the tracks in the Spotify catalog. The Audio Analysis describes the track's structure and musical content, including rhythm, pitch, and timbre.' Whereas the Features gives track data on 'audio feature information' which include: acousticness, danceability, energy, instrumentalness, liveness, loudness, tempo, speechiness and valence. All of these features are measured from a scale of 0 to 1, except for tempo and loudness which are measured in BPM and dB, respectfully. A more detailed explanation of each feature can be found here.

For the analysis of the tracks, we are going to focus on the audio features. These features have been measured/evaluated by Spotify. In the future, there may be other things apart from the audio features like popularity or explicitness. But for now, we are going to focus on just the audio features.

Data Wrangling

The end goal of this data wrangling step would be to create a pandas dataframe that includes basic information and features of the tracks of a given playlist. The various steps included are below:

1. **Choose tracks for each genre:** Using Spotify, a genre oriented playlist was created that holds around 500 tracks each. Each track was either recommended by Spotify or chosen by the user and was screened for accuracy of classification.
2. **Familiarization with Spotify API:** The first step consisted of understanding how to utilize the Spotify API to get the track information and features among other commands. Luckily, the Spotify API and documentation is comprehensive. In order to get information from Spotify, be it playlist data or track data, one would have to submit a request from a url issues commands from Spotify. For example to get track info, you would use the url, https://api.spotify.com/v1/tracks/{track_id}. On top of the url, an authorization token is

also needed. This can be requested from Spotify. However, each token expires, so a new request would be needed every time the token expires. Commands that extract information are in json format.

3. **Extraction of track ids from playlist:** After familiarization with the API, a list of the track ids of a given playlist needs to be extracted. However, there was a max limit of 100 tracks issued for each command. Which means, if a playlist has more than 100 tracks, you would need to offset the command by a 100 or whatever set of hundred to get the rest of the tracks.
4. **Get track information:** Using the track ids, a command was issued to get the a specific set of (i.e. name, artists, album, etc.) information of each of the track ids. Once all the track info is combined into a list, that list is converted to a dataframe.
5. **Get track features:** Like the track info, a command was issued to get the features of each of the track ids. Also like the track info, this information was combined and then converted into a dataframe.
6. **Merge to form final dataframe:** Finally the end goal is reached by merging the two dataframes from above by their 'id'.

Fortunately, it seems to be that the information from Spotify is pretty complete so there were no need to deal with missing values. As each track is screened for accuracy, the need to remove outliers should not be needed. Also, since there are multiple features for each track, it would be difficult to ascertain what an track outlier would be.

Using the steps above, I was able to successfully get information from the Spotify catalog using its API and converted it into a dataframe that would be the basis of further exploration and analysis.

Exploratory Data Analysis

It should be clear that different genres of music have different sounds to make them distinct from each other. In order to show if there any differences between the features with respect to the genres, I used a swarm plot in conjunction with a violin plot which show the different distributions.

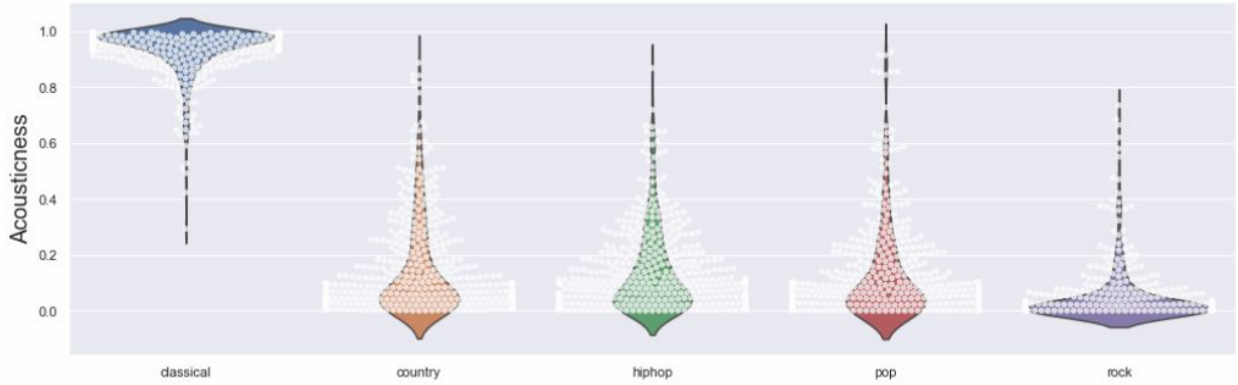

Figure 1. Feature plot showing the difference of acousticness of each track based on their genre. Each white dot correlates with a different track.

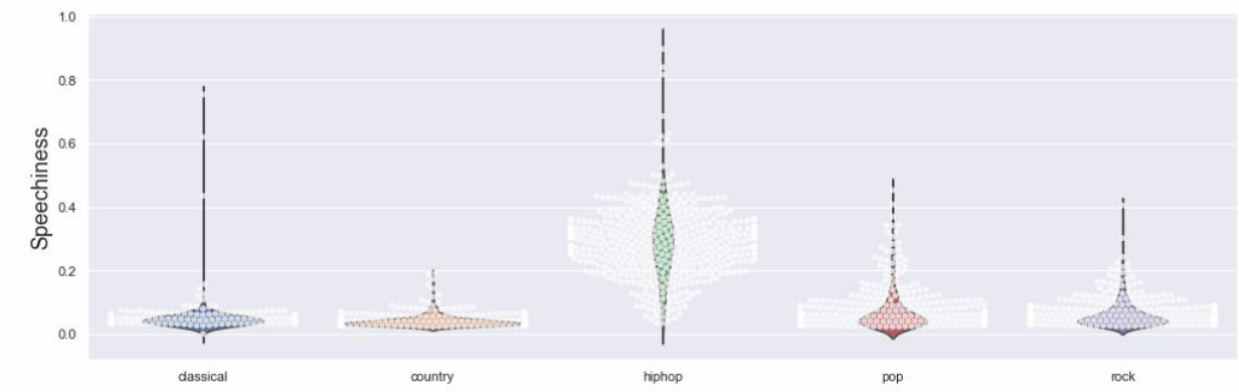

Figure 2. Feature plot showing the difference of speechiness of each track based on their genre. Each white dot correlates with a different track.

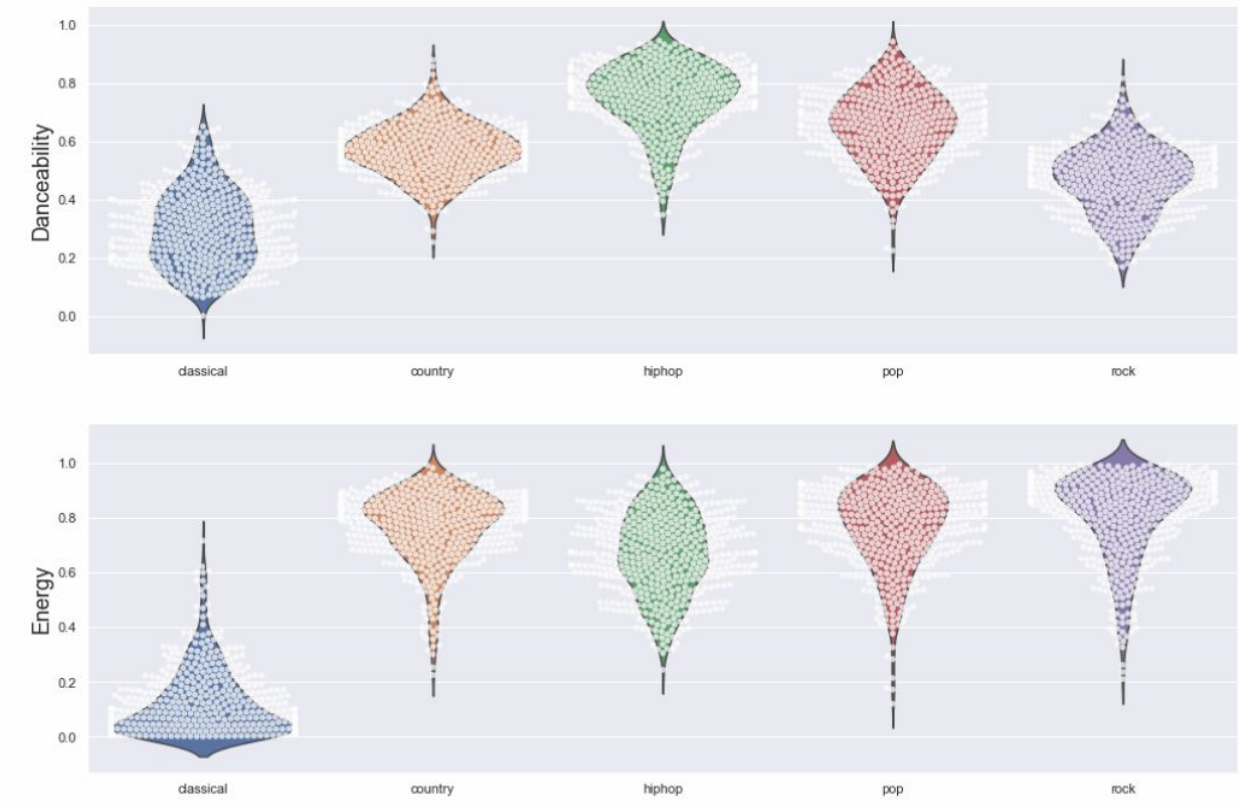

Figure 3. Feature plots showing the danceability and energy of each track based on their genre. Each white dot correlates with a different track.

Observations from the plots were that classical music tended to be acoustic and instrumental. Tends to lack energy, danceability and valence. Greater distribution of loudness compared to the other genres. The country tempo seems to be bimodal, but shares similar distributions with

pop. Hiphop has more speechiness compared to the rest of the genres. It is also the most danceable than all of the other genres. Like country, tempo seems to have a bimodal population. Finally, rock seems to be the least acoustic genre.

As we can from above, some of the different features have the same distributions as other features while some features are very different from the others. In order to quantitatively measure this, we can use a correlation matrix between the different features to see their correlations with one another. The following would tell us how strong a relationship is between the different features depending on their correlation value:

- |1.00| A perfect linear relationship
- |0.70| A strong linear relationship
- |0.50| A moderate relationship
- |0.30| A weak linear relationship
- |0.00| No linear relationship

Basically, the closer the the correlation value is to one, the stronger the relationship between the two features are.

	acousticness	danceability	energy	instrumentalness	liveness	loudness	speechiness	valence	tempo
acousticness	1	-0.55284	-0.868843	0.781645	-0.184027	-0.805688	-0.182699	-0.567468	-0.219939
danceability	-0.55284	1	0.472092	-0.562811	0.0471952	0.526474	0.40867	0.669614	-0.0746512
energy	-0.868843	0.472092	1	-0.729134	0.233415	0.885641	0.116896	0.623373	0.273053
instrumentalness	0.781645	-0.562811	-0.729134	1	-0.151355	-0.759115	-0.21284	-0.550656	-0.19793
liveness	-0.184027	0.0471952	0.233415	-0.151355	1	0.167566	0.144169	0.128352	0.0515195
loudness	-0.805688	0.526474	0.885641	-0.759115	0.167566	1	0.0933862	0.582664	0.252483
speechiness	-0.182699	0.40867	0.116896	-0.21284	0.144169	0.0933862	1	0.231568	-0.0676006
valence	-0.567468	0.669614	0.623373	-0.550656	0.128352	0.582664	0.231568	1	0.168341
tempo	-0.219939	-0.0746512	0.273053	-0.19793	0.0515195	0.252483	-0.0676006	0.168341	1

Table 1. Correlation matrix of the different features.

According to the matrix above, acousticness, energy, instrumentalness, and loudness have strong linear relationships with each other as they all have correlations above .7 (positive or negative) from each other. Since we are finding strong relationships, we need to find a way to combine them. A good method to do this is to do a principal component analysis (PCA). A PCA basically takes the features and transforms them into an array with the same amount of features that loses all the correlations between the features. Table 2 (below) shows the transformation of the feature values to the first three principal components.

	acousticness	danceability	energy	instrumentalness	liveness	loudness	speechiness	valence	tempo
0	0.425970	-0.337313	-0.427913	0.406291	-0.113744	-0.422367	-0.137819	-0.362286	-0.122520
1	0.097023	0.435944	-0.191635	0.034130	0.000214	-0.174251	0.614263	0.138209	-0.579539
2	0.025921	-0.195495	0.023555	0.058718	0.927537	-0.062078	0.258613	-0.107427	0.120449

Table 2. The ‘weight’ matrix of the first 3 principal components with respect to the various features.

The above table shows that the first component seems to favor acousticness and instrumentalness which are traits of the classsical genre. The second component seems to favor speechiness and danceability which are traits of the hiphop genre.

Another useful feature about PCA is that you can plot the first two principal components and see if you can find anything interesting. This is seen below:

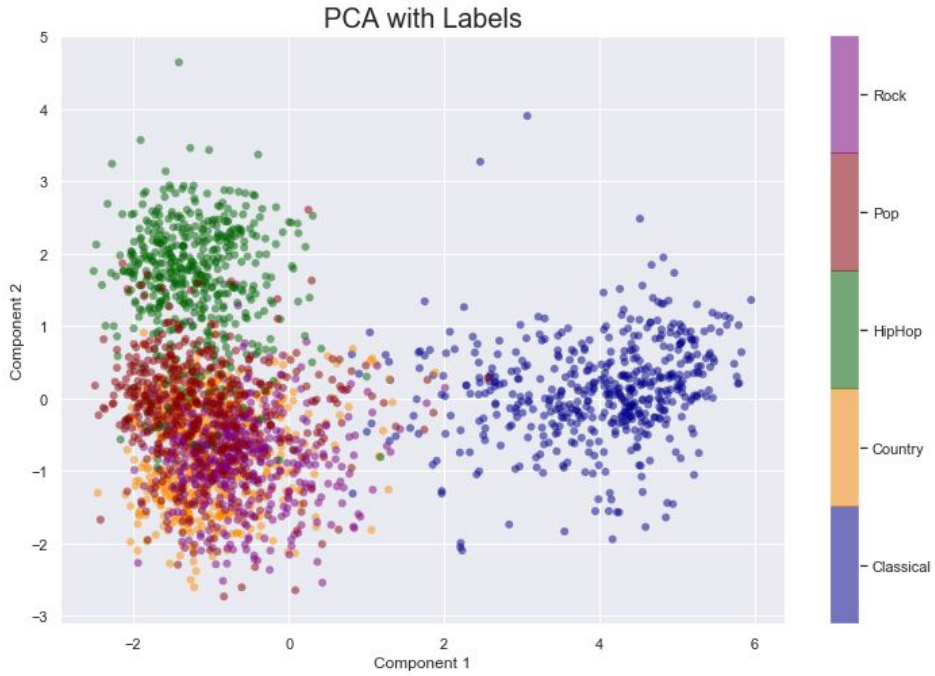

Figure 5. PCA plot of the first two principal components of the dataset labeled in their different respective genres (i.e. rock, pop, hiphop, country and classical).

From the PCA plot, we can see that classical seems to be very different from the other genres based on the first principal component, whereas hiphop seems to be different from the other genres based on the second principal component. The aforementioned observations are in line with the analysis of the PCA 'weights' of the features above. Therefore, the classical and hiphop genres look to be the easiest to discern based on a couple of features where they are both clearly separate from the genres. This can be seen in instrumentalness and acousticness for classical, and speechiness for hiphop. A classifier might have a more difficult time with country,

pop, and rock as there is only minor differences in the feature values. These differences and similarities is further exposed in the PCA plot above.

Another useful feature of PCA is that the principal components are ordered by the amount of variance of each component meaning that the first couple of principal components will contain a bulk of the variance. This can lead to dimension reduction in modeling as you want to explain as much variance as possible with as little features.

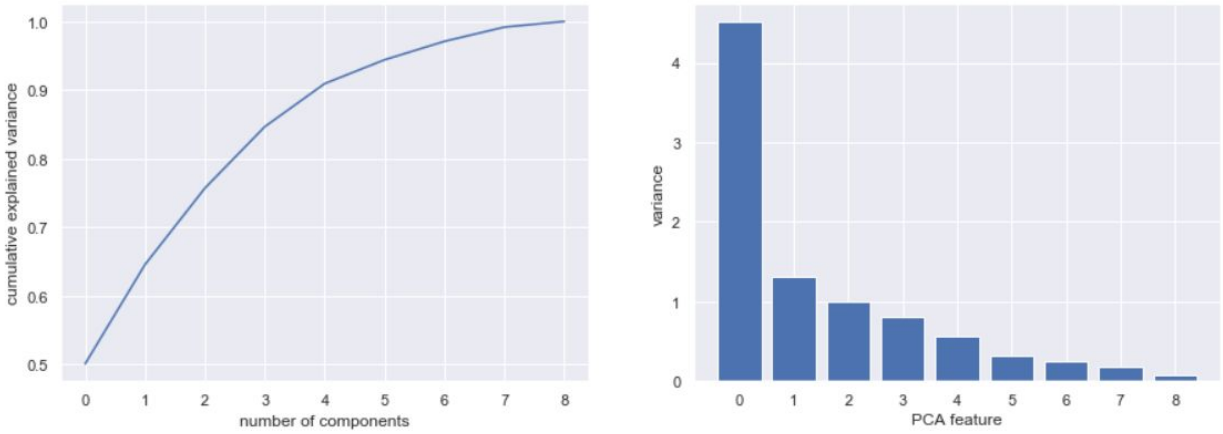

Figure 5. Explained variance of the PCA.

In-depth Analysis & Modelling

After exploring the data, we have some intuition on if it would be possible to classify the tracks by genres. From the previous analysis, it can be concluded that there is enough variance between the groups that a classifier may be successful at classifying the data. Therefore, we can continue on to create a model. For the model to be successful, it would need to be able to classify the tracks by genre somewhat accurately.

As this is a classic classifier problem, a logistic regression model would probably be best because the feature values are continuous and a regression model does well with that type of data. Other classifiers like k-nearest neighbors or SVM would probably not work as well because as we've seen earlier the genres of pop, rock and country have similar values close to each other in the PCA plot. Therefore, it would be difficult for KNN or SVM to create boundaries for these genres without overfitting like crazy. Another plausible candidate for a classifier is Naive Bayes, but for now, we will be using a logistic regression model from the sklearn library.

These are the steps taken to create the model:

1. The first step before training the model would be to preprocess the data to get it ready. In this step I first used was to scale the values of the data. This is an important step to take because this allows your data to be on a similar scale. I used a standard scaler. After scaling, I transformed the data set to its principal components using PCA. As explained before, the PCA removes the correlation between the different features which is useful.
2. The next step would be to create an instance of the logistic regression model.
3. After creating the instance, a pipeline was created to go through scaling, the PCA transformation, and the classifier model.
4. The data would then be split between training and testing. I used a 75-25 split between training and testing groups, respectively, with stratification so there is a congruency between the testing and training sets.
5. Then I would want to tune the hyperparameters. For this model, there are two hyperparameters to tune, the number of components from the PCA to use and the C from the logistic regression classifier. To tune it, I am going to utilize a grid search cross validation technique with 5 groups so it can which hyperparameters are optimal to use.
6. After setting up the previous steps, I can now train the pipeline using the training set.
7. Finally, I can validate and analyze the model by testing the pipeline by using the testing set.

The following table shows the results after training and testing the data:

```
The optimal parameters from training: {'logisticregression__C': 10, 'pca__n_components': 8}
The best score on the training data: 0.7869727709556861

Accuracy on the test set: 0.816
```

	precision	recall	f1-score	support
classical	0.98	0.99	0.99	125
country	0.68	0.72	0.70	125
hiphop	0.93	0.96	0.94	125
pop	0.69	0.54	0.61	125
rock	0.77	0.86	0.82	125
micro avg	0.82	0.82	0.82	625
macro avg	0.81	0.82	0.81	625
weighted avg	0.81	0.82	0.81	625

Table 3. Accuracy testing on logistic regression model.

As we can see, the best parameters to use is a C value of 10 and all 8 PCA components. The

best score from the grid search cross validation turns out to be ~79% accurate. With using this model and hyperparameters, we get a final testing accuracy score of 81.6% which surprisingly does better than the training set. By looking deeper into the scoring, we can see the precision and recall for each of the genres. This is very close to what we can expect from what the EDA showed us in the previous section. The two genres that did the best in the classification is classical and hiphop with accuracies higher than 90% whereas the other genres did not do so well. Recall that the classical and hiphop genres had populations that were separate while the other genres' populations were not so easily separable.

To get more intuition on the incorrect predictions, a confusion matrix was created. As can be seen, the test set has a total of 125 samples per genre with a total test set of 625 samples. In it, we can see that the model had an affinity to predict the rock genre with having 140 predictions of rock resulting in an increase of the false positive rate (i.e. a track was considered rock when in reality it was part of another genre). Pop, on the other hand, had only 98 predictions which means it contained more false negatives (i.e. a track was considered as another genre when in reality it was part of pop).

	classical_pred	country_pred	hiphop_pred	pop_pred	rock_pred	total
classical	124	1	0	0	0	125
country	0	90	1	20	14	125
hiphop	0	0	120	5	0	125
pop	1	31	7	68	18	125
rock	1	10	1	5	108	125
total_pred	126	132	129	98	140	625

Table 4. This table shows the actual count versus the predicted count of each genre for the testing.

Another way to evaluate a model and its classification ability is by showing its receiving operating characteristic (ROC) curve (below). It basically shows a binary classification (true versus false) of the intended class. As a rule of thumb, areas under the curve (AUC) values higher than 0.80 indicate that the model does a good job discriminating the focused label versus the rest.

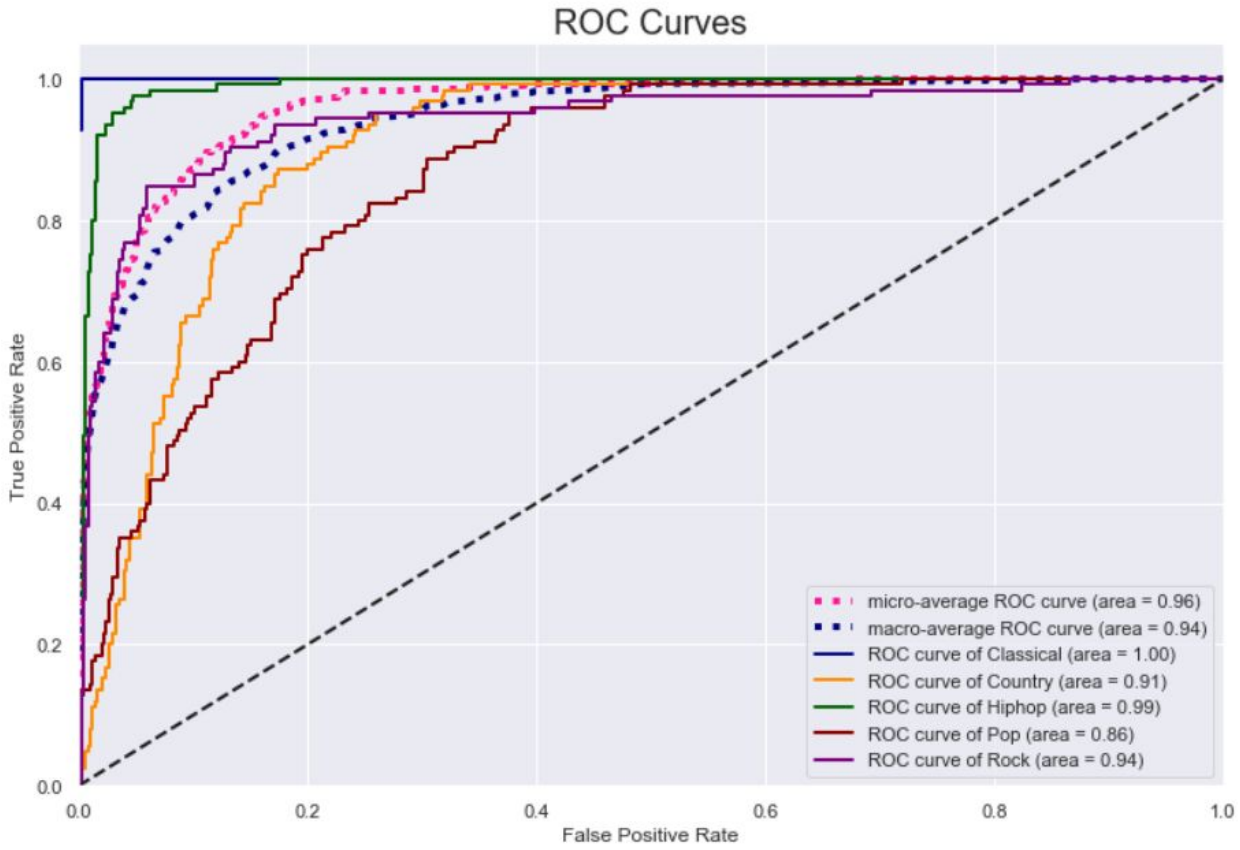

Figure 6. ROC curves of the different classifications based on Logistic Regression classification.

As can be seen in the above figure, the AUC for each of the genres are greater than 0.80.

Conclusions

Write some of the conclusions here...