Final Report

Automatic analysis of musical performance

Daniel Law - 1821780

Supervisor - Andrew Jones

Abstract

This report will explore my investigation into automatic music genre classification. This is an important task as so much musical history exists online and requires indexing. This paper describes my process of extracting audio features from a dataset and building a classifier to categorise pieces of music according to genre. My approach also places emphasis on how specific audio features relate to the overall accuracy of a classifier. I achieved a mean accuracy of 82% across ten music genres which is comparable to previous work in this field.

Acknowledgements

Thanks to my supervisor Andrew Jones for good feedback throughout the project. Thanks also go out to the computational music team for consistent support.

Table of Contents

Abstract **1**

Acknowledgements **1**

Table of Contents **1**

Introduction **5**

- Goals 5
- Impact 5
- Assumptions 6

Background **6**

- Previous work 6
- Critical aspects 6
- Introduction to audio features 6
- Machine learning 7

Approach **7**

- Solution requirements 7
- Audio features used 8
  - Chromagram 8
  - Root-mean-square energy 10
  - Spectral Bandwidth 10
  - Spectral Centroid 11

Spectral Contrast 12
Spectral Flatness 13
Spectral Flux 14
Spectral Rolloff 15
Tempo 15
Zero-crossing rate 16
Algorithms used 17
Dimensionality reduction 17
T-distributed Stochastic Neighbour Embedding 17
Uniform Manifold Approximation and Projection 18
Machine learning 18
GridSearchCV 18
Logistic Regression 18
SGD Classifier 19
Decision Tree 19
Random Forest 20
Support Vector Machine 20
K-Nearest Neighbours 21
Multilayer perceptron 22
Feature Scaling 22
Standardisation 23
Normalisation 23
Train test split 23
Ensemble 23
Voting Classifier 24
Evaluation 24
Confusion Matrix 24
List of tools and resources 25
Datasets 25
GTZAN 26
Programming language 26
Libraries 26
Code resources 26
Implementation 26
My algorithms 26
Feature extraction 27
Get_features 27
Read_process_songs 27
Visualisation 27
Get_tsne_embeddings 27
Get_umap_embeddings 27
Plot_components 27
Classifier 27
Plot_confusion_matrix 27

Problems encountered 27

Results and Evaluation **27**
Empirical observation of clusters obtained 27
Final testing 32

Conclusions **36**

Future Work **36**
Feature selection 36
Naive bayes implementation 37
Evaluation methods 38
Dataset experimentation 38
Standalone program 39

Reflection on Learning **39**

References **40**

Table of abbreviations **42**

Appendices **42**
1: Confusion matrices for various algorithms on full dataset of features 43
2: Confusion matrices for individual features 49

List Of Tables

- Table 1. A discussion of considered datasets
- Table 2. A table of abbreviations used throughout this report

List Of Images

- Figure 1. A visual representation of my proposed pipeline
- Figure 2. A visualisation of a chromagram
- Figure 3. A visualisation of MFCC coefficients
- Figure 4. The calculation for RMSE
- Figure 5. A few visualisations of RMSE
- Figure 6. The calculation of spectral bandwidth
- Figure 7. A visualisation of spectral bandwidth
- Figure 8. The calculation of spectral centroid
- Figure 9. A visualisation of spectral centroid
- Figure 10. A visualisation of spectral contrast
- Figure 11. The calculation of spectral flatness
- Figure 12. A visualisation of spectral flatness
- Figure 13. A visualisation for spectral flux
- Figure 14. A visualisation of spectral rolloff
- Figure 15. A visualisation of tempo
- Figure 16. The calculation for zcr
- Figure 17. A visualisation for zcr

- Figure 18. A calculation for the logit function
- Figure 19. A visualisation for the logit function
- Figure 20. A visualisation of the decision boundary of a SGD Classifier
- Figure 21. A graph detailing the flow of a decision tree algorithm
- Figure 22. Visualisation of a hyperplane calculated by an SVM
- Figure 23. An illustration of how the KNN algorithm works
- Figure 24. A visualisation of a hidden layer MLP
- Figure 25. The calculation for standardisation
- Figure 26. The calculation for normalisation
- Figure 27. An example confusion matrix
- Figure 28. 2D representation of the t-SNE algorithm performed on my dataset for various perplexities and iterations
- Figure 29. 2D representation of the UMAP algorithm performed on my dataset for various neighbours and min distance (parameters)
- Figure 30. 3D representation of the t-SNE algorithm performed on my dataset
- Figure 31. 3D representation of the UMAP algorithm performed on my dataset
- Figure 32. The confusion matrix obtained by the voting classifier on my dataset
- Figure 33. The confusion matrix obtained by the voting classifier on just the MFCC feature's values
- Figure 34. The confusion matrix obtained by the voting classifier on the values for MFCC, Chroma and Spectral Bandwidth
- Figure 35. The confusion matrix obtained by the voting classifier on the values for MFCC, Chroma, Spectral Bandwidth and Zero Crossing Rate
- Figure 36. The confusion matrix obtained by the voting classifier on the values for MFCC, Chroma, Spectral Bandwidth,Zero Crossing Rate and Spectral Flux
- Figure 37. The confusion matrix obtained by the voting classifier after the SelectKBest algorithm was implemented
- Figure 38. The calculation for naive bayes theorem
- Figure 39. The confusion matrix for the Logistic Regression algorithm on the full suite of features

Appendix 1: Confusion matrices for various algorithms on full dataset of features

- Figure 40. The confusion matrix for the SGD Classifier algorithm on the full suite of features
- Figure 41. The confusion matrix for the Decision Tree algorithm on the full suite of features
- Figure 42. The confusion matrix for the Random Forest algorithm on the full suite of features
- Figure 43. The confusion matrix for the KNN algorithm on the full suite of features
- Figure 44. The confusion matrix for the SVM algorithm on the full suite of features
- Figure 45. The confusion matrix for the MLP Classifier algorithm on the full suite of features

Appendix 2: Confusion matrices for individual features

- Figure 46. The confusion matrix for the Voting Classifier algorithm on solely the Spectral Bandwidth feature
- Figure 47. The confusion matrix for the Voting Classifier algorithm on solely the Chroma feature
- Figure 48. The confusion matrix for the Voting Classifier algorithm on solely the

Spectral Contrast feature
- Figure 49. The confusion matrix for the Voting Classifier algorithm on solely the Spectral Flatness feature
- Figure 50. The confusion matrix for the Voting Classifier algorithm on solely the Spectral Flux feature
- Figure 51. The confusion matrix for the Voting Classifier algorithm on solely the Root-Mean-Square-Energy feature
- Figure 52. The confusion matrix for the Voting Classifier algorithm on solely the Spectral Rolloff feature
- Figure 53. The confusion matrix for the Voting Classifier algorithm on solely the Tempo feature
- Figure 54. The confusion matrix for the Voting Classifier algorithm on solely the Zero Crossing Rate feature

Introduction

Goals

The aim of this project is to successfully evaluate the genre of a piece of music from an audio file using machine learning. The goal is to achieve an accuracy of at least 70% on any given piece of music as this is the minimum I would consider to be useful. A secondary goal of the project is to evaluate which audio features are most useful when evaluating genre as a combination of solid audio features might result in high accuracy. Additionally, it would be useful to know which features are not worth utilising as unnecessary data is a waste of processing time.

Impact

There is an increased demand for fast and accurate music classification by large companies such as Spotify, Apple and Amazon. Suggested music for their customers is informed by the metadata of what they are currently listening to. It is therefore imperative that this data is categorised precisely in order for these companies to deliver appropriate recommendations for new music on streaming platforms such as Spotify and Apple Music. Arguably the most important piece of this metadata is the genre of music which a song belongs to as it allows people to discover other music that is similar enough to belong in the same genre. As mentioned previously the goal of this project is to achieve an accuracy of 70%. This would be useful for the aforementioned platforms as it would allow them to deliver similar music to their customers the majority of the time. In the expected 30% cases that the music is not similar then their customers can simply skip the song until they find another song they like. Even in cases where the classifier is wrong there is still a chance that a user likes the suggested song. This could lead to them exploring recommendations based on the new song and potentially even lead to the same scenario of liking a song that the classifier incorrectly labelled. The advantage of my theoretical outcome over a method that randomly picks music for consumers to listen to is that it is correct enough times that it will still be useful at finding similar music.

This might lead to consumers gaining trust in the system and being more willing to try suggested music by the algorithms.

Assumptions

This project is founded on the assumption that there is a distinguishable difference in classification when utilising different audio features. If this is not the case, then the secondary goal of the project is not applicable.

Background

Previous work

There have already been plenty of investigations into the issue of genre classification. However, my investigation differs from these somewhat as I placed a greater emphasis on discovering how different audio features relate to the overall accuracy of a classifier. In 2002 Tzanetakis and Cook [1] produced the first major piece of mainstream work on genre classification. They collected three sets of features that represented timbral textures, rhythmic content and pitch content and utilised a number of classifiers in order to categorise their results into a genre hierarchy. They achieved an accuracy of 61% for their ten chosen genres, which was incredible for the time. This work is the basis that many others extended from. This includes many of the other works referenced in this report as they also reference their 2002 study. Additionally, Tzanetakis created the GTZAN dataset[2] which has been popularised to the extent that it is used in many evaluations today. Li et al. [3] showed that it is possible to achieve better results by using a hierarchical taxonomy. He automatically generated musical genre hierarchies by applying hierarchical clustering on confusion matrices generated via a validation set. Consequently, he achieved an accuracy score of 75% on the GTZAN dataset. Whilst this study provides valuable insights Ning et al. [4] compared two different audio features, spectral contrast and mel-frequency cepstral coefficients (MFCC). They concluded that using spectral contrast achieved a higher accuracy than MFCC features, with an average accuracy of 82.3% on 10-second music clips.

Critical aspects

To carry out this project, it is required to collect information that relates to music genres and to teach a machine how to understand this information. We call this information audio features and the teaching process machine learning. The following two subsections of this report give an overview of audio features and machine learning in more detail. Detailed in the approach section of this report is the specific audio features and machine learning algorithms used in this project and their relevance.

Introduction to audio features

In order for a machine to interpret genre from an audio file, we must first extract useful features from an audio signal. These features describe the audio signal in a variety of different ways, depending on the feature. We can categorise them in three different ways according to Mahanta [5]. High-level features refer to those understood and enjoyed by humans such as chords, key, rhythm and instrumentation. Mid-level features are those perceived, but not necessarily

appreciated by humans. This includes but is not limited to, pitch, beat-related descriptors and MFCCs. Finally, low-level features refer to statistical features that are imperceivable to humans. Examples of these are spectral centroid, spectral flux and zero-crossing rate. Usually low-level features are preferred when attempting music genre recognition (MGR), but there are some mid-level features that appear in many peoples work. For example, MFCCs are commonly used as a defining feature vector in many people's investigations as they contribute heavily to the overall accuracy of a classifier.

Machine learning

Machine learning refers to the process of teaching a computer how to interpret and learn from received data without being explicitly programmed to do so. Machine learning algorithms are already popularised, being used so frequently that some may not even realise it. This includes speech recognition in virtual assistants such as Alexa, image recognition of faces on social media or even simply filtering emails by spam. This project focuses on the supervised learning category of machine learning. To paraphrase IBM [6], supervised learning refers to the process of teaching models how to yield a desired output by using a training dataset. This training dataset is made up of inputs and correct outputs, which allows the model to learn as it attempts to yield the desired output. It computes the distance between the current output of the algorithm and the expected output, also known as the loss function. This allows the algorithm to adjust its learning accordingly until the error has been sufficiently reduced. The two types of supervised learning techniques are regression and classification. Regression is used to predict an output based on previous data which is commonly used in areas such as the stock market to project the future of stocks. Classification is used to separate data points into distinct groups, such as categorising emails into spam and not spam. This project will focus on using classification algorithms instead of regression algorithms as the aim is to predict a class value from discrete data whereas regression algorithms are used with continuous data to output a continuous variable. The classification algorithms used in this project are detailed in the approach section of this report.

Approach

Solution requirements

In order to achieve genre classification a number of requirements must be fulfilled. An appropriate dataset must be chosen as a good classifier requires a large sample size of data to learn from. From this dataset a number of audio features should be extracted. These features must be selected based on their performance in aiding MGR later on. As a precaution these extracted features should be visualised in order to observe that they are suitable for machine learning. Following this the new dataset containing the extracted features must be standardised in preparation for machine learning. A number of classification machine learning algorithms can then be trained on this new dataset, aiming to yield the correct genre. These algorithms should be selected based on their suitability and possibly their previous results in past works. An ensemble classification method should then be utilised in order to achieve an

optimal final result for the classifier. These results then need to be evaluated using an appropriate method in order to observe if the classifier is performing as expected. These requirements, including the tools and resources I used during this project, are detailed later on in this section of the report. Figure 1 below shows a visual representation of my proposed solution for a pipeline that fits the above requirements.

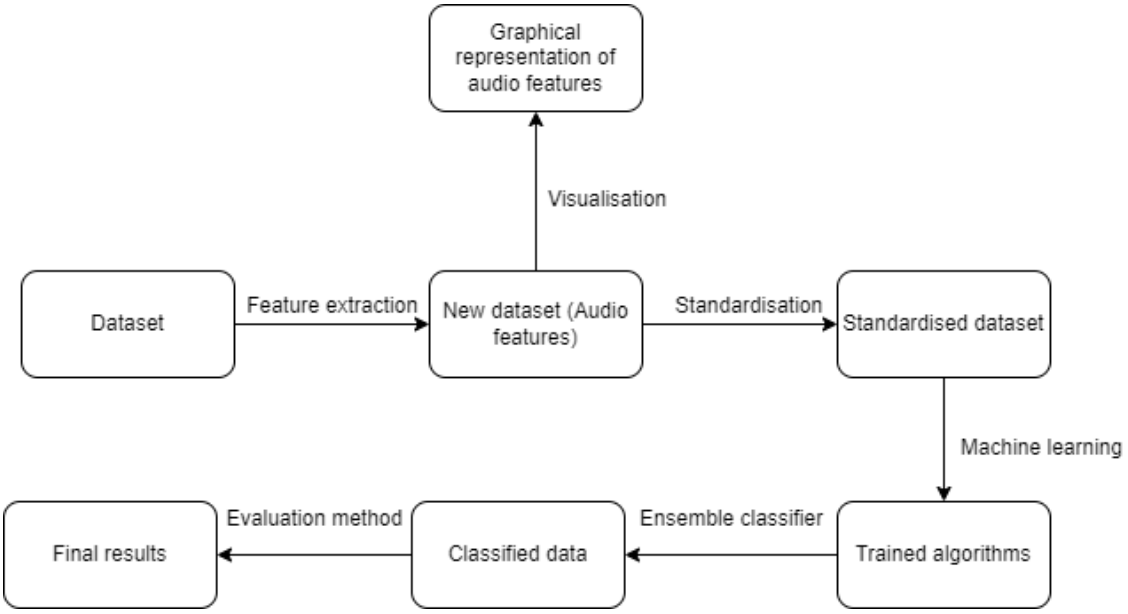

Figure 1. A visual representation of my proposed pipeline

Audio features used

This subsection details my chosen audio features. These were selected based on their usage in previous works and ability to aid MGR. Each feature is explained and where applicable a figure for its computation and visualisation is given.

Chromagram

A chromagram is a mid-level audio feature that describes the pitch of an audio signal. As Müller explains [7], “the human perception of pitch is periodic in the sense that two pitches are perceived as similar in "colour" (playing a similar harmonic role) if they differ by an octave. Based on this observation, a pitch can be separated into two components, which are referred to as tone height and chroma.” Tone height refers to the octave number and chroma refers to the pitch spelling attribute. Pitch spelling attributes are also referred to as musical notes, such as C, E♯ or G ♭ . A pitch class is defined as a set of all pitches that share the same chroma. For example the pitch class D contains the set of D’s for all octaves. A chromogram is derived by aggregating for a given time window all information that relates to a given chroma into a single coefficient. According to Shah et al. [8] the conversion of an audio music into a chromagram representation can be performed by using Short Time Fourier Transform (STFT) in combination with binning strategies. This feature was chosen as it can help identify chords. Some genres, such as pop are notorious

for sharing chord progressions between each other. This means that a chromagram should be able to assist a classifier identify genres with similar chords or keys.

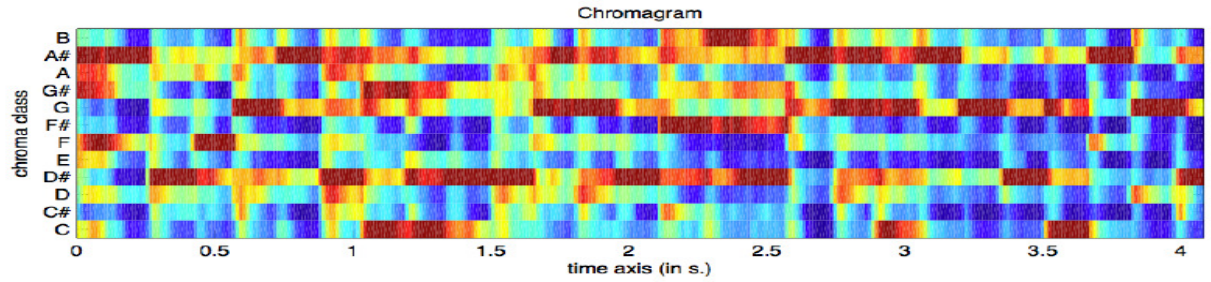

Figure 2. A visualisation of a chromagram

Mel-frequency cepstral coefficients

Chathuranga and Jayaratne [9] describe MFCC's as follows: "Mel Frequency Cepstral Coefficients (MFCCs) are compact, short time descriptors of the spectral envelope audio feature set and typically computed for audio segments of 10-100ms." These coefficients describe concentrations of spectral energy within the envelope. For example, a negative MFCC value means that most of the spectral energy is concentrated at higher frequencies and the inverse is true for positive MFCC values. A number of steps must be carried out to calculate these coefficients according to Lyons [10]:

1. Frame the signal into short frames.
2. For each frame calculate the periodogram estimate of the power spectrum.
3. Apply the mel filterbank to the power spectra, sum the energy in each filter.
4. Take the logarithm of all filterbank energies.
5. Take the DCT of the log filterbank energies.
6. Keep DCT coefficients 2-13, discard the rest.

This feature was chosen as it describes timbral information. This allows differences in instrumentation to be identified and therefore genres that contain repeated instrumentation could be categorised by the classifier.

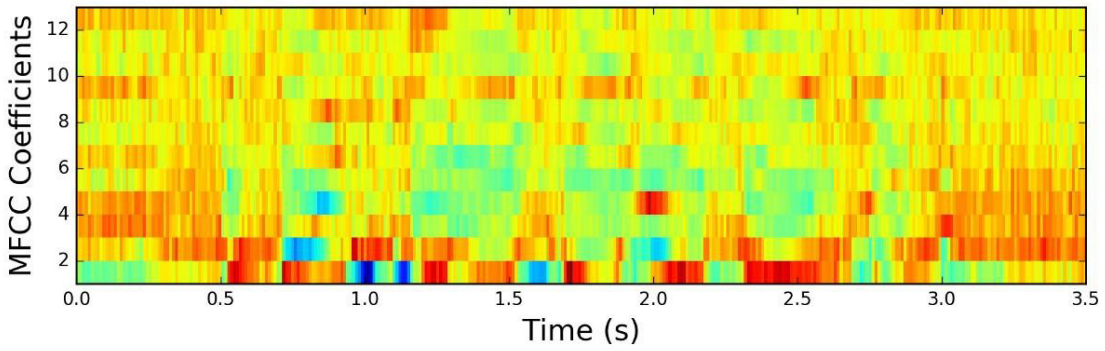

Figure 3. A visualisation of MFCC coefficients

Root-mean-square energy

The energy of an audio signal is defined by its magnitude, which relates to the volume of the audio. The root mean square energy refers to the average loudness of an audio signal. This feature was chosen in an attempt to better identify the metal genre. A common theme within metal music is loud volume, so the classifier is more likely to identify items with a high RMSE as metal.

$$\mathrm{RMS}_t = \sqrt{\frac{1}{K} \cdot \sum_{k=t \cdot K}^{(t+1)\cdot(K-1)} s(k)^2}$$

Figure 4. The calculation for RMSE

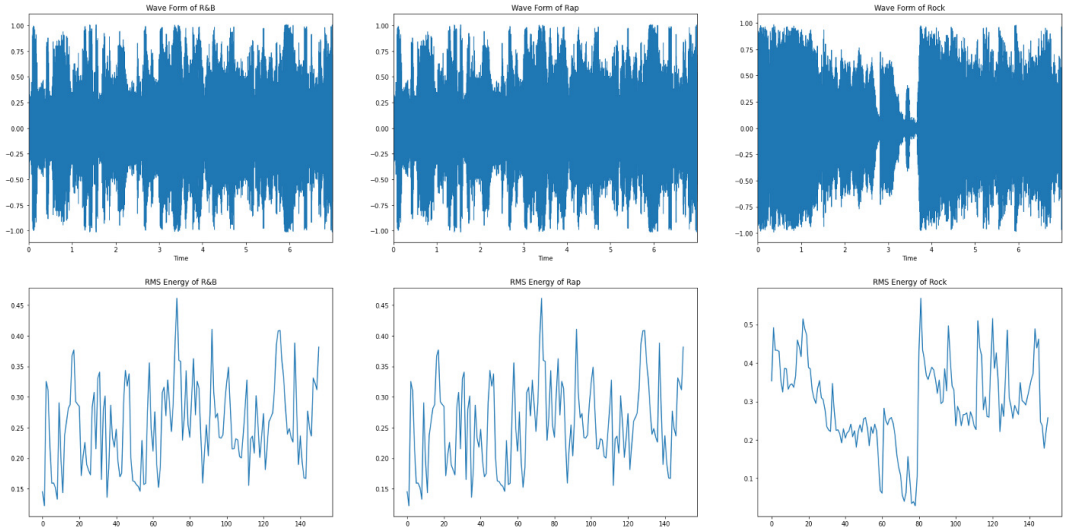

Figure 5. A few visualisations of RMSE

Spectral Bandwidth

The spectral bandwidth is the difference between the upper and lower frequencies in a continuous band. This feature was chosen for the same reason as MFCC as it describes timbre.

```
(sum_k S[k, t] * (freq[k, t] - centroid[t])**p)**(1/p)
```

Where p is order and t is time.

Figure 6. The calculation of spectral bandwidth

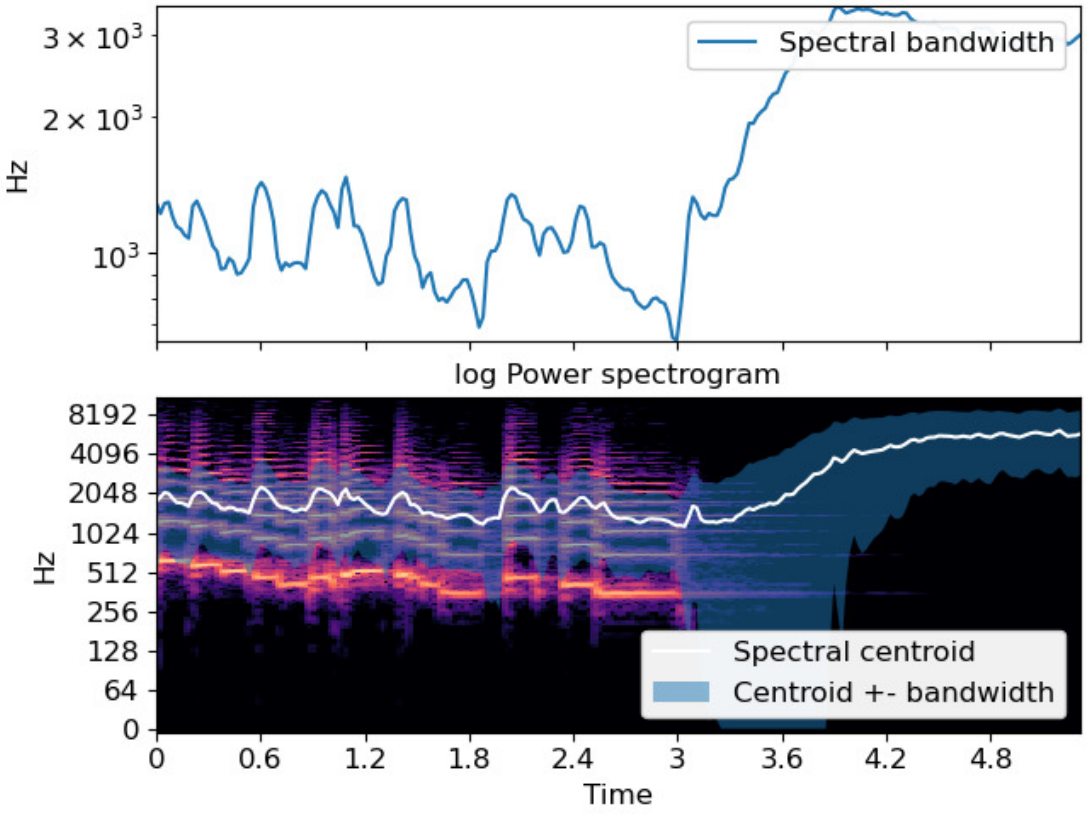

Figure 7. A visualisation of spectral bandwidth

Spectral Centroid

The location of the centre of mass of the spectrum. It is calculated using a Fourier transform with the magnitudes of the frequencies present in the signal used as weights, resulting in a weighted mean of these frequencies. This feature was chosen for the same reason as MFCC as it describes timbre.

$$\text{Centroid} = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)}$$

Where

- x(n) is the weight frequency value
- n is the bin number.
- f(n) is the centre frequency of the bin.

Figure 8. The calculation of spectral centroid

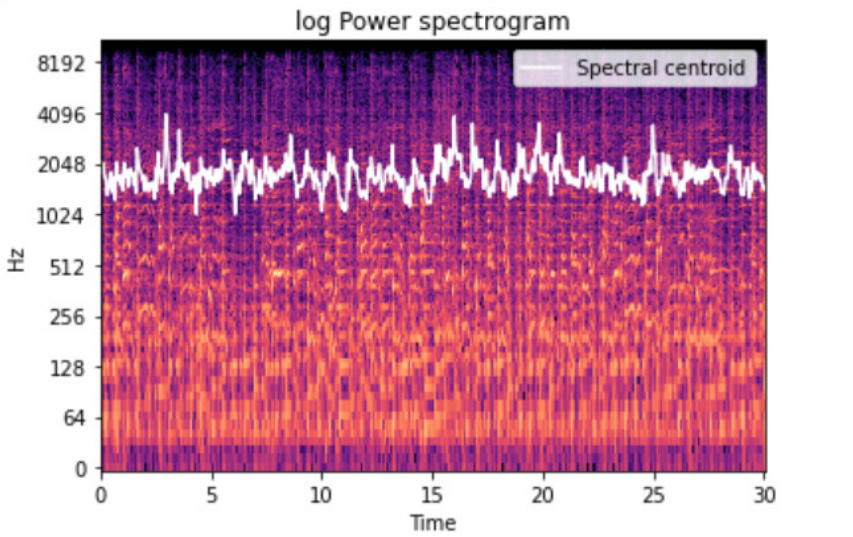

Figure 9. A visualisation of spectral centroid

Spectral Contrast

Spectral contrast is defined as the difference between spectral peaks and valleys within a spectrum. To calculate this, each frame of a spectrogram is divided into sub-bands. For each sub-band, the energy contrast is estimated by comparing the mean energy in the top quantile (peak energy) to that of the bottom quantile (valley energy). High contrast values generally correspond to clear, narrow-band signals, while low contrast values correspond to broad-band noise [4]. This feature was chosen for the same reason as MFCC as it describes timbre.

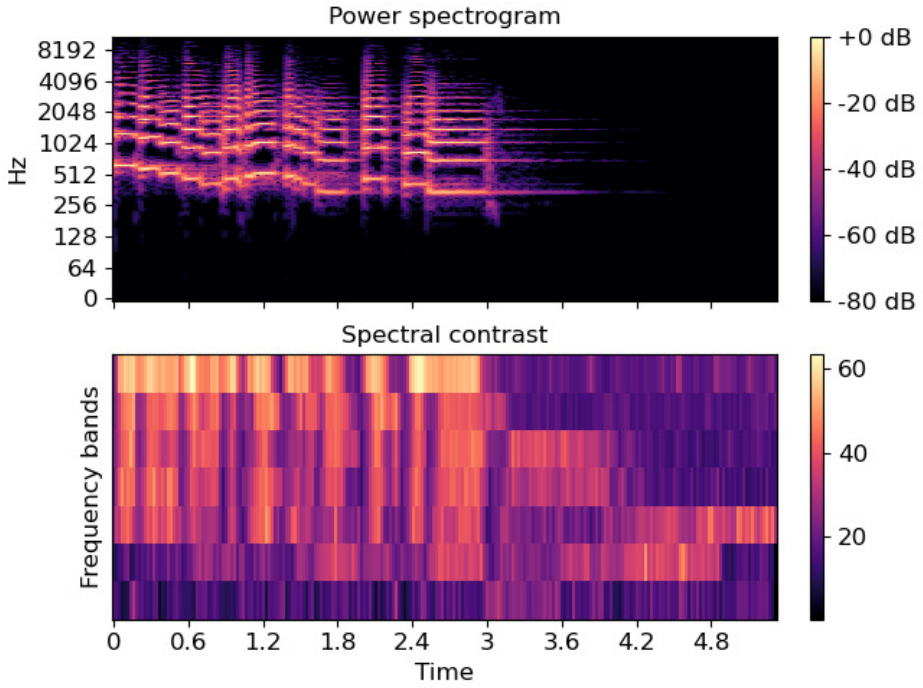

Figure 10. A visualisation of spectral contrast

Spectral Flatness

The spectral flatness is a measure of the tonality of an audio signal. Its value reflects whether the audio is tone-like or noise-like, with white noise having a high spectral flatness. This was chosen as the tonality of music can relate to chords, therefore similarly to chroma this feature might help identify genres such as pop.

$$\text{Flatness} = \frac{\sqrt[N]{\prod_{n=0}^{N-1} x(n)}}{\frac{\sum_{n=0}^{N-1} x(n)}{N}} = \frac{\exp\left(\frac{1}{N}\sum_{n=0}^{N-1} \ln x(n)\right)}{\frac{1}{N}\sum_{n=0}^{N-1} x(n)}$$

Figure 11. The calculation of spectral flatness

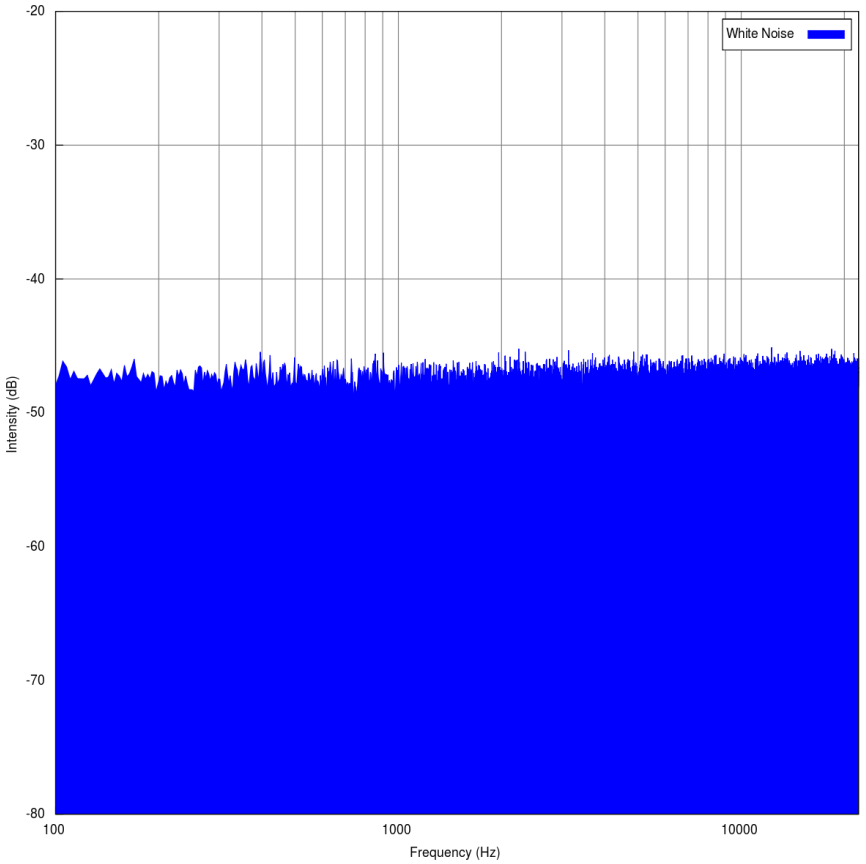

Figure 12. A visualisation of spectral flatness

Spectral Flux

Spectral flux refers to the rate of change between the power spectrum between frames within an audio signal. It is computed as the normalised difference between normalised magnitudes of two consecutive frames. This feature was chosen for the same reason as MFCC as it describes timbre.

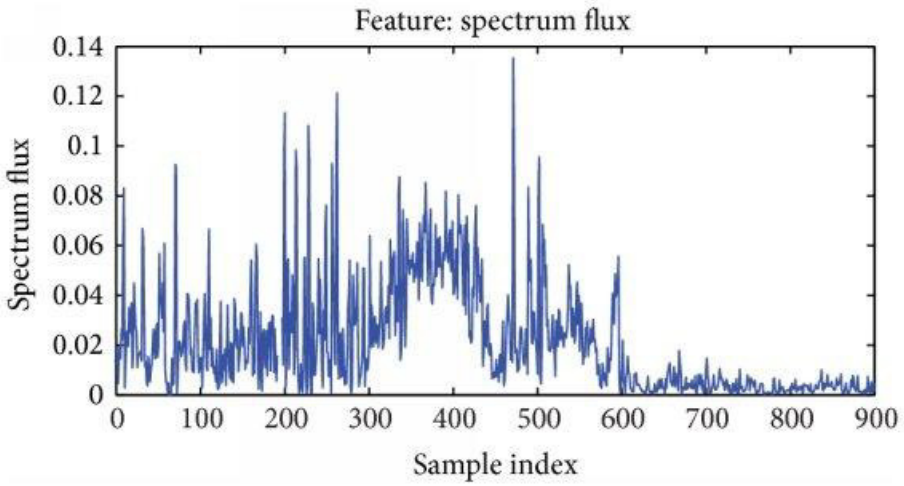

Figure 13. A visualisation for spectral flux

Spectral Rolloff

Spectral rolloff refers to the frequency below which a specified percentage of the total spectral energy lies, e.g 90%. This feature was chosen for the same reason as MFCC as it describes timbre.

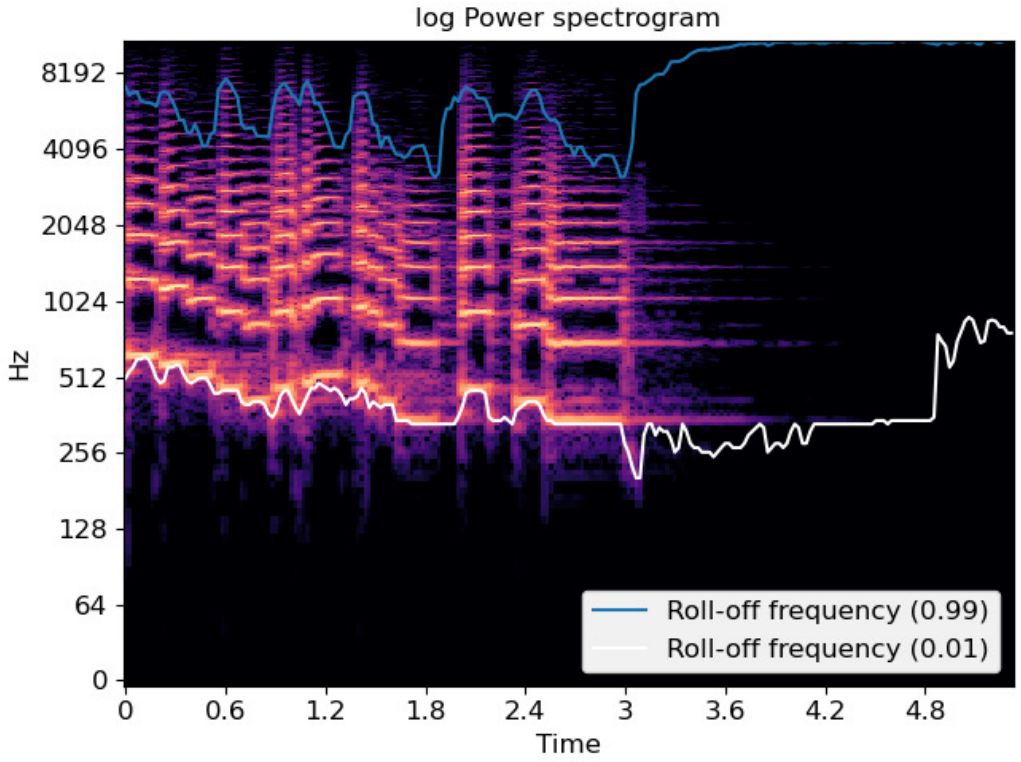

Figure 14. A visualisation of spectral rolloff

Tempo

Tempo refers to the number of beats per minute within an audio signal. This feature was chosen as some genres share similar tempos. For example reggae typically has a slow tempo so this feature might help the classifier in identifying it.

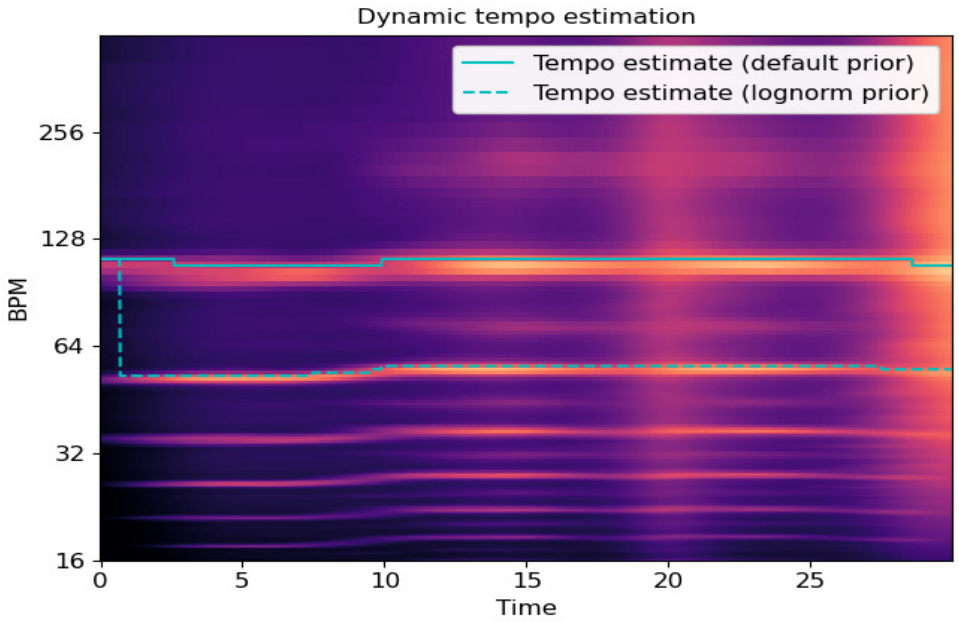

Figure 15. A visualisation of tempo

Zero-crossing rate

The zero-crossing rate (ZCR) refers to the number of times the signal changes from a positive value to a negative value, and vice versa. This feature was chosen for the same reason as MFCC as it describes timbre.

$$zcr = \frac{1}{T-1}\sum_{t=1}^{T-1} 1_{\mathbb{R}_{<0}}(s_t s_{t-1})$$

where s is a signal of length T and $1_{\mathbb{R}_{<0}}$ is an indicator function.

Figure 16. The calculation for zcr

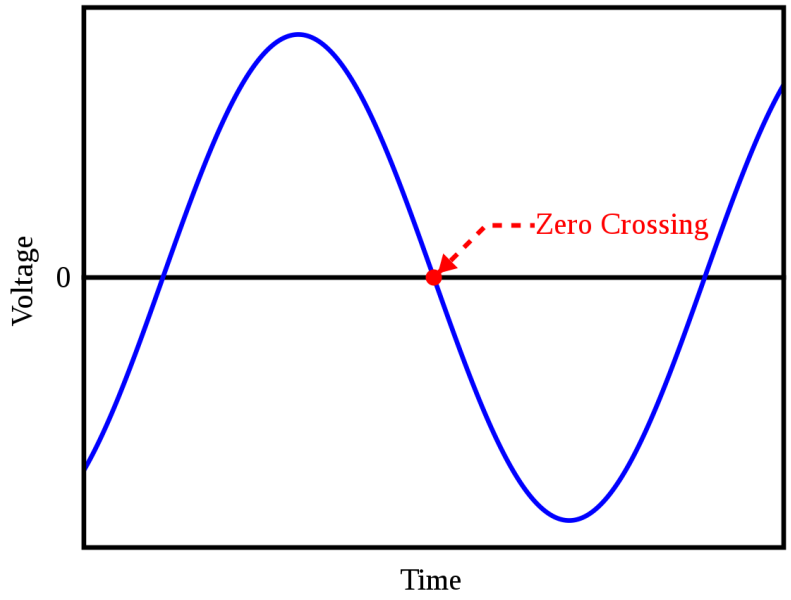

Figure 17. A visualisation for zcr

Algorithms used

This subsection details existing algorithms that I implemented during various stages of this project.

Dimensionality reduction

I utilised two different dimensionality reduction techniques in the visualisation stage of my project. Dimensionality reduction techniques transform data from a high-dimensional space to a lower one whilst still retaining some of the identity of the original data. This allows analysis on datasets that might have been difficult to interpret without, as this process facilitates visualisation of the dimensionally reduced data.

T-distributed Stochastic Neighbour Embedding

This algorithm, abbreviated t-SNE, is a non-linear dimensionality reduction algorithm. It explores patterns in the data by searching for similarities between data points. This results in similar objects being mapped close together whilst dissimilar objects are spaced further apart. From van der Maaten [11], it converts similarities between data points to joint probabilities and tries to minimise the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. Two important parameters for t-SNE are perplexities and iterations. Perplexity is an estimate for the number of near neighbours each point has. Changes in its value can massively affect the results of a t-SNE plot, with recommended values being between 5-50. The number of iterations can affect the stability of a plot so it is important to modify this parameter until a stable plot can be observed.

Uniform Manifold Approximation and Projection

Abbreviated UMAP, this algorithm is similar to t-SNE with a few differences. As described by Coenen and Pearce in [12] “In the simplest sense, UMAP constructs a high dimensional graph representation of the data then optimises a low-dimensional graph to be as structurally similar as possible.” Its two most important parameters are number of neighbours and minimum distance. The number of neighbours (n_neighbors) refers to the number of neighbouring points used in local approximations of manifold structure. Larger values will preserve the integrity of the global structure whilst losing detail in the local structure. This parameter is recommended to stay in a range of 5-50. Minimum distance (min_dist) refers to the minimum distance between two points in a low dimensional space. Large values result in a more even distribution between embedded points whilst smaller values cause tight embeddings. Compared to t-SNE this algorithm computes faster and maintains the global structure of data better.

Machine learning

I utilised multiple different machine learning algorithms in my project. A variety of algorithms were chosen so that their results could be compared and evaluated afterwards. Additionally, a number of different algorithms could lead to a stronger result when utilised in an ensemble method.

GridSearchCV

This algorithm performs an exhaustive search over specified parameter values for an estimator. The benefit of this algorithm is that it allows tuning of hyper-parameters within an estimator, which refers to parameters not directly learnt within estimators. The remaining machine learning algorithms in this subsection were passed as estimators to this algorithm during the machine learning stage of my project.

Logistic Regression

Despite its name, this refers to a classification algorithm rather than regression. There are multiple variations of this algorithm such as binary, multinomial and ordinal logistic regression. Multinomial logistic regression is used in cases where there are more than two potential outcomes for a target label. This algorithm predicts the probability of a class label by fitting data to a logit function. Simply put this transforms real numbers into a value between 0 and 1, representing the probability. This algorithm was used as it performs well at multinomial regression.

$$f(x) = \frac{1}{1 + e^{-(x)}}$$

Figure 18. A calculation for the logit function

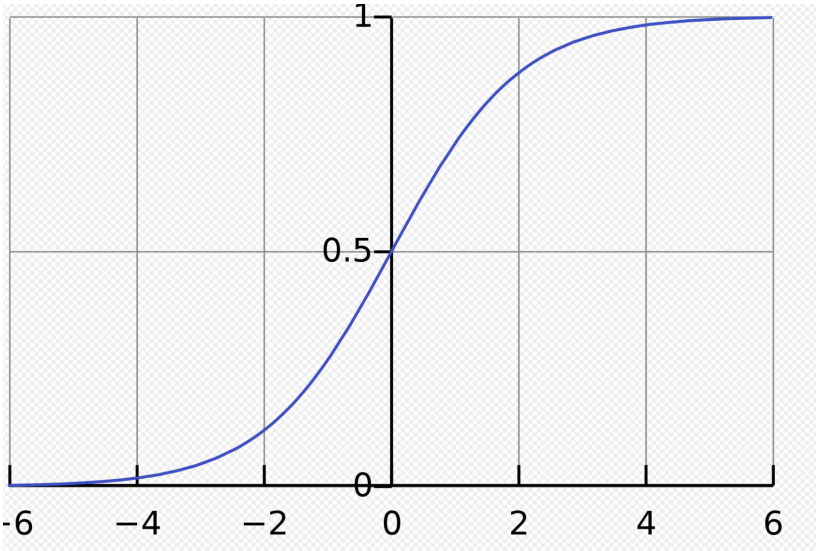

Figure 19. A visualisation for the logit function

SGD Classifier

From the scikit learn website [13] “This estimator implements regularised linear models with stochastic gradient descent (SGD) learning: the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate). SGD allows minibatch (online/out-of-core) learning via the partial_fit method. For best results using the default learning rate schedule, the data should have zero mean and unit variance.” In shorter terms, this algorithm finds the parameters that have minimal convex loss. This algorithm was chosen due to its ability to be tuned with hyperparameters. If it doesn’t perform well then further experimentation can be performed with these hyperparameters to observe if better accuracy can be achieved.

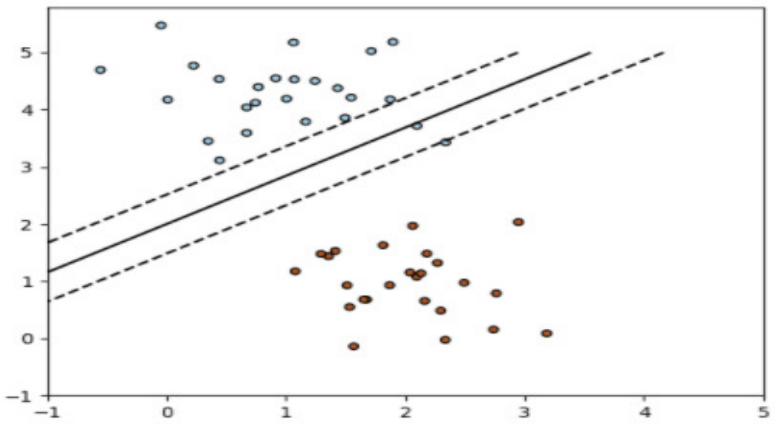

Figure 20. A visualisation of the decision boundary of a SGD Classifier

Decision Tree

This algorithm predicts class labels by learning decision rules from training data. Notably this can be used for both classification and regression problems. The process involves splitting the dataset into distinct nodes. The dataset is continually split and nodes keep getting added until all options for rules are exhausted. The

advantage of this method is that it is easy to understand even with little to no statistical knowledge. This algorithm was chosen due to it being an incredibly popular algorithm in machine learning. Not only does it historically perform well, but using it might allow my results to be comparable to other work in the MGR field.

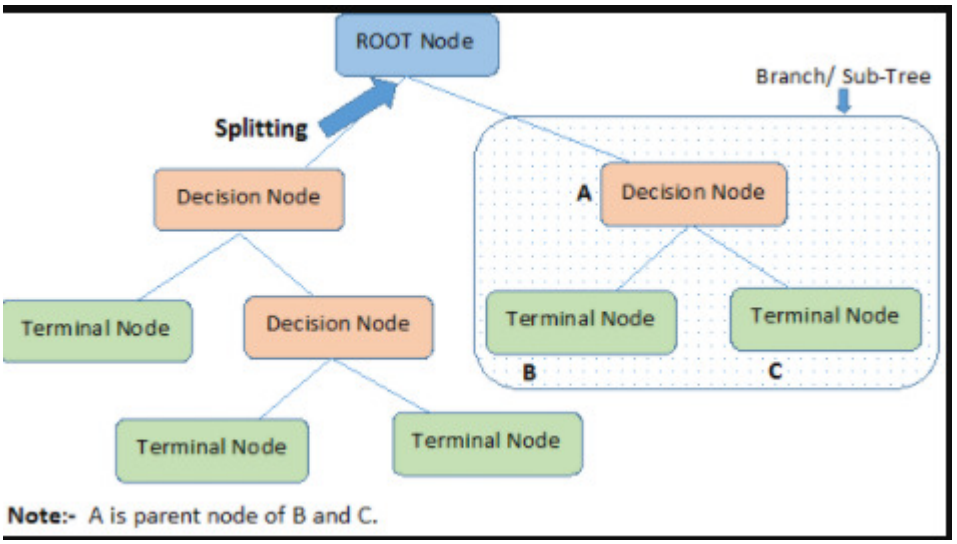

Figure 21. A graph detailing the flow of a decision tree algorithm

Random Forest

This algorithm is based on the decision tree algorithm. It randomly splits the dataset into a number of sub samples and fits a number of decision tree classifiers on them. This method is effective as it still computes quickly on large datasets and is able to accurately predict missing data. A disadvantage of this model is that outside of changing seed, it is difficult to change and predict the behaviour of the model. This algorithm was chosen due to its very high performance. It is very common to use this algorithm as a swiss army knife as it is able to produce good results almost every time compared to other algorithms that are more conditionally dependent.

Support Vector Machine

This algorithm maps the data to a high-dimensional space so that data points can be categorised. Following this a hyperplane is calculated as the boundary whose nearest distance to the nearest element of each tag is the largest. This algorithm was chosen because it doesn't perform well when the margin between classes is fuzzy. The margin between genres can be vague when assessed by humans. Therefore the results of this classifier could provide valuable insight about how a machine's ability to interpret low-level audio features might allow a machine to outperform a human's ability to classify genre.

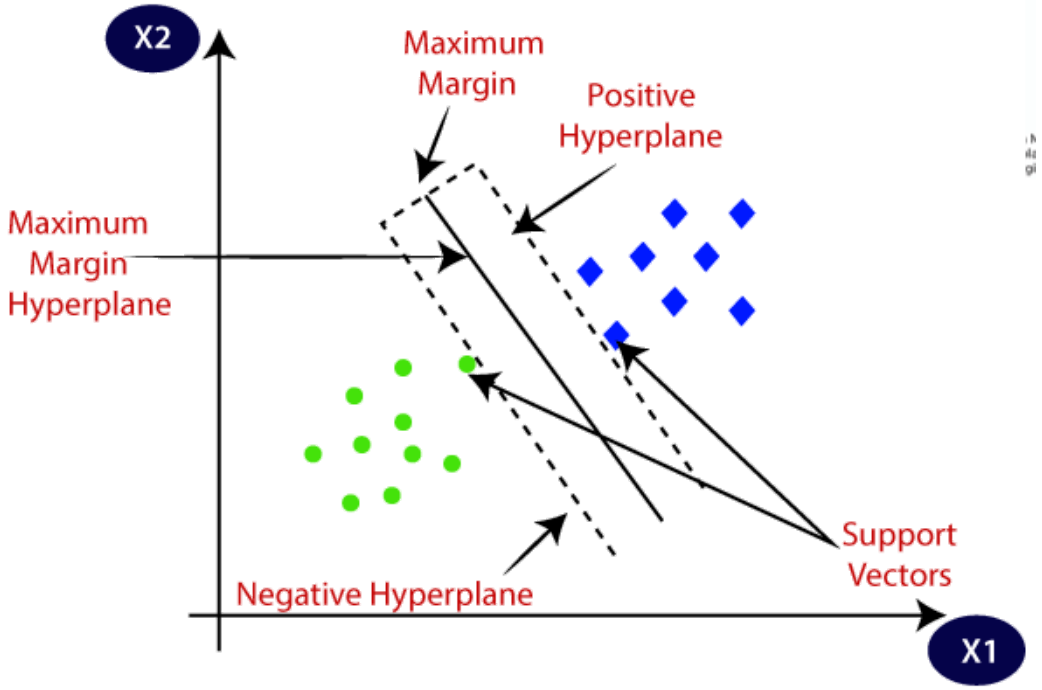

Figure 22. Visualisation of a hyperplane calculated by an SVM

K-Nearest Neighbours

To predict the class label of a given item, KNN identifies a number of neighbours equal to K. It then takes a majority vote from these neighbours with the assumption that points that are close in distance will have similar values. This algorithm was chosen as it performs well with multiclass classification. Additionally, as the boundary between genres can be slim it would be interesting to observe the performance on KNN when identifying similar genres such as hip hop and pop.

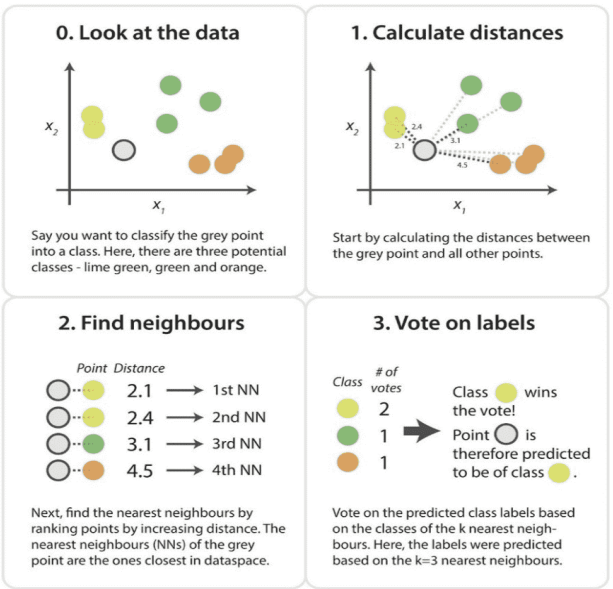

Figure 23. An illustration of how the KNN algorithm works

Multilayer perceptron

From the scikit learn website [14]: “Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function f(.) : R^m -> R^o by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features $X=x_1,x_2, ..., x_m$ and a target y,it can learn a nonlinear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers.” This algorithm was chosen as it has performed very well historically and thus may contribute heavily to an ensemble classifier.

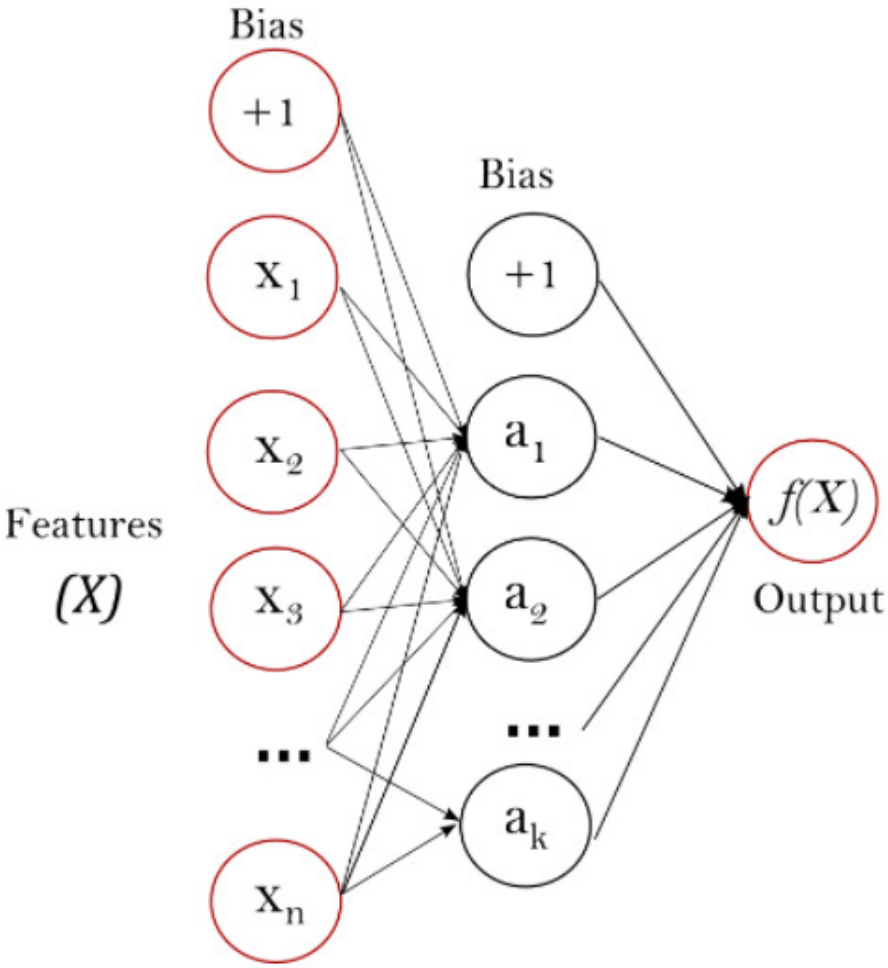

Figure 24. A visualisation of a hidden layer MLP

Feature Scaling

Before carrying out machine learning an important step is to scale the data used. This step is important as some machine learning algorithms could perform poorly without this. For example, distance based algorithms such as KNN use distances between data points to determine their similarity. Without scaling, this could lead to a bias towards features with higher magnitudes.

Standardisation

This involves removing the mean and scaling to unit variance, resulting in a distribution with unit standard deviation.

$$\text{Standardization:}$$
$$z = \frac{x-\mu}{\sigma}$$
$$\text{with mean:}$$
$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)$$
$$\text{and standard deviation:}$$
$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2}$$

Figure 25. The calculation for standardisation

Normalisation

Also known as min-max scaling, this algorithm shifts the values of the data so that they all fall in a range between 0 and 1.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Figure 26. The calculation for normalisation

Train test split

Another important step before carrying out machine learning is splitting the dataset into distinct groups. These groups are a train, test and validation set. The train set is used to fit the dataset whilst the test and validation set are used to evaluate the performance of the machine learning algorithms. This step is important as the classifier must be evaluated on new data rather than what it was trained on, so these sets must be independent of one another. In classification problems the stratify argument can also be passed to the train-test split algorithm. This argument approximates the same percentage of samples of the target class as the complete set.

Ensemble

Ensemble methods combine the predictions of several machine learning algorithms in order to improve upon the accuracy and robustness compared to a single estimator. There are two distinct groups of ensemble methods, averaging and boosting methods. Averaging methods utilise estimators that have been built independently of one another and attempt to average their predictions to obtain a better result. Boosting methods utilise estimators that are built sequentially in an

effort to reduce bias of the combined estimator. In this project, I utilised an averaging method known as a voting classifier.

Voting Classifier

This algorithm takes a number of estimators as its parameters and attempts to combine them by utilising both hard and soft voting. Hard voting classifies items based on the number of votes it receives from its estimators. For example, if four estimators classified an item as belonging to category A and three estimators classified it as belonging to category B, then it would classify the item in category A due to the majority vote. In soft voting, each estimator has a weight that informs the voting classifier how much to "trust" its vote. When an item is being classified, each estimator returns a prediction of its predicted class probabilities which is multiplied by the weight of the estimator. The final class label is then determined by the class label with the highest probability.

Evaluation

In order to evaluate the strength of a classifier an evaluation method must be used. These methods allow the classification process to be clearly visualised so that the behaviour of the classifier and its results become coherent. In this project, I used Confusion Matrices to evaluate the performance of my classifier.

Confusion Matrix

This refers to a table that shows both the actual class value and the predicted class values for a classifier. It provides easy visualisation to determine where a classifier is becoming confused between two classes and can provide insight on how to resolve these issues.

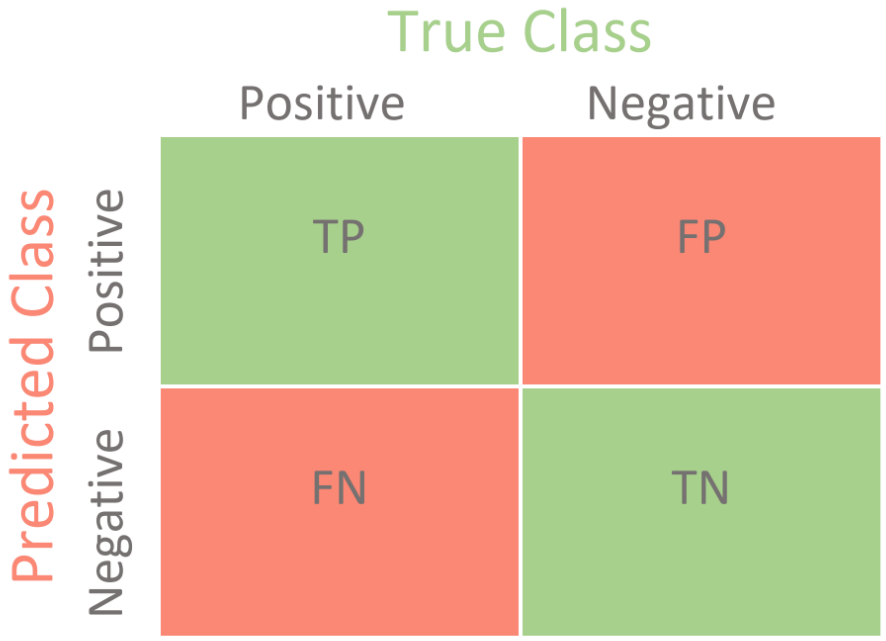

Figure 27. An example confusion matrix

List of tools and resources

Datasets

A number of datasets were considered for this project. The requirements for my dataset were to have a solid variety of genres and a large sample size of songs. Furthermore, I wanted to use an established dataset in order for my results to be comparable to previous work. Table 1 shows the advantages and disadvantages of these considered datasets and an evaluation whether it should be used in this project, future work or dismissed entirely.

Name of dataset	Advantages	Disadvantages	Evaluation
GTZAN	Used frequently historically, allowing comparisons between my work and others.	Limited range of artists. Some files are incorrectly labelled.	This will be used in the project.
Million Song Dataset	Incredible size and quality.	Audio is not directly available. Size of the dataset could lead to extremely long computation times.	This could be used in a future version of the project when there is more time for testing.
MTG-Jamendo	Very good size boasting 55000 songs. Large variety of genres.	Genre distribution is skewed, with electronic music comprising 30% of the total tracks.	The issue of genre distribution holds this dataset back too much for music genre recognition. Therefore this dataset will not be used in this project.
FMA	Over 100000 songs and 161 genres. Comes in four different sizes.	Genres are unbalanced in the larger versions of this dataset. The music in this dataset can be more indie, meaning it is less mainstream.	This would be the next dataset used in a future version of the project. For now GTZAN will be used due to its historical value but this dataset would be very interesting to compare with it.

Table 1. A discussion of considered datasets

GTZAN

This dataset contains ten genres with a hundred songs each. These genres are: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae and rock. It is used in many investigations into MGR as mentioned previously in this report. It is not without its shortcomings however. Sturm [15] dissects these issues, explaining that artist repetition and some incorrect labelling lead to imperfect results. I chose to still use GTZAN despite these issues as I believed it would allow me to gain insight from previous work that utilised this dataset and identify potential issues in my own project.

Programming language

The programming language used in this project was Python version 3.8.5, using Jupyter Notebook as an IDE.

Libraries

A number of libraries were used to assist various aspects of the project. Some of the most important libraries are detailed in the list below.

- Librosa, used for music and audio analysis.
- Sklearn, primarily used for machine learning.
- Numpy, primarily used for mathematical functionality.
- Pandas, used for data analysis and visualisation.

Code resources

During different stages of my project, I utilised existing code to model my implementation on. To complete the feature extraction and machine learning stages of my project I adapted methods from Guimarães [16]. The visualisation stage was adapted from previous work by Shanbhag [17]. An explanation of these algorithms is given at the start of the next section of this report.

Implementation

In order to carry out this project I utilised the algorithms, tools and resources detailed in the previous section to carry out my proposed solution that fits my requirements. My implementation also required me to create a number of my own methods and algorithms which are detailed throughout this section. Following this is an explanation of issues encountered whilst carrying out the project and how I resolved them.

My algorithms

This section details the methods and algorithms I created during my implementation. These are divided into three sections based on different stages of the project.

Feature extraction

Get_features

This method iterates through a dictionary containing my desired audio features and obtains the max, min, mean and standard deviation for each item.

Read_process_songs

This method iterates throughout the entire GTZAN directory and loads each file via librosa. It then calls the Get_features method to append the statistical information about each feature to an array.

Visualisation

Get_tsne_embeddings

This algorithm obtains the t-SNE embeddings for the dataset. It also normalises them after obtaining them. The reason for this method is to make it easier to retrieve the embeddings for various perplexity and iteration values.

Get_umap_embeddings

Functionally the same as Get_tsne_embeddings but retrieves the UMAP embeddings. Uses the n_neighbour and min_dist parameters instead.

Plot_components

This method iterates through a dictionary of embeddings and plots their graphs.

Classifier

Plot_confusion_matrix

This algorithm plots a confusion matrix. The code was adapted from the scikit learn website [18].

Problems encountered

During the feature extraction stage of the project I discovered that there was an error when attempting to compile my code. Further research allowed me to discover that a file in the GTZAN directory is corrupted, jazz054.wav. I replaced this file with another that I found posted by a user facing the same issue on the Kaggle website [19]. Additionally during the feature extraction stage, I attempted to take additional statistical features from each audio feature. I tried extracting the skew and kurtosis from each of the feature vectors but this resulted in errors on compiling meaning I had to eliminate these features.

Results and Evaluation

Empirical observation of clusters obtained

After plotting graphs for both t-SNE and UMAP in 2 and 3 dimensions I obtained the following results:

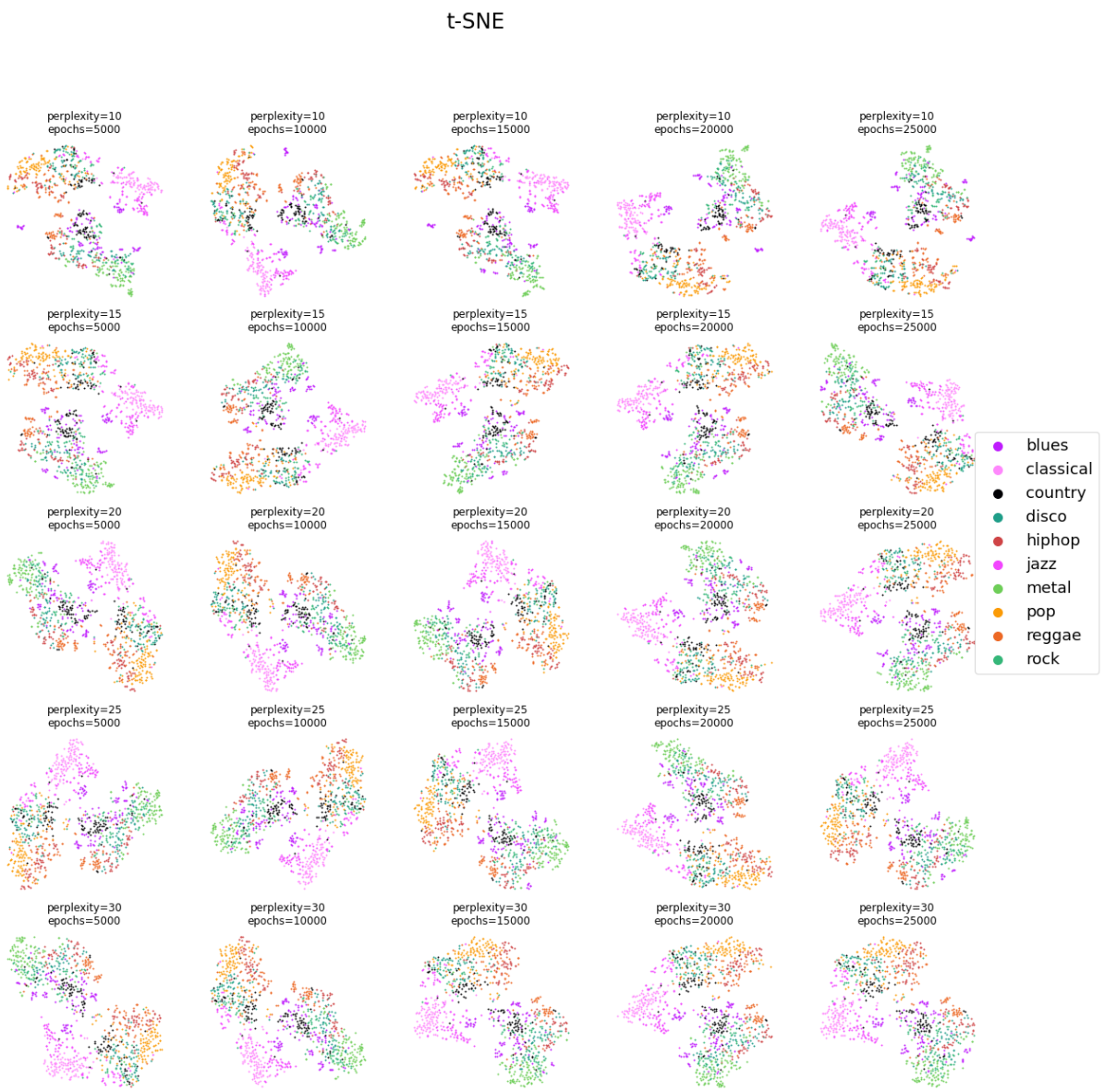

Figure 28. 2D representation of the t-SNE algorithm performed on my dataset for various perplexities and iterations

UMAP

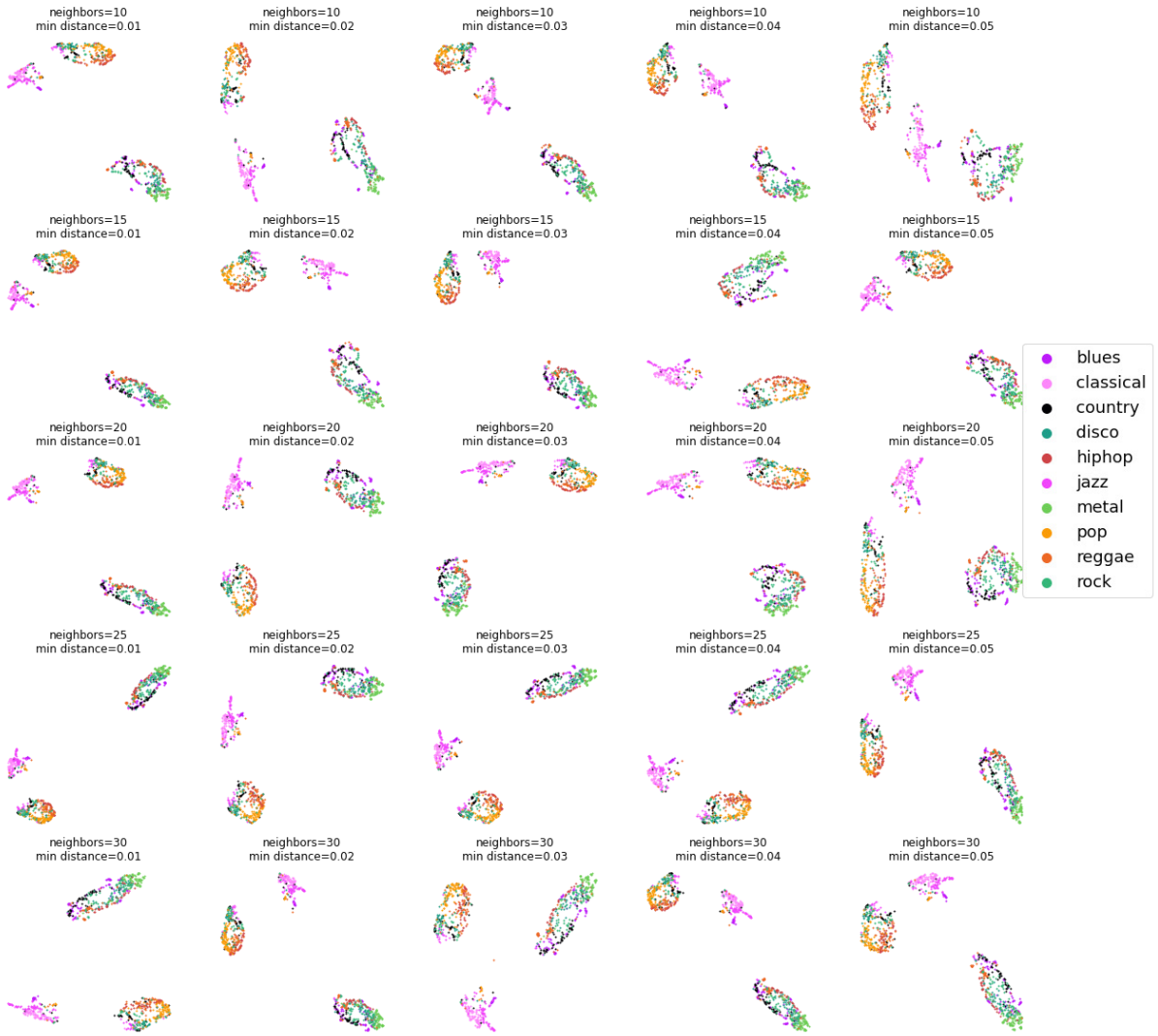

Figure 29. 2D representation of the UMAP algorithm performed on my dataset for various neighbours and min distance (parameters)

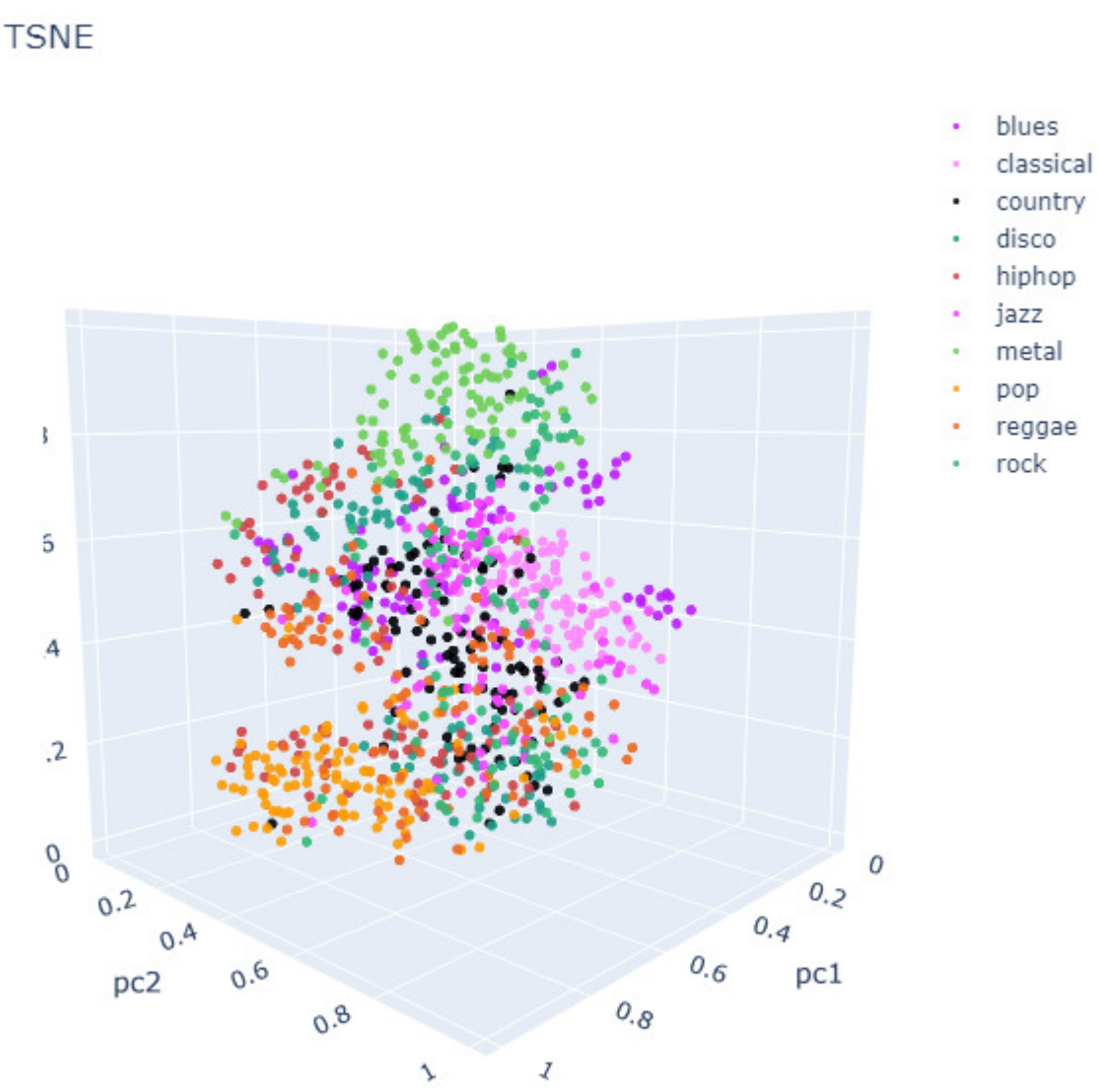

Figure 30. 3D representation of the t-SNE algorithm performed on my dataset

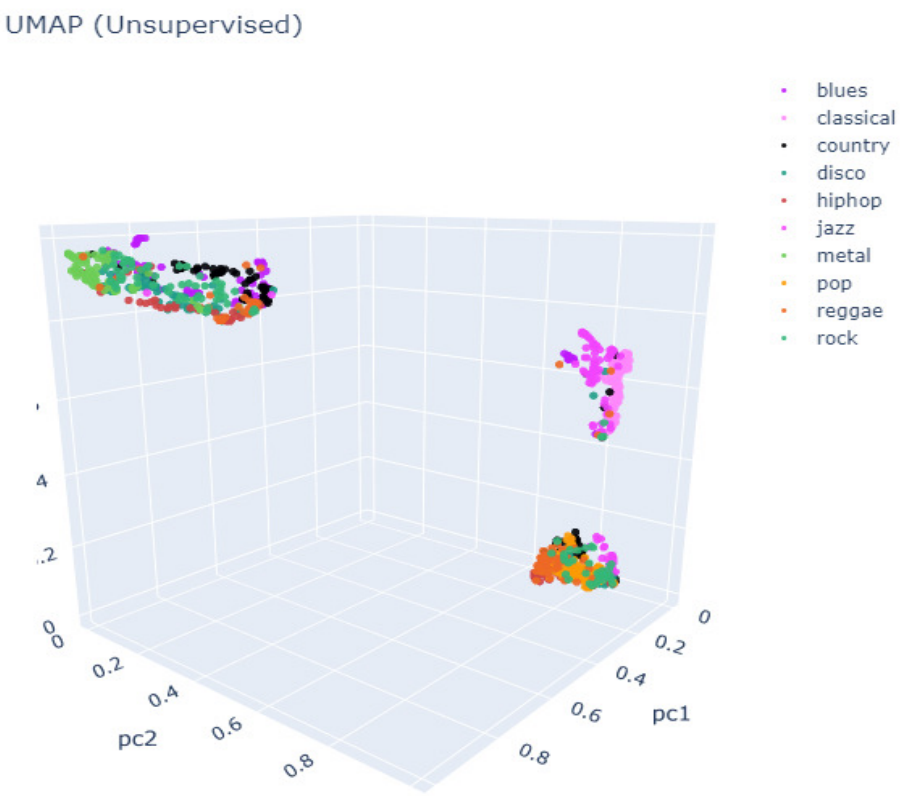

Figure 31. 3D representation of the UMAP algorithm performed on my dataset

As can be seen in figures 28-31, the behaviour of my data is suitable for a classifier. Figure 28 demonstrates this with clarity, as for each combination of epoch and perplexity we can observe the same pattern of genres clustering together. Similar genres such as hip-hop and pop tend to overlap in each of the above figures. This is a useful observation to make as this is a trend I expected to see before I began the visualisation stage since I know from listening to these genres that they share a lot of musical features between songs. Therefore observing these patterns in my data led me to believe that my dataset had merit in being utilised in building a classifier since it appeared how I believed it would initially. One interesting observation from the above results is the behaviour of the classical and blues genres. In all of the above figures I observed that these two genres would usually cluster together distinctly away from the other genres. These clusters also occasionally had overlapping items, leading me to conclude that there are similarities between blues and classical music which I had not considered prior to this project.

Final testing

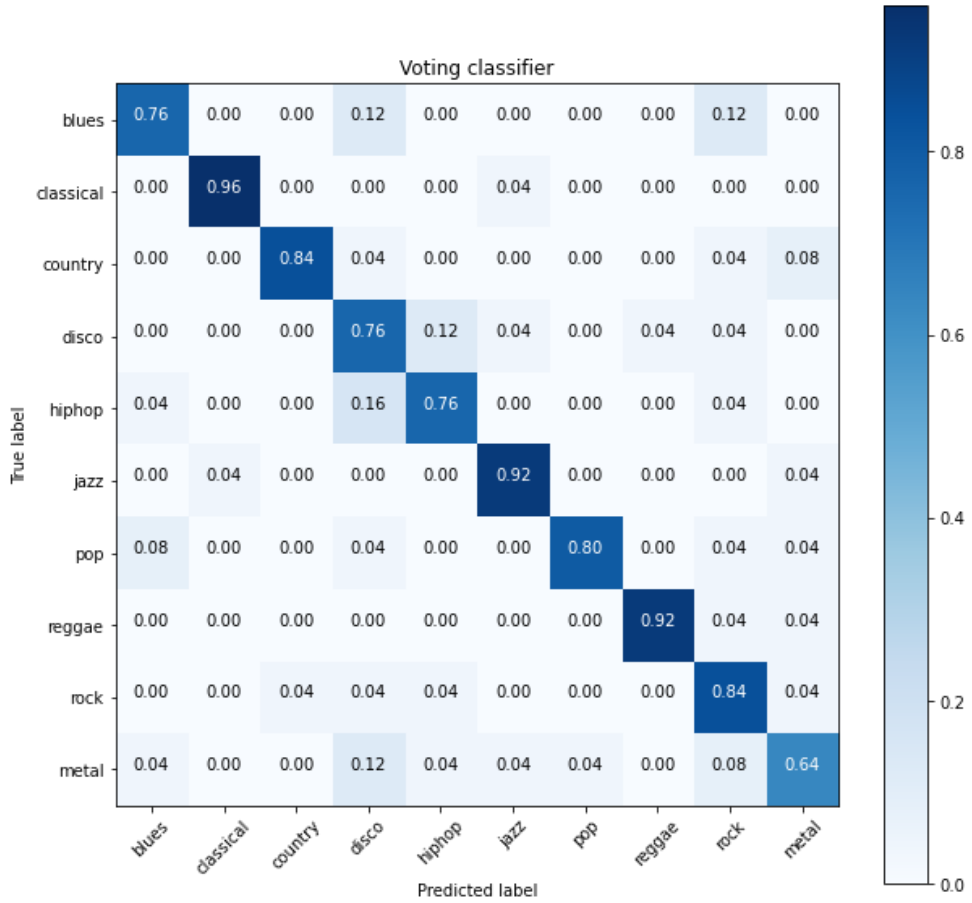

Figure 32. The confusion matrix obtained by the voting classifier on my dataset

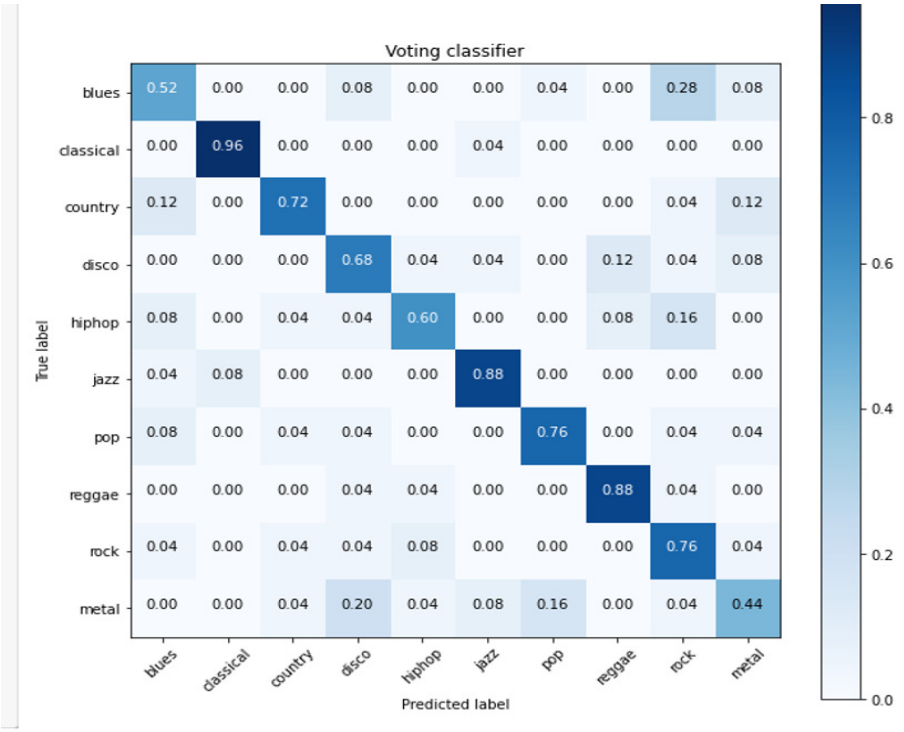

Figure 33. The confusion matrix obtained by the voting classifier on just the MFCC feature's values

Figure 32 is the confusion matrix produced by the voting classifier on my dataset. I was initially quite pleased with the results, achieving a minimum of 64% for each genre and maximum of 96% success rate for the classical music items of the dataset. Additionally, the mean accuracy across all genres was 82%, which exceeded my aim for the project. I would have preferred for the minimum accuracy of my results to also exceed 70%. However, it is worth noting that the errors within the GTZAN dataset mentioned previously may inhibit the accuracy of my classifier on certain genres such as metal. I experimented with manual feature extraction in an attempt to improve the minimum accuracy of my results. To begin, I isolated each feature and extracted a csv file that contained purely the results of that feature within the GTZAN dataset. I then rebuilt my classifier for each isolated feature and made a note of the results of the confusion matrix produced by the voting classifier for each feature. These results can be found in the appendices of this report. I then identified which of the features produced the best results on their own in order to estimate which combination would provide the most accurate genre classification. The criteria for my selection was to identify which features produced the best minimum accuracy as logically a combination of them could lead to a more accurate classifier than one that included the less useful features. However, upon looking at the results I noticed that the majority of the features always had at least one genre that it struggled to contribute towards identifying.

For example, figure 33 as shown above is the result when using only the MFCC values as data. This was by far the best performing feature as no other feature came remotely close to boasting the same minimum accuracy score for each genre. That being said, it still struggles to inform the classifier how to predict when a piece of music belongs to the metal genre with an accuracy score of 44%. Therefore I adjusted my criteria to be features that had a good minimum accuracy score as well as features that could help in identifying genres that the other selected features were weak at informing the classifier about. For this reason, I attempted to pick what I believed were the three most useful features: MFCC, Chroma and Bandwidth. These features all had an adequate minimum accuracy score and the chroma and bandwidth features were able to identify genres that the other had low accuracy scores on. This led to the results of figure 34, shown below.

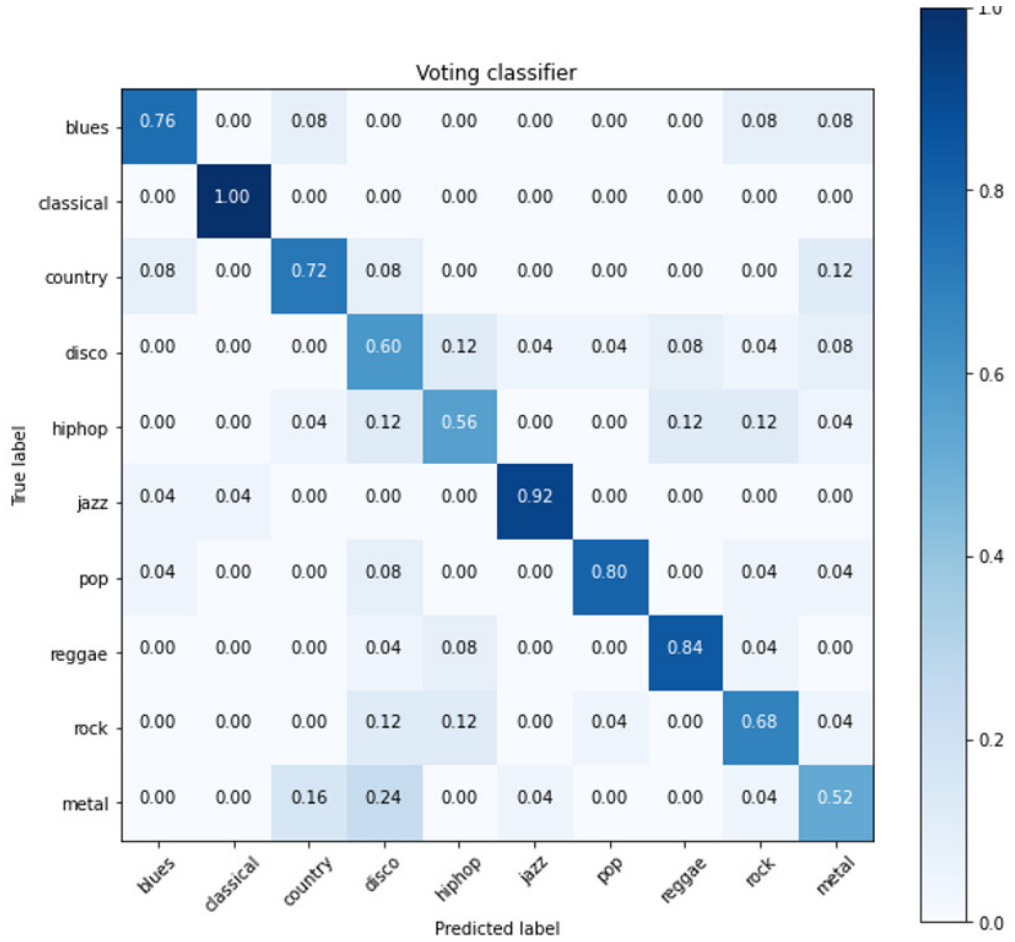

Figure 34. The confusion matrix obtained by the voting classifier on the values for MFCC, Chroma and Spectral Bandwidth

Whilst these results were worse on average than using all the features combined, it still provided valuable insight. The classical genre had been the most easily identified genre in almost all the confusion matrices I had viewed at this point but the accuracy score for this combination of features was able to accurately predict it 100% of the time. This was even better than the 96% accuracy provided by utilising all the features which led me to believe that there was a combination of features that could outperform all the features combined. I carried out further testing, trying to add features that did well at informing the classifier how to predict the metal and hip hop genres since they were the poorest performers. However I found that adding features into my combination did not always improve the final accuracy, with numbers fluctuating up and down as more features were added. I experimented with adding the Zero Crossing Rate and Flux to my combination of features but the results did not improve. The results of these experiments can be found in Figures 35 and 36 below for clarity.

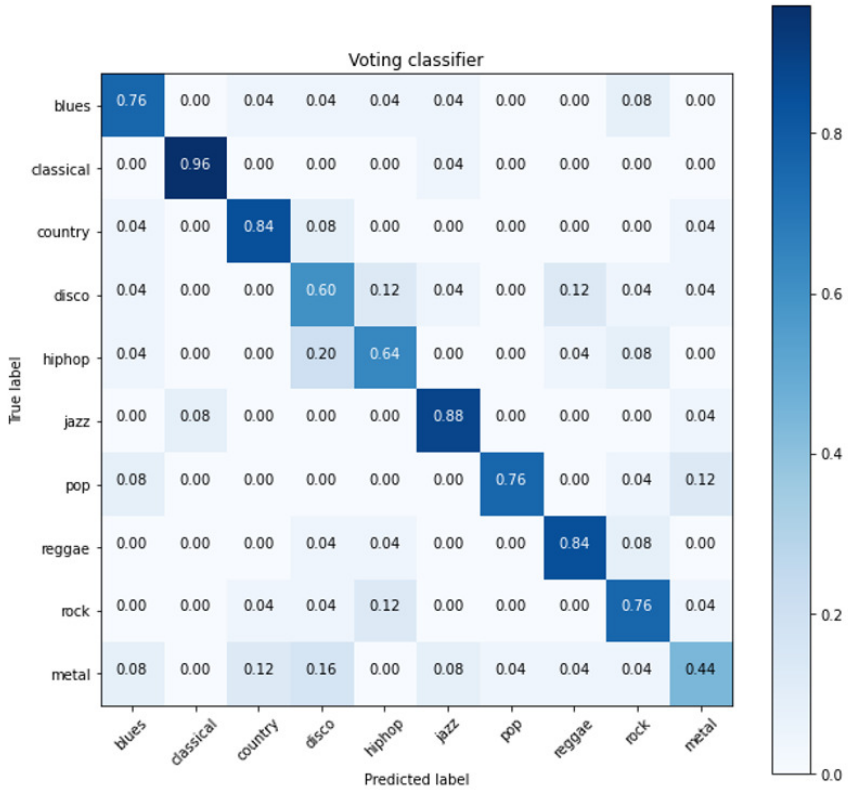

Figure 35. The confusion matrix obtained by the voting classifier on the values for MFCC, Chroma, Spectral Bandwidth and Zero Crossing Rate

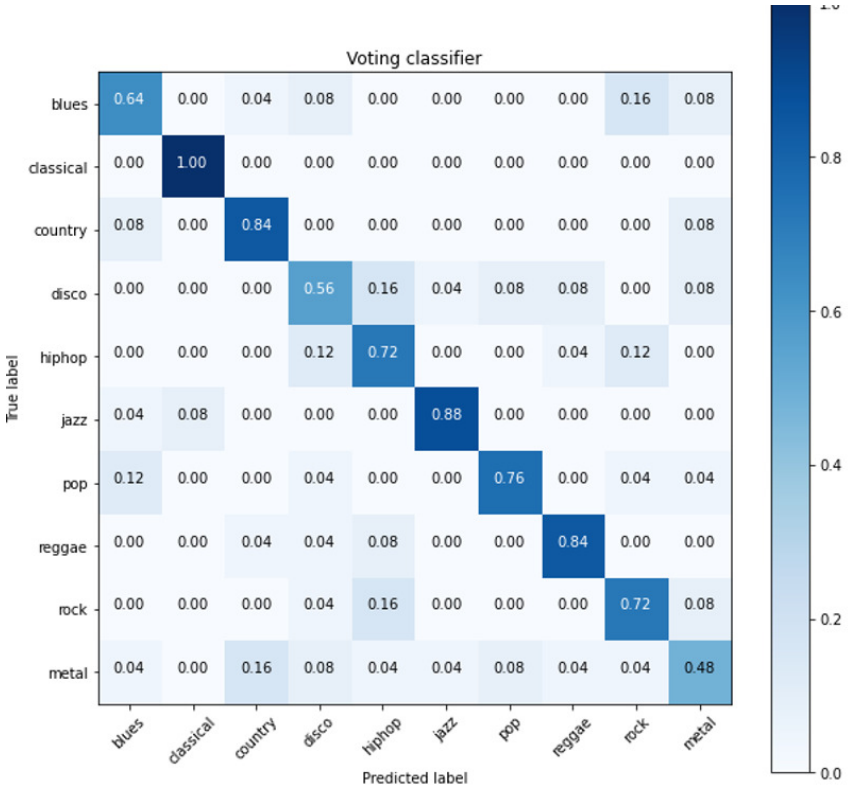

Figure 36. The confusion matrix obtained by the voting classifier on the values for MFCC, Chroma, Spectral Bandwidth,Zero Crossing Rate and Spectral Flux

Conclusions

The initial goal of this project was simple, to establish whether it was possible for a computer to correctly identify a genre of a piece of music. This goal was achieved quickly following some background research into this project. However this was not the sole focus of the project as the two main goals I had were predicated on the idea that it was possible for a computer to identify music. These goals were to build a classifier with a minimum accuracy of 70% and to identify the best combination of features that could be used as a dataset to help build said classifier. The former of these goals was achieved by utilising a voting classifier on a dataset composed of eleven different features to obtain a mean accuracy of 82%. It's also worth noting that the identification of the metal genre held back the minimum accuracy of the classifier to 64%. The project provided a lot of insight about the field of music genre recognition and future research could improve this further. In terms of finding the best combination of features, I managed to evaluate that there is a combination of features that could outperform the results of my voting classifier on all the features. However, due to not completing my experiments with feature selection; I have yet to find a combination of features that achieved a higher minimum accuracy across all genres than using all features combined. I did establish that it was possible to identify the classical genre 100% of the time which gave credence to the idea of an optimal combination of features existing that I hadn't found yet.

In conclusion, this project made a good start on establishing its goals and at this point in time has the potential to be useful in the field of MGR. With future work and research the accuracy of the classifier could be improved and more insights could be gained about the impact of certain features on the classifier.

Future Work

Feature selection

The results of my project were promising, but remained inconclusive due to time constraints. There is certainly room for further work, research and improvement that I could take on in the future. Due to time constraints I wasn't able to experiment with as many combinations of features as I would have liked to. Ideally, I would keep experimenting with feature selection until I found the most optimal combination of features to build a classifier with. At one point in my research I tried to implement the SelectKBest algorithm which is used to assist with feature selection. However, it delivered worse results than me manually selecting a combination of features.

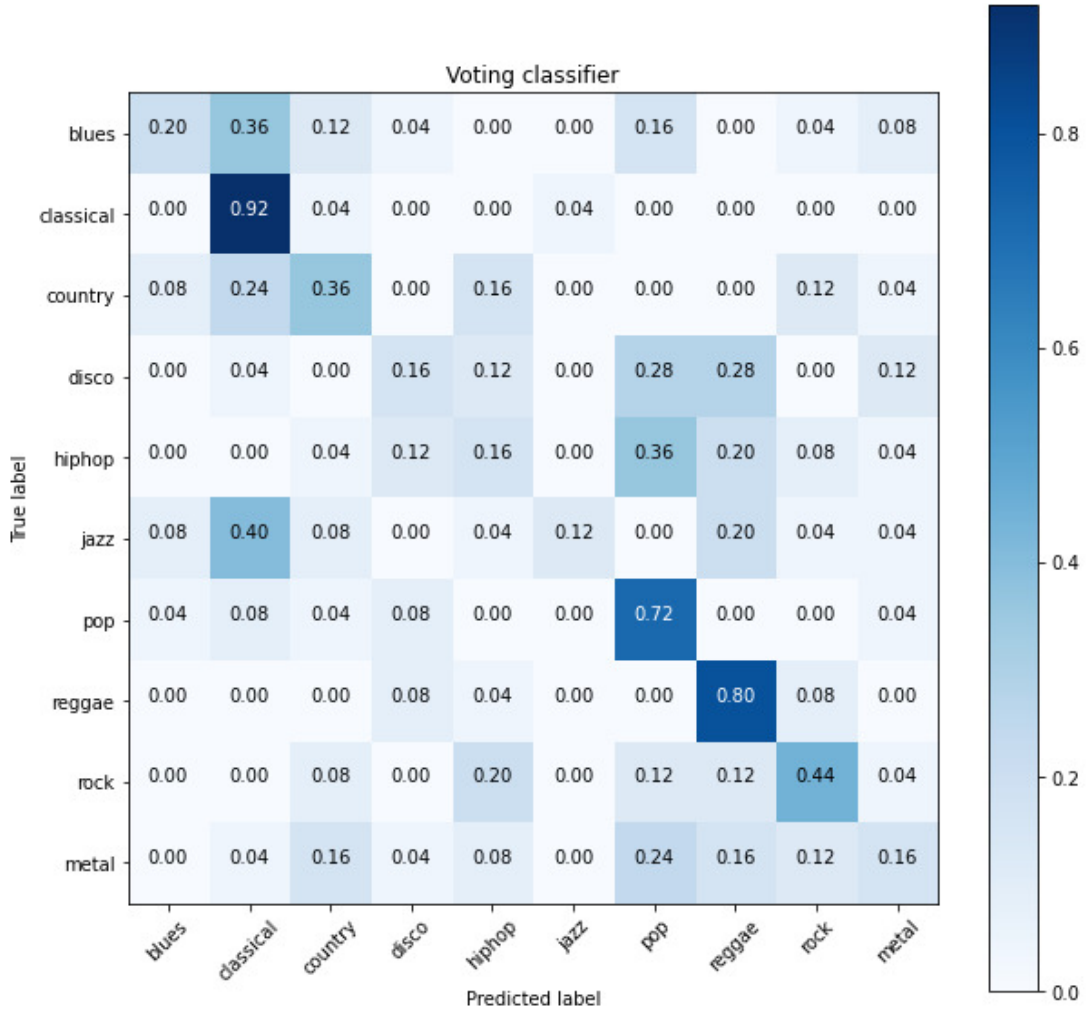

Figure 37. The confusion matrix obtained by the voting classifier after the SelectKBest algorithm was implemented

Figure 37 shows the result of the voting classifier after the algorithm had attempted to select the four best columns of data to train and test the classifier on. As you can see, it still performed well on genres such as classical, pop and reggae but on other genres its accuracy was subpar. It is likely that there is an algorithm that I could implement that would enable me to determine the best selection of features with greater success but due to lacking time and expertise it is not possible for me to implement this until further in the future.

Naive bayes implementation

In a future iteration of my project there is potential for me to improve the accuracy of my classifier. To do so, I would need to implement naive-bayes theorem within the weighting of my voting classifier. In my current implementation, the weighting of the classifier is very simplistic as it only takes into account the overall accuracy of each algorithm. However I observed in my results that often some algorithms were better at predicting specific genres of music than others. If the weighting instead implemented naive bayes theorem then each genre could be evaluated based on previous evidence by that algorithm. If an algorithm was very good at predicting

classical music for example, then it would have a heavily weighted vote within the voting classifier when evaluating classical music.

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$

Figure 38. The calculation for naive bayes theorem

However, at this time I cannot think of a way to code this behaviour in my implementation. Whilst I can grasp the idea of naive-bayes theorem I don't understand it enough to be able to articulate my intentions with it via code. Even with research into the topic I still struggled to find a suitable way in which I could implement this even though it could vastly improve the overall accuracy of my classifier. A question that remains unanswered about my project is whether the lower accuracy when predicting the metal genre is a result of faults within the GTZAN dataset or an issue with my classifier. A good implementation of naive bayes theorem could help in answering this question in future research.

Evaluation methods

Additional evaluation methods could be used to evaluate the accuracy of my classifier. In future research, methods such as k-fold cross validation could be implemented to complement the confusion matrices I already used. This would give additional clarity about how my classifier performs and potentially provide insight on how to improve it.

Dataset experimentation

It might be worth experimenting with additional datasets in the future in order to determine if the faults within the GTZAN dataset compromise the robustness of my results. New results could easily be compared to my existing results to determine if my classifier actually struggles to identify genres such as metal or if it is a result of the faults within the GTZAN dataset.

Standalone program

Finally, an inessential piece of work that could follow this project is an implementation of a standalone program to identify genre. One of my initial ideas for a project was to create a program that allowed a user to submit an audio file on their computer which would be categorised into a musical genre. After researching this idea further I realised how lengthy the process would be to create a classifier that was able to accurately predict the genre of a piece of music which led me in the current direction of my project which is more scientific. However I still hoped to be able to implement this after I had built an accurate classifier as I thought it would be a useful standalone addition to the project. Unfortunately due to time constraints and my classifier not being as accurate as I had hoped it to be I never implemented this standalone program. In future developments of this project I could return to this idea. The program does not necessarily have to only interpret the genre of a piece of music as some users may find it interesting to learn about the other features of a song. Music enthusiasts might enjoy learning about the similarities in tempo between two different genres of music for example, so having statistics in addition to the genre of a song returned to the user upon evaluation could result in a niche but interesting standalone program that I would enjoy developing in the future.

Reflection on Learning

I felt that this project had a great impact on my thought process towards learning. When I began, I knew very little about implementing machine learning and almost nothing about the process of analysing an audio file. Within a few weeks of beginning the project however, I had quite a lot of confidence in my planned approach and felt that I had the resources and knowledge to complete the project on time. I believe that this change in mentality stemmed from the structure of this final project. By meeting with my supervisor every week I was able to consolidate all the knowledge from the last seven days and consistently validate that the current direction of the project was viable. In previous work I've encountered issues with procrastination due to a lack of confidence in my ability to complete certain assignments. Therefore having a member of staff who was able to reassure me that my current abilities are what is expected of me at this point in time allowed me to confidently and consistently produce quality work week on week. I also would argue that this structure made me challenge my preconceptions towards aspects of this course. Due to the impact this project will have on my final grade I treated the meetings with my supervisor with far more respect than I have with other aspects of the course in the past. For example, I've chosen to catch up on a lecture via panopto at a later date rather than attend it in person in the past because I'd evaluated that it wouldn't impact my learning too heavily. In contrast, I learnt to discipline myself enough to attend all meetings with my supervisor where extenuating circumstances didn't apply. I had the realisation that not attending these meetings could only hinder myself and so I attended even if I felt I hadn't produced an adequate amount of work for that week. This attitude towards learning was not limited to my studies. I found

that this change in approach also spilled over into other aspects of my life such as exercise. I began treating activities that would only hinder me if I skipped them such as going to the gym with the same respect that I did the weekly supervisor meetings, leading to me disciplining myself to achieve all the tasks that I needed to do every day even when I didn't feel like it.

My preconceptions about this project from the initial plan have been challenged somewhat. For example my work plan aimed to evaluate whether a computer could interpret genre from an audio file by week eight of the project. In reality, I discovered that this was possible within minutes of background research when I began seriously working on this project. At the time I found this alarming as it made me question the naivety of my thought process just weeks earlier. However, upon reflecting upon this now I realise that my actions in the past will always be less informed than I am in the present. Even my approach to this project might seem flawed or overcomplicated when I reflect upon it in a few months time because the knowledge I have then will surpass my ability now. I realise now that it is important to always challenge your preconceived notions if you want to improve your work as becoming complacent leads to a stagnation in output. As an individual I acknowledge that sometimes I assume my way of doing things is the correct way, but I seek to improve this by reevaluating what I do after I do it. I won't exaggerate that this project brought about this revelation entirely, but it certainly played a role in allowing me to understand a method to gain a more beneficial relationship with learning as a whole.

References

1. Tzanetakis, G. and Cook, P. 2002. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing* 10(5), pp. 293–302. doi: 10.1109/tsa.2002.800560.
2. Olteanu, A. [a]. GTZAN Dataset - Music Genre Classification. Available at: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification [Accessed: 4 May 2022].
3. Tao Li and Ogihara, M. [C. 2005]. Music Genre Classification with Taxonomy. In: *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.* IEEE. Available at: http://dx.doi.org/10.1109/icassp.2005.1416274 [Accessed: 4 May 2022].
4. Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao and Lian-Hong Cai [no date]. Music type classification by spectral contrast feature. In: *Proceedings. IEEE International Conference on Multimedia and Expo*. IEEE. Available at: http://dx.doi.org/10.1109/icme.2002.1035731 [Accessed: 5 May 2022].
5. Saranga-K-Mahanta-google 2021. Audio Feature Extraction. Available at: https://devopedia.org/audio-feature-extraction [Accessed: 5 May 2022].

6. Education, I.C. 2020. What is Supervised Learning? Available at: https://www.ibm.com/cloud/learn/supervised-learning [Accessed: 9 May 2022].
7. Müller, M. 2015. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer.
8. Shah, A., Kattel, M. and Nepal, A. 2019. ResearchGate. Available at: https://www.researchgate.net/publication/330796993_Chroma_Feature_Extraction.
9. Chathuranga, D. and Jayaratne, L. 2013. Automatic Music Genre Classification of Audio Signals with Machine Learning Approaches. *GSTF Journal on Computing (JoC)* 3(2). doi: 10.7603/s40601-013-0014-0.
10. Lyons, J. Practical Cryptography. Available at: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/ [Accessed: 8 May 2022].
11. TechTalks, G. 2013. Visualising Data Using t-SNE. *YouTube* . Available at: https://www.youtube.com/watch?v=RJVL80Gg3lA&list=UUtXKDgv1AVoG88PLl8nGXmw. [Accessed: 8 May 2022].
12. Understanding UMAP. Available at: https://pair-code.github.io/understanding-umap/ [Accessed: 9 May 2022].
13. sklearn.linear_model.SGDClassifier.Available at: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html [Accessed: 9 May 2022].
14. 1.17. Neural network models (supervised). Available at: https://scikit-learn.org/stable/modules/neural_networks_supervised.html [Accessed: 9 May 2022].
15. Sturm, B.L. 2013. The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. Available at: https://arxiv.org/abs/1306.1461.
16. Guimarães. Available at: https://notebook.community/Hguimaraes/gtzan.keras/nbs/1.0-handcrafted_features [Accessed: 9 May 2022].
17. vinayshanbhag 2020. dimensionality-reduction-TSNE-UMAP. *Kaggle* 13 July. Available at: https://www.kaggle.com/code/vinayshanbhag/dimensionality-reduction-tsne-umap/notebook [Accessed: 9 May 2022].
18. Confusion matrix. Available at: https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html [Accessed: 9 May 2022].

19. Olteanu, A. [b]. GTZAN Dataset - Music Genre Classification. Available at: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification/discussion/158649 [Accessed: 9 May 2022].

Table of abbreviations

Abbreviation	Explanation
MGR	Music genre recognition
t-SNE	T-distributed stochastic neighbour embedding
UMAP	Uniform manifold approximation and projection
RMSE	Root-mean-square-energy
MFCC	Mel-frequency cepstral coefficient
ZCR	Zero-crossing rate
SGD	Stochastic gradient descent
SVM	Support vector machine
KNN	K-nearest neighbour
MLP	Multi-layer perceptron

Table 2. A table of abbreviations used throughout this report

Appendices

Appendix 1: Confusion matrices for various algorithms on full dataset of features

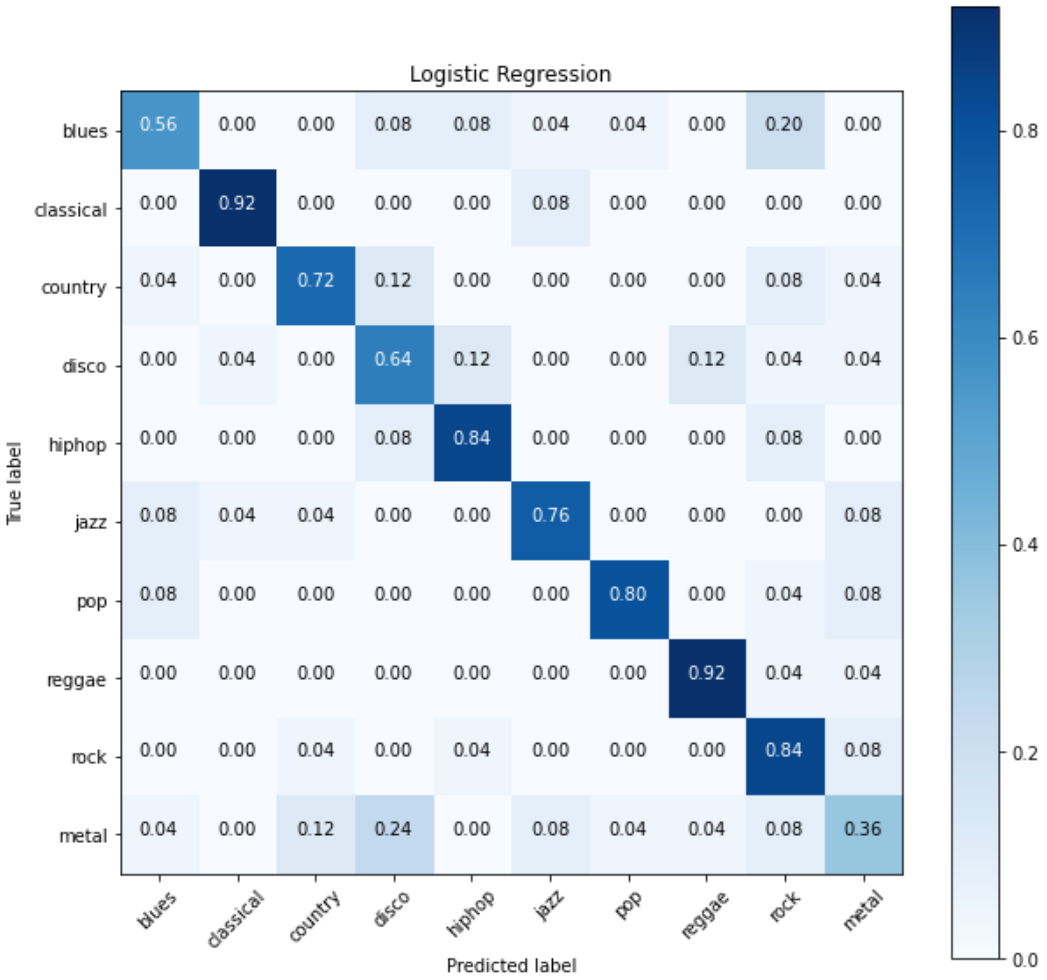

Figure 39. The confusion matrix for the Logistic Regression algorithm on the full suite of features

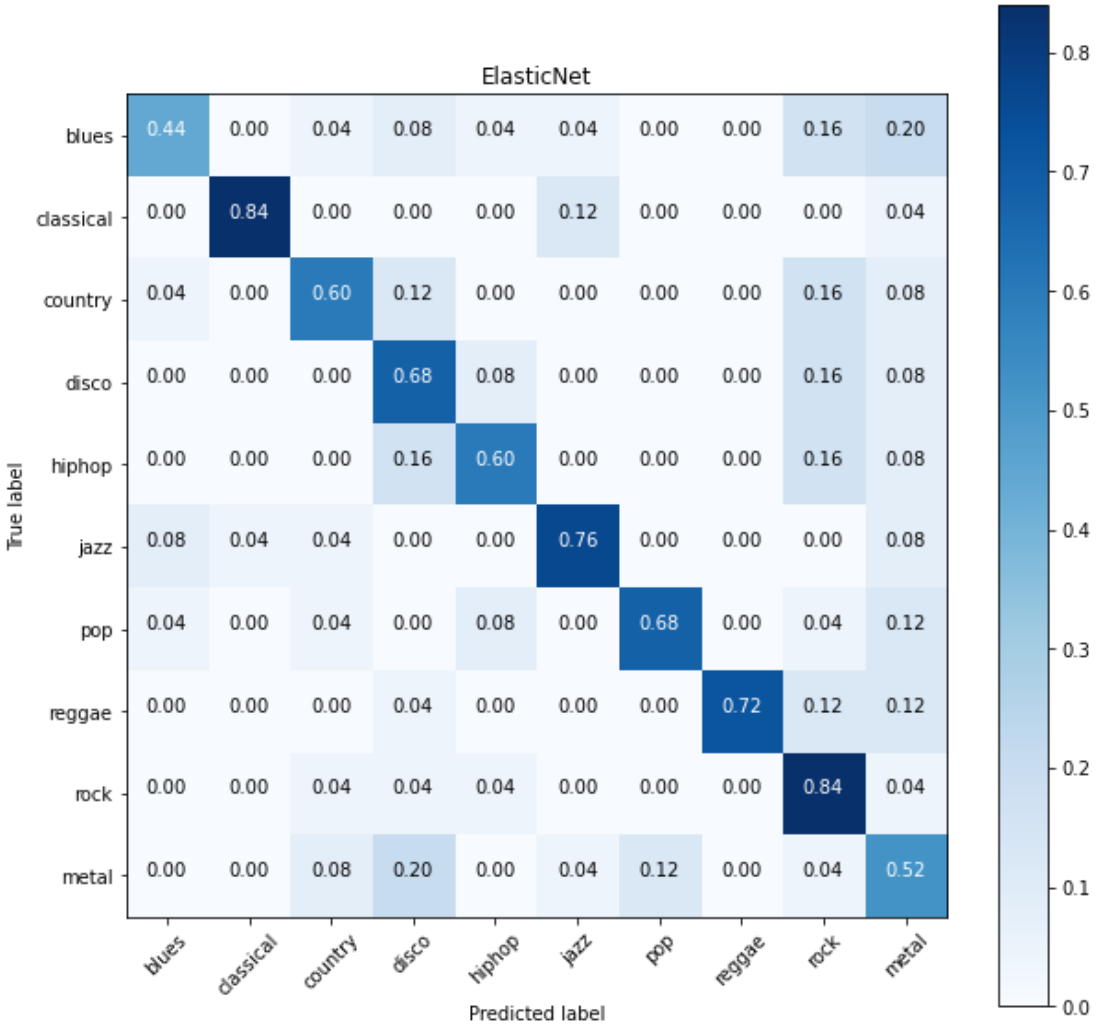

Figure 40. The confusion matrix for the SGD Classifier algorithm on the full suite of features

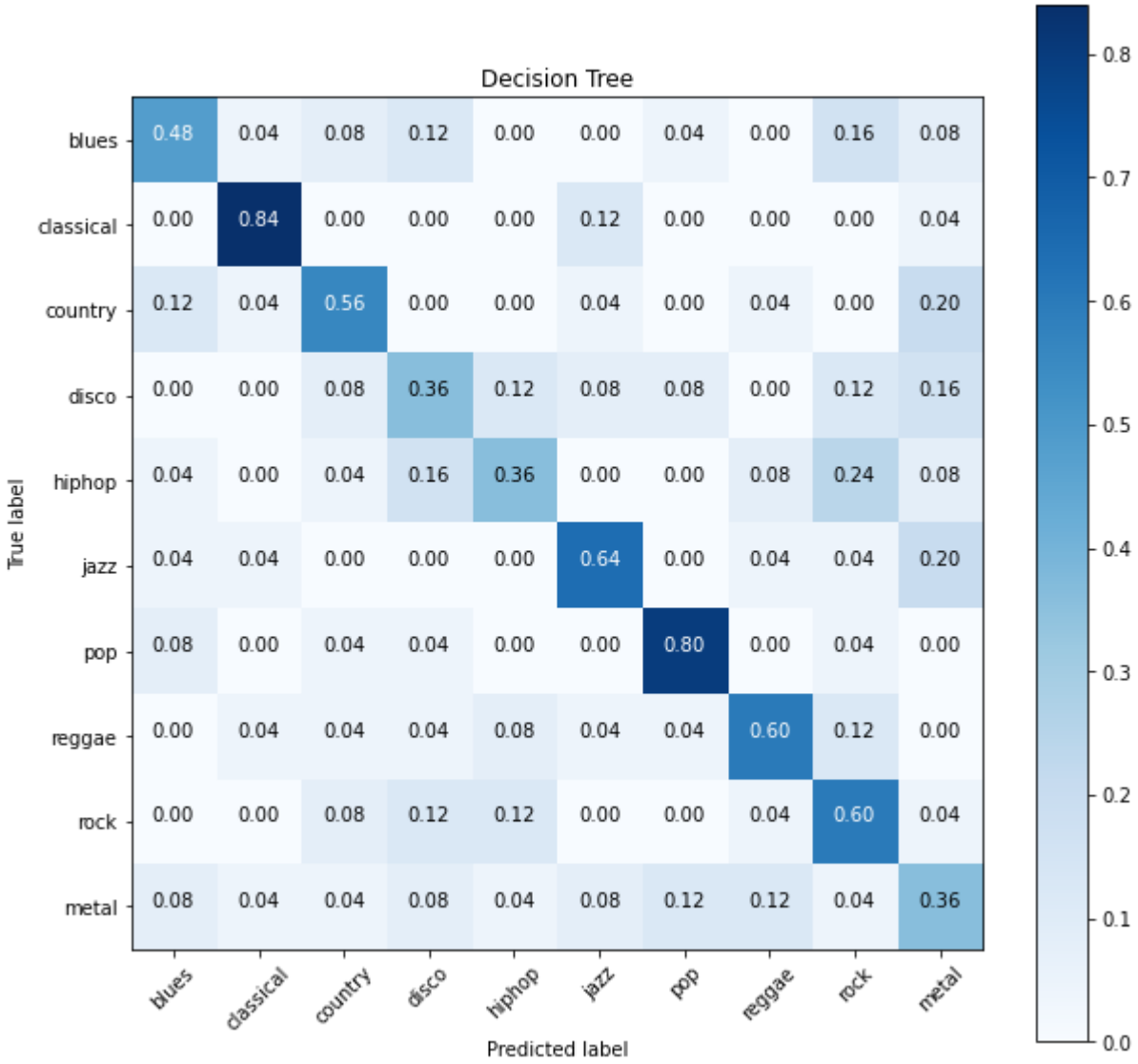

Figure 41. The confusion matrix for the Decision Tree algorithm on the full suite of features

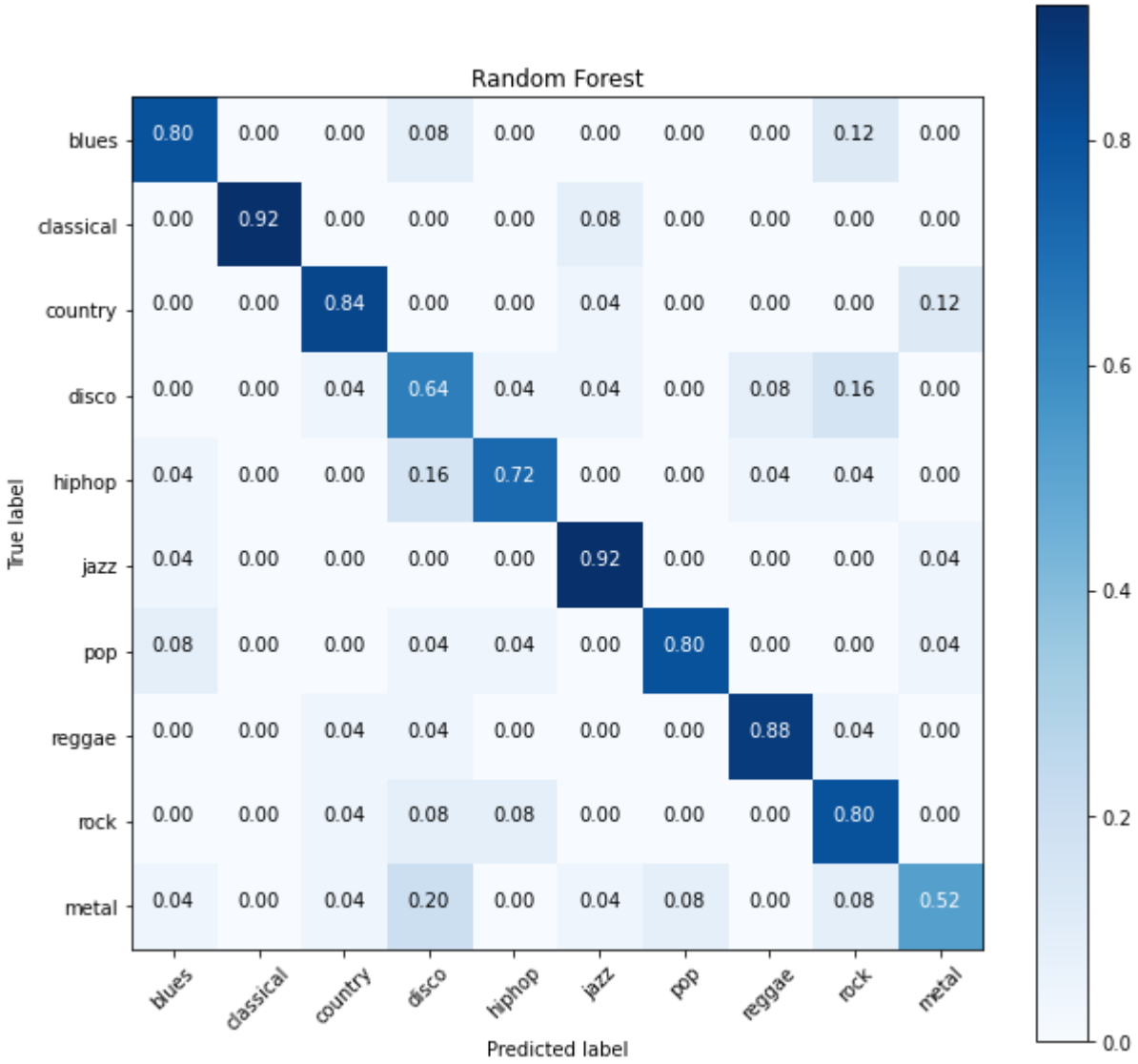

Figure 42. The confusion matrix for the Random Forest algorithm on the full suite of features

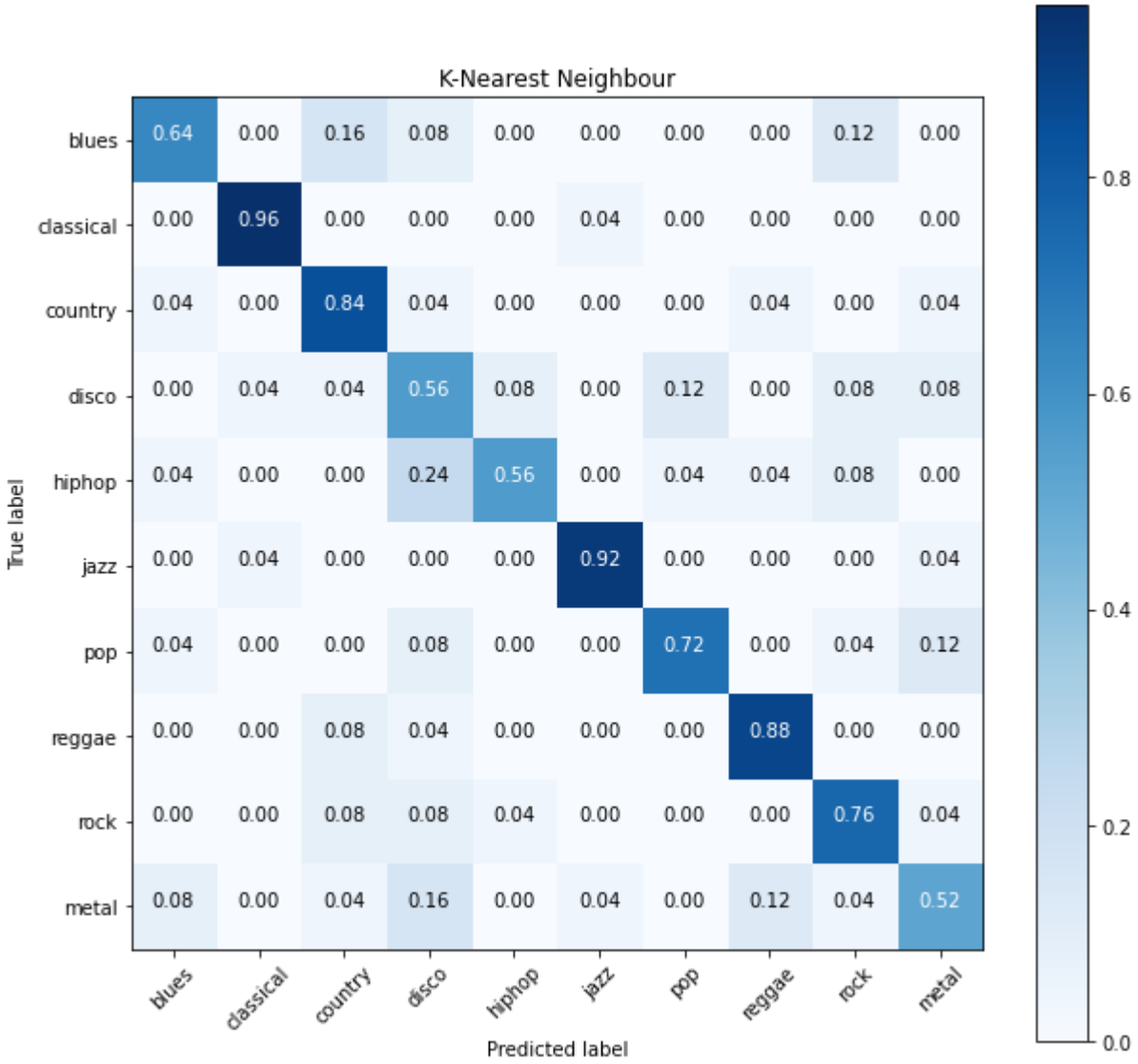

Figure 43. The confusion matrix for the KNN algorithm on the full suite of features

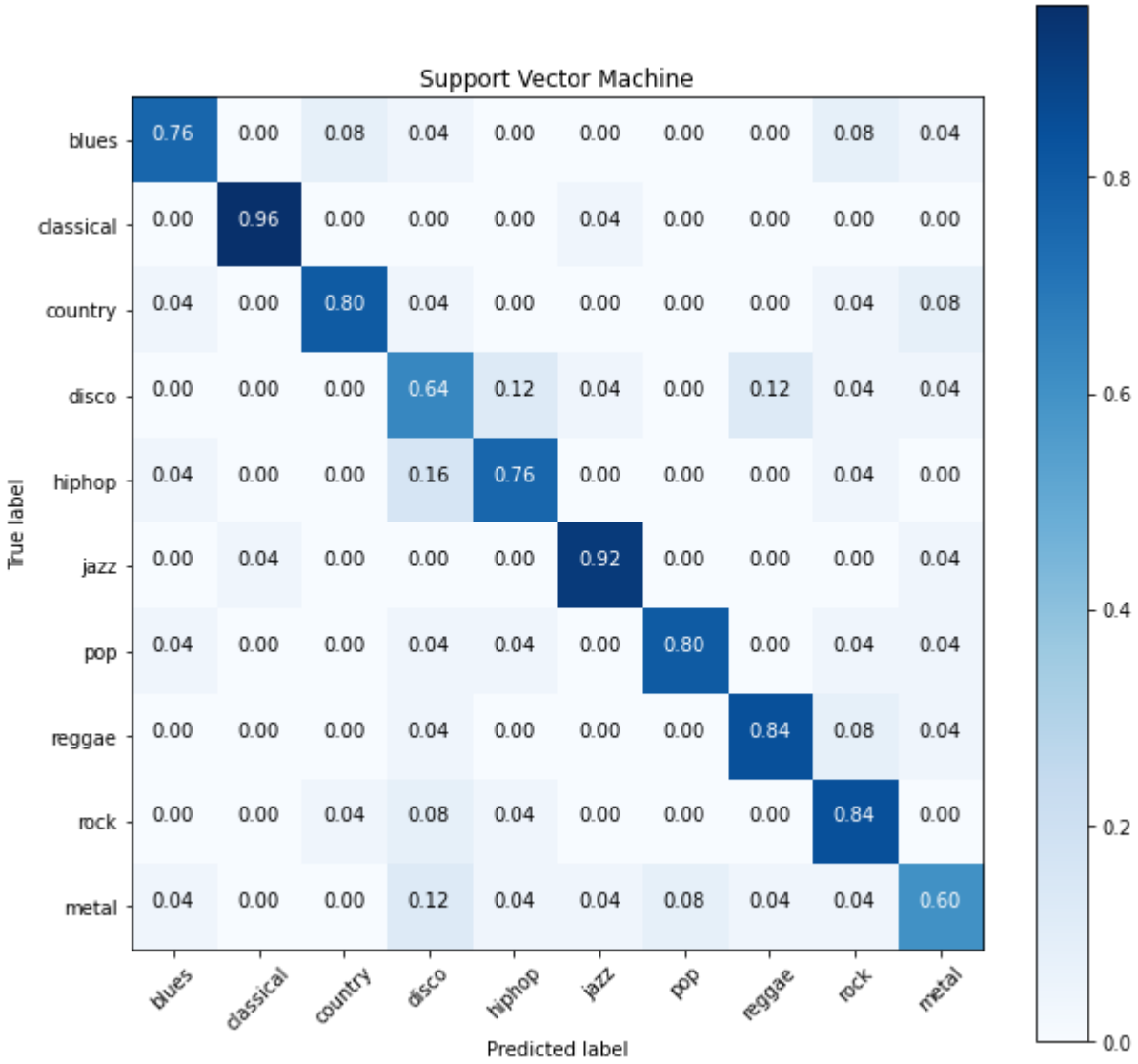

Figure 44. The confusion matrix for the SVM algorithm on the full suite of features

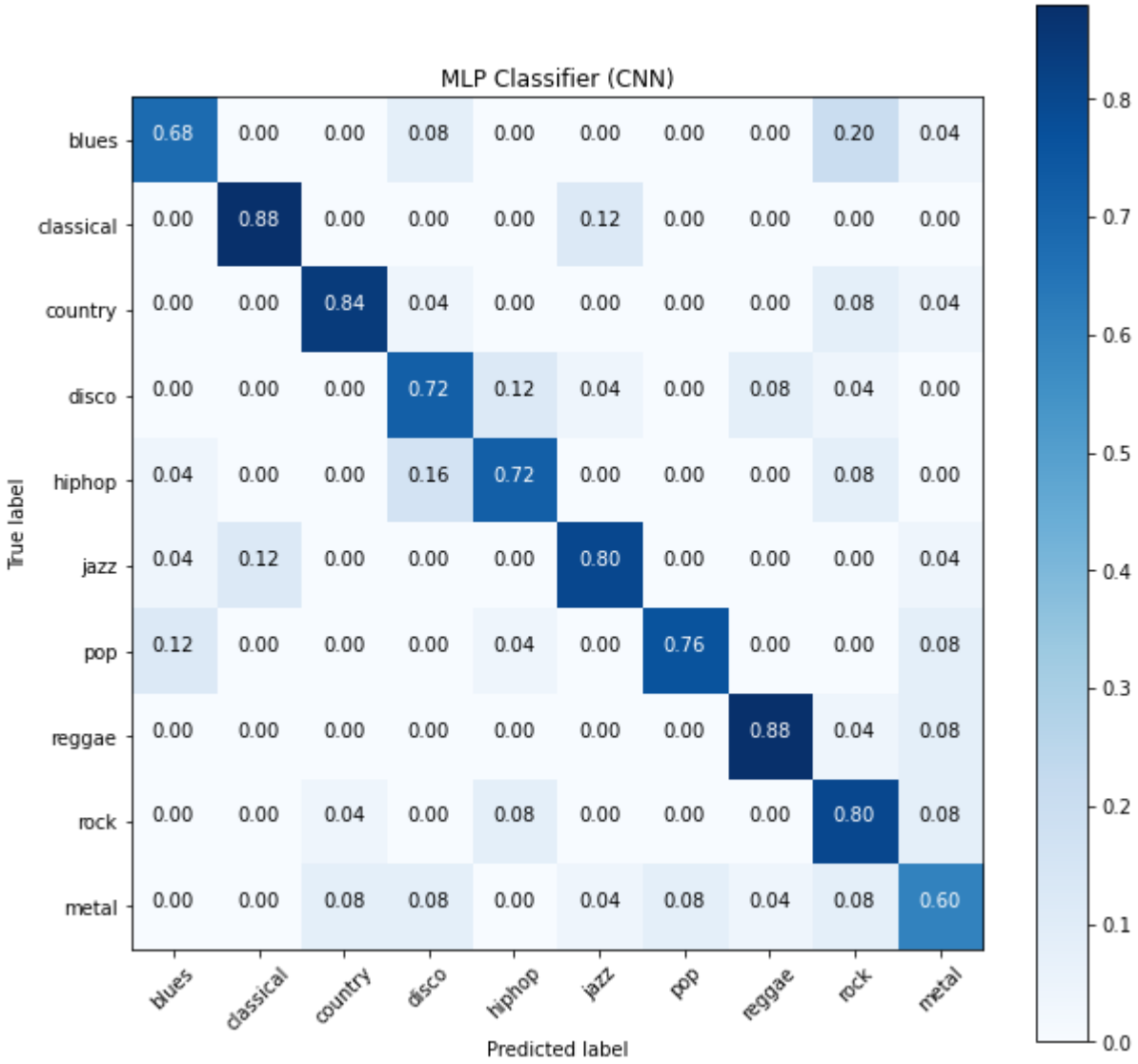

Figure 45. The confusion matrix for the MLP Classifier algorithm on the full suite of features

Appendix 2: Confusion matrices for individual features

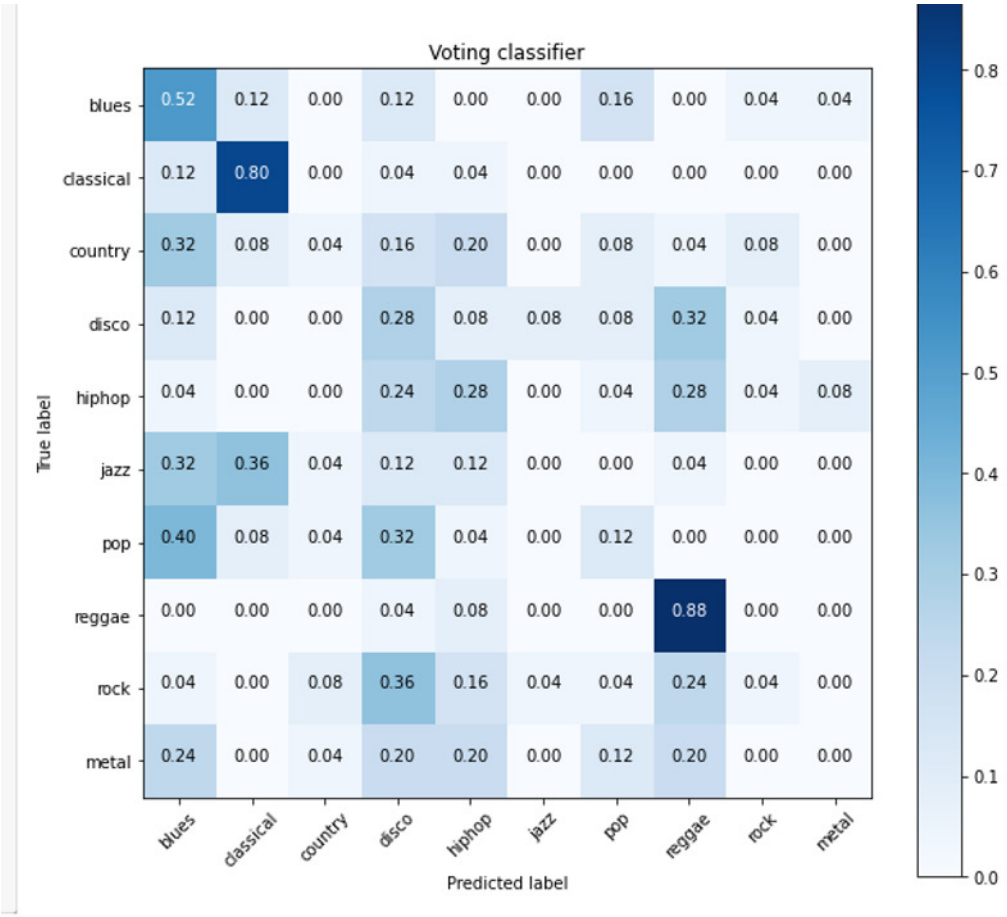

Figure 46. The confusion matrix for the Voting Classifier algorithm on solely the Spectral Bandwidth feature

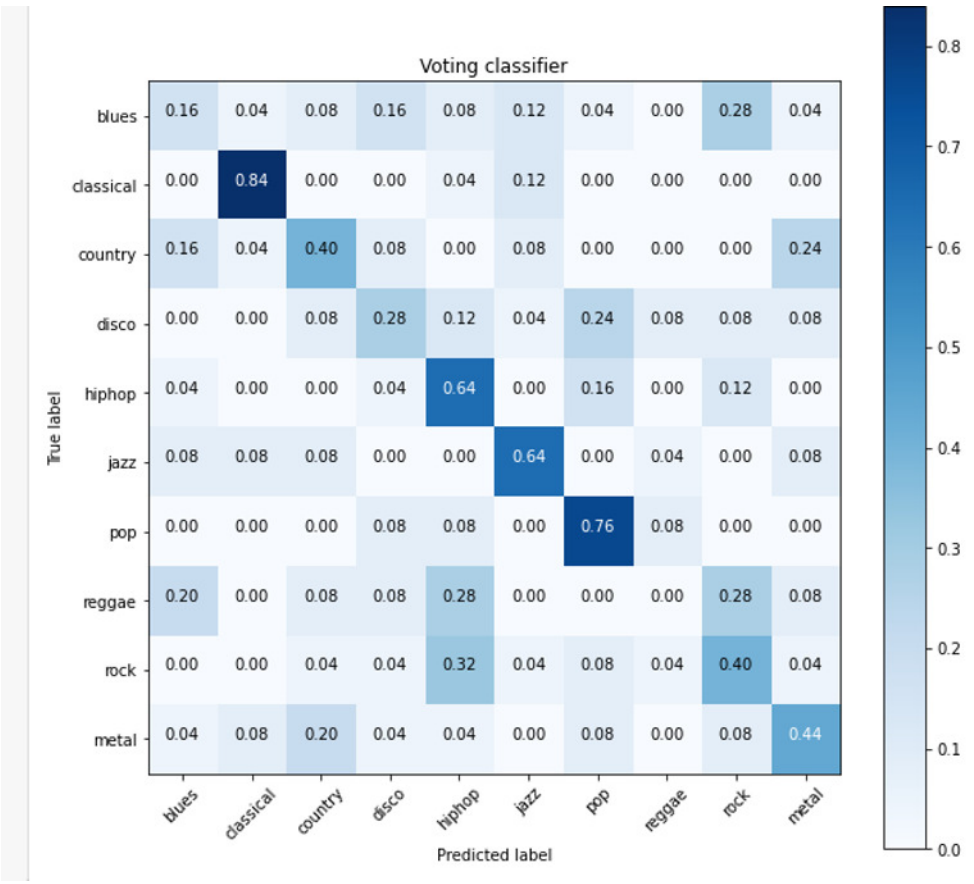

Figure 47. The confusion matrix for the Voting Classifier algorithm on solely the Chroma feature

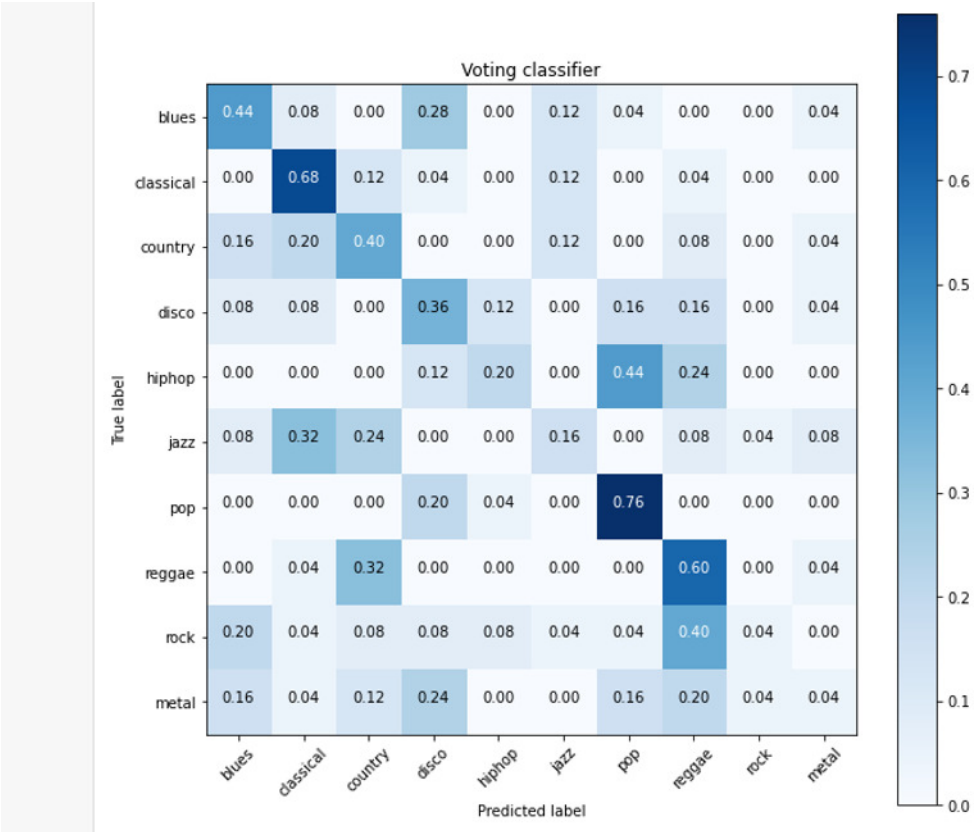

Figure 48. The confusion matrix for the Voting Classifier algorithm on solely the Spectral Contrast feature

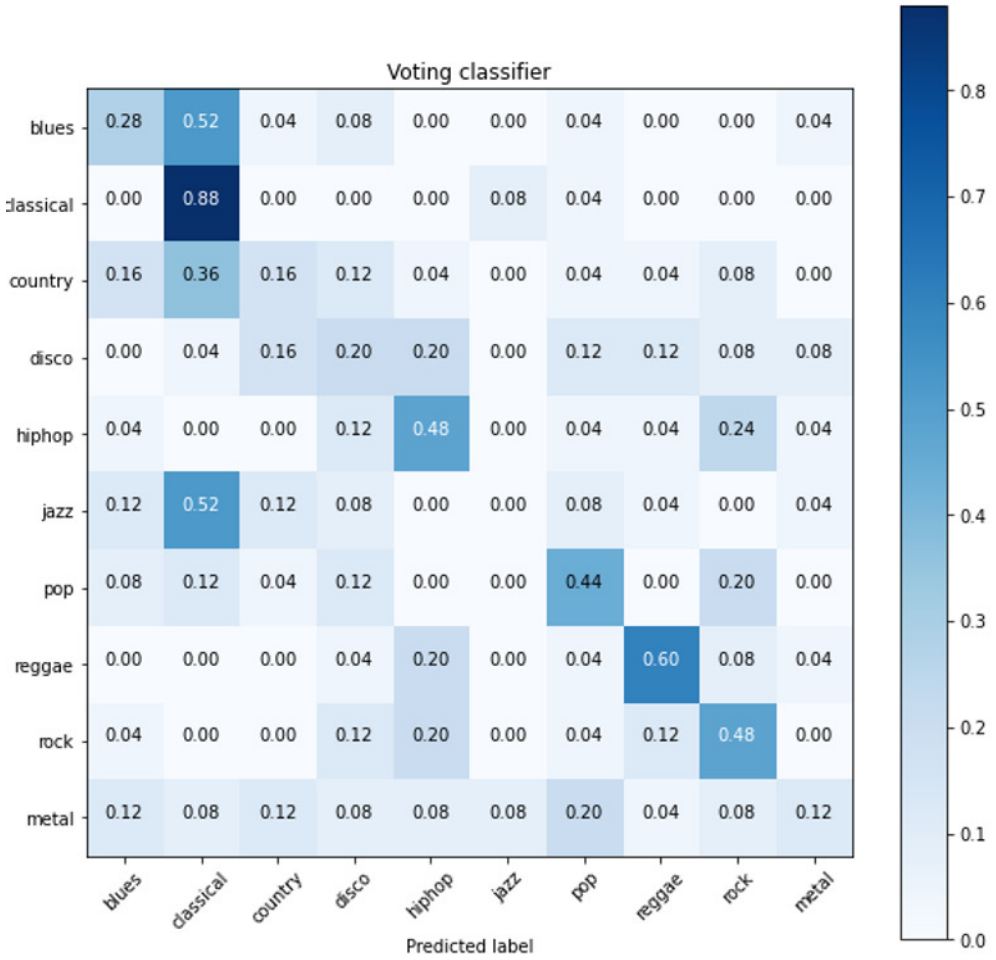

Figure 49. The confusion matrix for the Voting Classifier algorithm on solely the Spectral Flatness feature

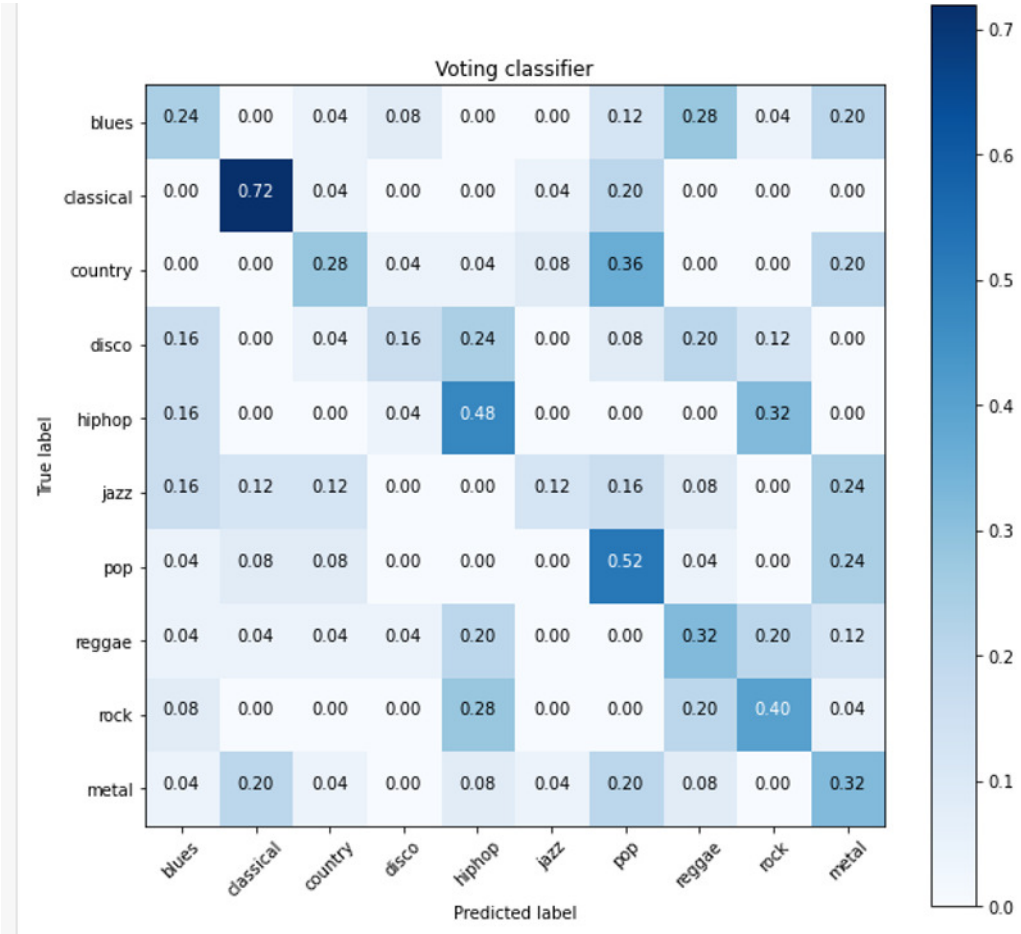

Figure 50. The confusion matrix for the Voting Classifier algorithm on solely the Spectral Flux feature

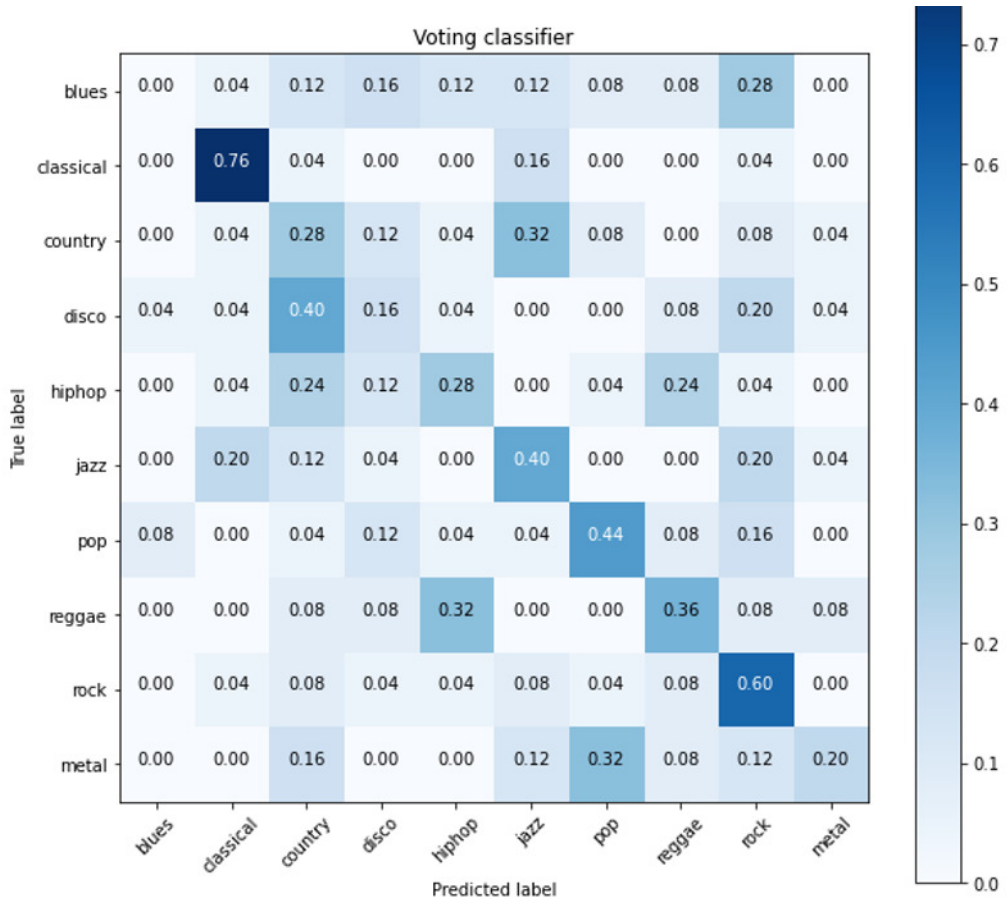

Figure 51. The confusion matrix for the Voting Classifier algorithm on solely the Root-Mean-Square-Energy feature

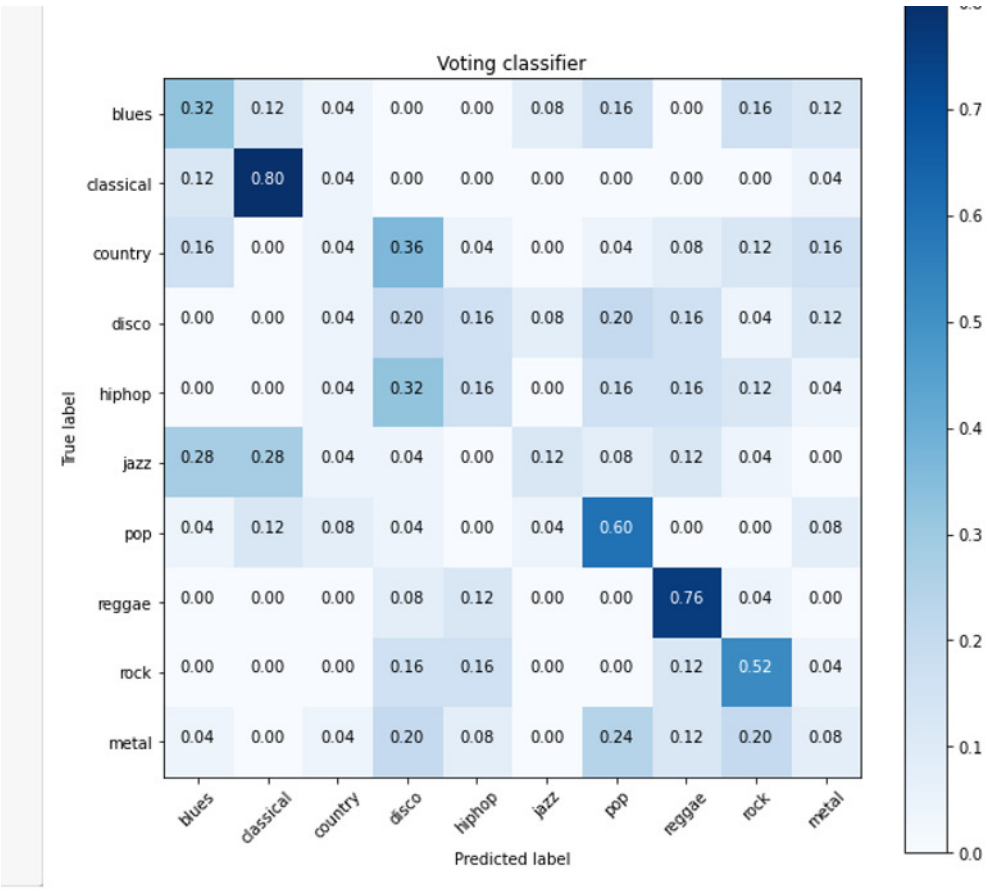

Figure 52. The confusion matrix for the Voting Classifier algorithm on solely the Spectral Rolloff feature

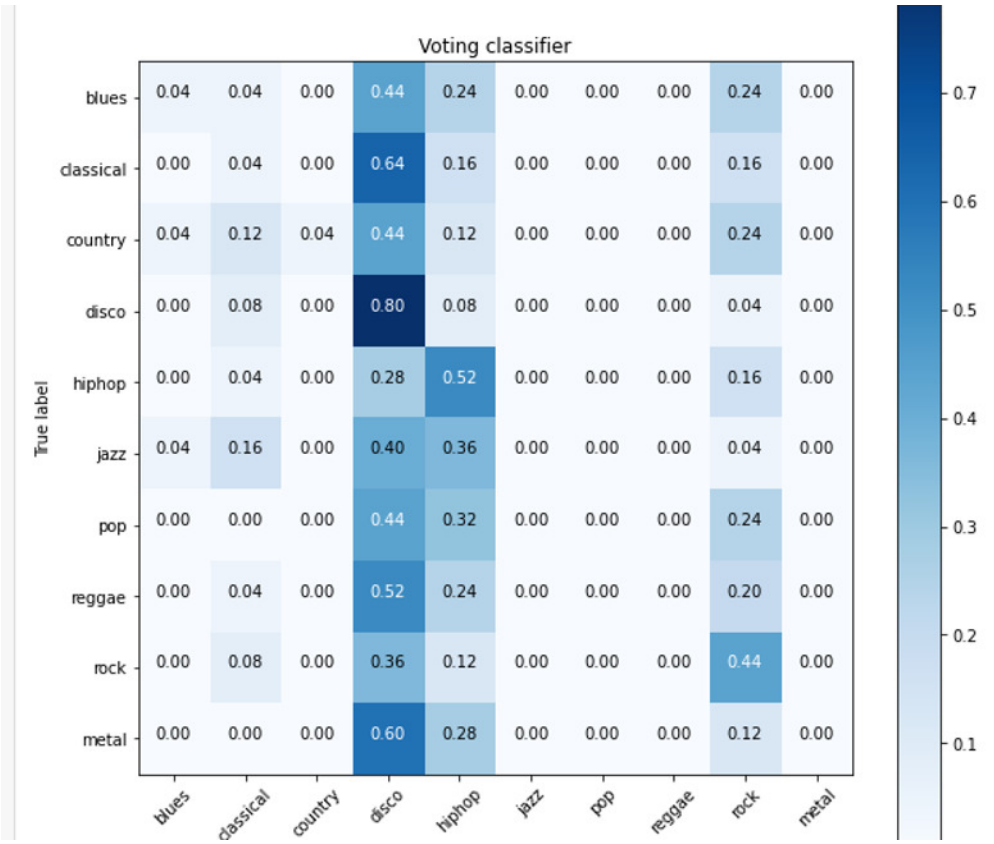

Figure 53. The confusion matrix for the Voting Classifier algorithm on solely the Tempo feature

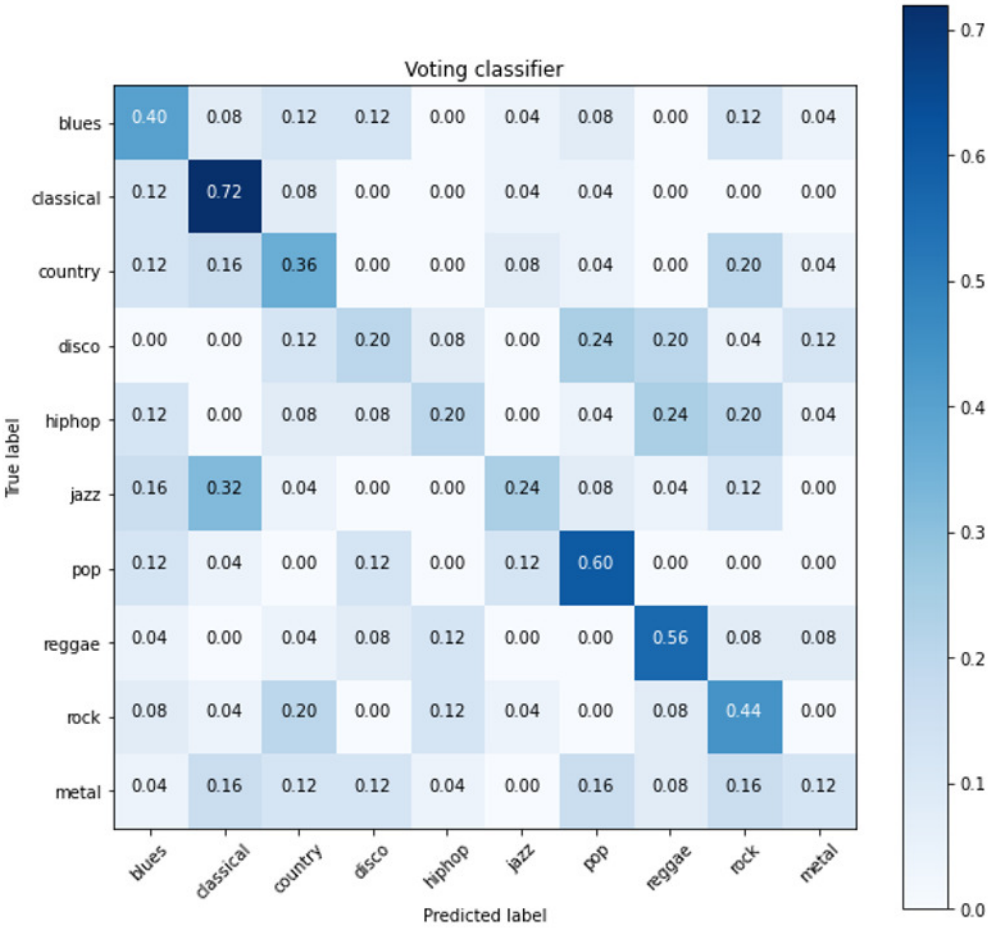

Figure 54. The confusion matrix for the Voting Classifier algorithm on solely the Zero Crossing Rate feature