Music Genre Classification

University examination paper for the subject "Audio Pattern Recognition"

Stavros Ntalampiras
Assistant Professor and Associate Researcher
Department of Computer Science of the University of Milan
Milan, Italy
stavros.ntalampiras@unimi.it

Francesco Ruggeri
Computer Science Master's degree student
University of Milan
Milan, Italy
francesco.ruggeri2@studenti.unimi.it

***Abstract*—Audio signal processing is a widely studied topic used for the creation of automatic musical genre recognition. In this paper we present a complete pipeline which comprehend audio processing, feature extraction, and the comparison of 2 classification methodologies.**

***Index Terms*—music, audio, genre, classification**

I. Introduction

A. Dataset

This study was performed using the dataset "FMA: A Dataset For Music Analysis", freely distributed by the "International Society for Music Information Retrieval Conference". Among the available version, this study is based on the small version of the datased which includes 8000 tracks of 30 second each, spread in 8 balanced genres (electronic, experimental, folk, hip-hop, instrumental, international, pop, rock). All the described procedure have been performed using Python 3.8 as programming language, wrapped around a Jupyter notebook for code development. The only exception resides in the audio format conversion explained in the next sections.

II. Pre-processing

A. Filesystem organization

The dataset provides a metadata file, which has been used to retain the wav file genre. This information is used to organize audio files into sub folder, where every folder corresponds to one genre, as shown in the following tree:

$|| \ dataset/$
$||\rule{2em}{0.5pt} \ electronic/*.wav$
$||\rule{2em}{0.5pt} \ experimental/*.wav$
$||\rule{2em}{0.5pt} \ folk/*.wav$
$||\rule{2em}{0.5pt} \ hip-hop/*.wav$
$||\rule{2em}{0.5pt} \ instrumental/*.wav$
$||\rule{2em}{0.5pt} \ international/*.wav$
$||\rule{2em}{0.5pt} \ pop/*.wav$
$||\rule{2em}{0.5pt} \ rock/*.wav$

The classification target label has been taken directly from the folder name of the audio files, allowing to bypass the whole metadata content.

B. Audio format and re-sampling

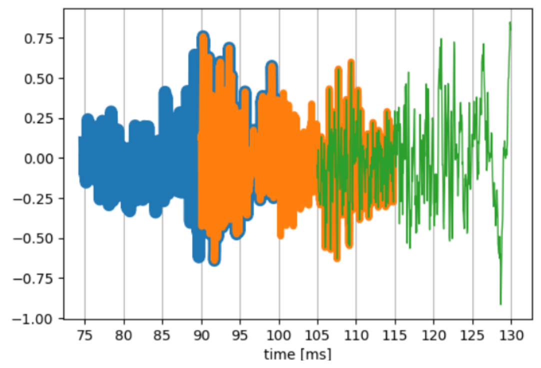
Each audio format file has been converted from MP3 to WAV, in order to speed up the importing process preceding the feature extraction. The conversion has been done using the open source tool FFMPEG, with the following command:

```
ffmpeg -i audio-file.wav audio-file.mp3
```

Files are imported using the python package for music and audio analysis called Librosa. The importing procedure uses as parameters a sampling frequency of 22KHz and the signal conversion from stereo to mono channel.

C. Windowing

The preprocessing proceeds with the generation of rolling windows, created from the original signal of every audio file. Each window has a length of 25 milliseconds, corresponding to 550 samples. Among windows we kept a 10 millisecond overlap. The figure below shows an example of this process.

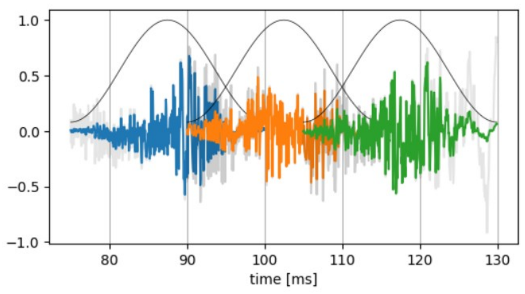


D. Windowing function

The Hamming function has been applied on windows in order to smooth the discontinuities at the beginning and end of the sampled signals. The parameters used to generate the function have been defined as alpha=0.54 and beta=0.56.

w(i) = 0.54 - 0.46 $\cos\frac{2\pi i}{n-1}$

The generated window has been multiplied over existing windows, producing the following smoothed signals:



E. FFT

The fast Fourier transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence. This analysis converts a signal from its time domain to a representation in the frequency domain. The FFT has been applied to all overlapping windows after applying the Hamming windowing function. The result is not directly used during the classification, but it is a required step for the extraction of some features explained in the following section.

III. Time-domain audio feature extraction

In the following section we can find a brief explanation of the time-domain features that have been extracted from the windows. Time domain are referred to variations of amplitude of signal along with time.

A. Zero-crossing rate

The zero-crossing rate is the rate of sign changes happening along a signal. This rate represents the amount of signal crossing from less than zero to great and viceversa.

The ZCR feature is mathematically defined as follows:

$$Z(i) = \frac{1}{2W_L} \sum_{n=1}^{W_L} |\ sgn[x_i(n)] - sgn[x_i(n-1)]$$

where s is a signal of length T and R is the indicator function.

B. Signal Energy

The energy of a signal is defined as a sum of square of magnitude. This measure represents the area under the squared magnitude of the considered signal. Mathematically, it has been evaluated as follows:

$$E(i) = \frac{1}{W_L} \sum_{n=1}^{W_L} |x_i(n)|^2$$

C. Energy Entropy

The entropy of energy is a measure which quantifies the abrupt changes of the energy level of a signal. To evaluate this feature, we firstly divide the signal window into sub-window of fixed duration. In our case, every window has been divided into 10 sub-windows. We evaluate the energy of every sub window as seen in subsection *B* and we divide it by the energy of the whole window.

$$e_j = \frac{E_{subFrame_j}}{E_{shortFrame_i}}$$

where

$$E_{shortFrame_i} = \sum_{k=1}^{K} E_{subFrame_k}$$

We finally extract the entropy evaluating the following equation:

$$H(i) = -\sum_{j=1}^{K} e_j \cdot \log_2(e_j)$$

IV. Frequency-domain audio feature extraction

In the following section we can find a brief explanation of the frequency-domain features that have been extracted from the windows. Frequency-domain features are based on the DFT of the audio signal.

A. Spectral Centroid

The spectral centroid is a measure used to characterise a spectrum. This signal processing indicates the location of the center of mass of the spectrum. It has been evaluated with the following equation:

$$C_i = \frac{\sum_{k=1}^{Wf_L} k X_i(k)}{\sum_{k=1}^{Wf_L} X_i(k)}$$

B. Spectral Spread

The spectral spread is the second central momentum of the spectrum. It is a measure of the average spread of the spectrum in relation to its centroid.

$$S_i = \sqrt{\frac{\sum_{k=1}^{Wf_L} (k - C_i)^2 X_i(k)}{\sum_{k=1}^{Wf_L} X_i(k)}}$$

C. Spectral Entropy

The spectral entropy is a measure of its spectral power distribution. It is evaluated similarly to the energy of entropy seen in section *III* subsection *C*. This feature is evaluated on the signal DFT, so we are considering the frequency domain.

We firstly divide the window into short-term bins, in our case 10. We proceed by evaluating the energy of every sub-band which is later normalized by the total spectral energy. Finally, we evaluate the spectral entropy with the following equation:

$$H = -\sum_{f=0}^{L-1} n_f \cdot \log_2(n_f)$$

D. Spectral Rolloff

The spectral rolloff point is the fraction of bins in the power spectrum at which a specified percentage of the power is at lower frequencies. The spectral rolloff has been evaluated with the following formula:

$$\sum_{k=1}^{m} X_i(k) = C \sum_{k=1}^{Wf_L} X_i(k)$$

where C is the specified percentage (in our case 90%). The obtained value has been normalized (divided by the window lenght) in order to obtain values from 0 to 1.

E. Spectral Flux

Th spectral flux is a measure of how quickly the power spectrum of a signal is changing, calculated by comparing the power spectrum for one frame against the power spectrum from the previous frame. We then evaluate the following equation among the current and the previous windows. This feature is evaluated using the signals extracted DFT.

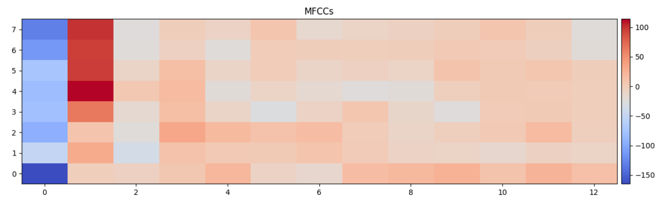
$$Fl_{(i,i-1)} = \sum_{k=1}^{Wf_L} (EN_i(k) - EN_{i-1}(k))^2$$

F. Mel Spectrogram

The Mel spectrogram is an approximate representation of the power spectrum of a sound perceived by the human auditory system. In this spectrogram the frequency bands are distributed according to the mel-scale instead of the linearly spaced approach. We used the pre-built library librosa for MFCCs extraction. In our case, we chose to extract 13 components. In general, this feature are commonly derived as follows:

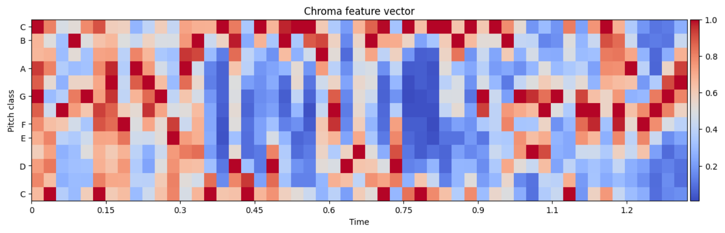
- take a window signal and extract the DFT
- generate triangular overlapping windows
- multiply the powers of the spectrum of the DFT and the triangular windows
- take the logs of the powers at each of the mel frequencies
- take the discrete cosine transform of the list of mel log powers
- the resulting MFCCs are the amplitudes of the resulting spectrum that we will use as a feature

In this figure we can see an example of 13 MFCCs extracted on 7 windows.



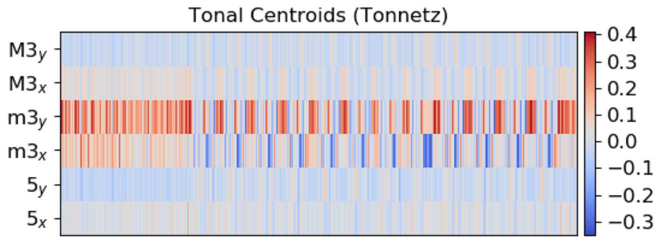
G. Chroma Spectrogram

The main idea of the chroma feature is to aggregate all spectral information that relates to a given frequency (in this case we can refer to pitch) into a single coefficient. This representation is useful if we consider the human auditory system, where the same pitches from different octaves are perceived with the same "color". The chroma feature is generated by grouping the DFT coefficients into bins which express the magnitude of the specified frequency range of the bin. Using the pre-built library librosa, we extracted 13 bins of pitches, to cover a full octave (from C to B) and including a C from an upper octave.



H. Tonnetz Spectrogram

The Tonnetz is a pitch space defined by the network of relationships between musical pitches in just intonation. This feature express the tonal centroid an audio signal. For each window we extracted 6 bins expressing the tonal centroid.



V. Common Statistics

In addiction, we extracted some common features, evaluated on each window. These features are the following ones:

- Max
- Mean
- Median
- Min
- Skew
- Std
- Kurtosis

VI. Post-processing

We extracted a total of 18 features, described above. The number of feature raises up to 47 if we consider all the bins coming from the Mel spectrogram, the Chroma vector and the Tonnetz spectrogram.

A. *Feature averaging*

In the context of music genre classification it is useful to reduce the amount of features per audio because their representation on longer windows can result acceptable.

In this study, the short-term features are generated on windows of 25 milliseconds each. Because of 10 milliseconds of overlap, each window has just 15 milliseconds of "unseen" signal. In order to obtain features at every 2 seconds, we averaged the resulting set in block of 133 rows.

$$window\_length = 25ms$$

$$overlap\_length = 10ms$$

$$n\_seconds = 2000$$

$$step = n\_seconds//(window\_length - overlap\_length)$$

B. *Median Filter*

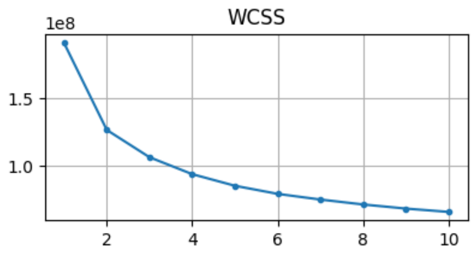
A median filter has been applied to the extracted features, in order to smooth the generated values and to reduce the chance of outliers. The kernel size used as a parameter for the filter have a value of 3.

VII. Classification

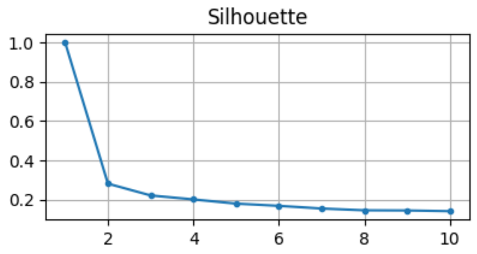
A. *K-Means*

K-Means is an unsupervised learning algorithm which aims to classify a given data set through a certain number of clusters (k), that must be fixed apriori. The k parameter is the most important to evaluate the algorithm and it has been chosen using the Elbow Method. The Elbow method is a heuristic used in determining the number of clusters in a data set. The method consists in plotting the variation of a metric evaluation as the number of clusters varies. After this, we can pick the elbow of the curve as the optimal number of clusters to use. We used two metrics for this evaluation:

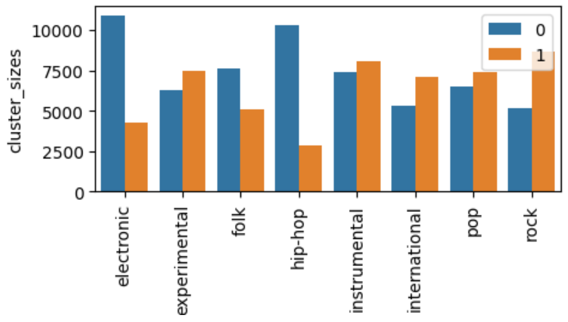
- WCSS (within cluster sum of squares): this value is the sum of squares of the distances of each data point in all clusters to their respective centroids.



- Silhouette: this value represents a measure of how similar an object is to its own cluster, compared to other clusters. The silhouette ranges from -1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.



The discriminative power of the K-Means resulted limited in this study and with this dataset. The elbow method shows us that 2 is the optimal parameter. We expected an higher value, given that we are clustering a dataset containing 8 different genres. When setting k as 2, we will have 2 resulting cluster. The amount of genres in each cluster is heterogeneously disposed and this is expected because we are considering 2 clusters containing 8 genres.



These results made us decide to not use the resulting cluster information inferred by the k-means, into the feature set for classification.

B. *Random Forest*

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. This is the first chosen model for this study. The parameters have been chosen with an hyperparameter tuning methodology explained later in this document.

C. *MultiLayer Perceptron*

A multilayer perceptron is a feedforward artificial neural network model that has one layer or more of hidden units and nonlinear activations. An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Also this model parameters have been studied and optimized with the hyperparameter tuning technique.

D. *Cross-validation*

The validation of the models have been performed on partitions of the feature set. We generated 10 stratified folds that will later be used by both RandomForest and the MLP-Classifier. The stratify strategy allowed us to maintain the genre proportion in each training/test set of the folds.

E. Hyperparameter tuning

In order to improve each classifier performances, the hyper-parameters have been tested on both classification algorithms before the final evaluation on the dataset. We used a pre-built tool called GridSearchCV, available in the Scikit-learn python library. This tool tests all combinations of given parameters and returns the best combination among them.

In both classification methods, the hyperparameters were chosen using a cross-validation method with just 2 folds, in order to reduce the computing time.

1) Random Forest: the parameters fed to the GridSearchCV are the following:

- criterion: [gini, entropy]
- n_estimators: [100, 250, 333]
- max_depth: [25, None]
- max_features: [15, auto]
- min_samples_split: [2, 10]
- class_weight: [balanced, balanced_subsample]

The best combination, later used for this study is:

- criterion: entropy
- max_depth: 25
- max_features: 15
- n_estimators: 333
- min_samples_split: 2
- class_weight: balanced

2) MultiLayer Perceptron: the parameters fed to the Grid-SearchCV are the following:

- solver: [adam, lbfgs, sgd, adam]
- hidden_layer_sizes: [50, (100, 50, 50)]
- max_iter: [100, 500]
- activation: [identity, logistic, tanh, relu]

The best combination, later used for this study is:

- solver: lbfgs
- hidden_layer_sizes: 50
- max_iter: 50
- activation: identity

F. Three sets of feature comparison

in order to test the discriminant power of features, we evaluated both models with three subsets of feature sets.

1) Temporal features:

- zero_crossing_rate
- energy
- energy_entropy

2) Spectral features:

- spectral_centroid
- spectral_spread
- spectral_entropy
- spectral_rolloff
- spectral_flux
- mel_bins
- chroma_bins
- tonnetz_bins

3) All features:

- {Temporal features}
- {Spectral features}
- mean
- median
- min
- skew
- std
- kurtosis

G. Majority label

The classification training and prediction made with the model lies on each feature window extracted from each audio signal. After evaluating the model on the test-set, we are able to find a label prediction for each window in the test-set. In order to obtain one label for each audio file, we applied a majority on all the labels obtained. For instance, for each audio file, we select the most common label predicted during the model evaluation.

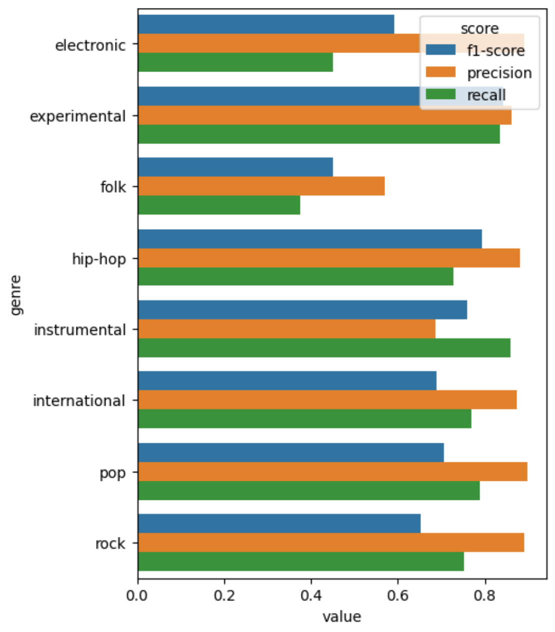
H. Random Forest

The overall model validation has been performed on these three feature subsets enriched by the cross-validation folds. The single accuracy value for a feature subset shown below is obtained by averaging all accuracies returned during the cross-validation:

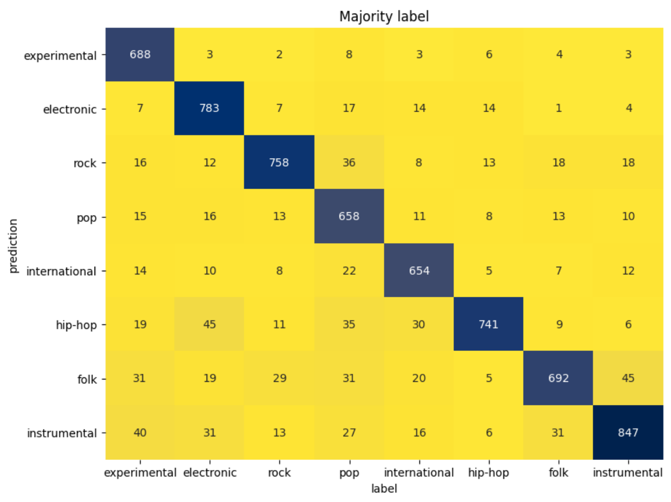
Feature sub-set	Accuracy
All	0.842
Spectral	0.821
Temporal	0.456

All 47 features returned the maximum accuracy, followed right after by the Spectral features which obtained a slightly lower score. The Temporal features alone are not able to generalize the concept of genre. Most of the work is done by the spectral features.

We decided to proceed the evaluation using the all-features set. In the following plot we can see the metric score, calculated for each genre.



From the last cross-validation fold we extracted the confusion matrix. Genres are enoughly well concentrated in the diagonal where the genre truth value meets the model prediction on the test set.

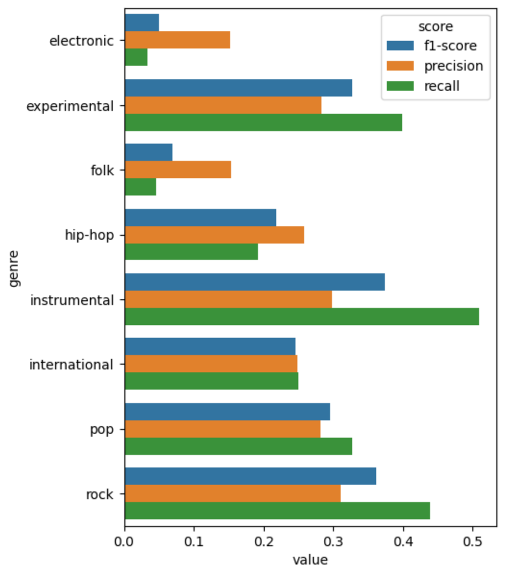


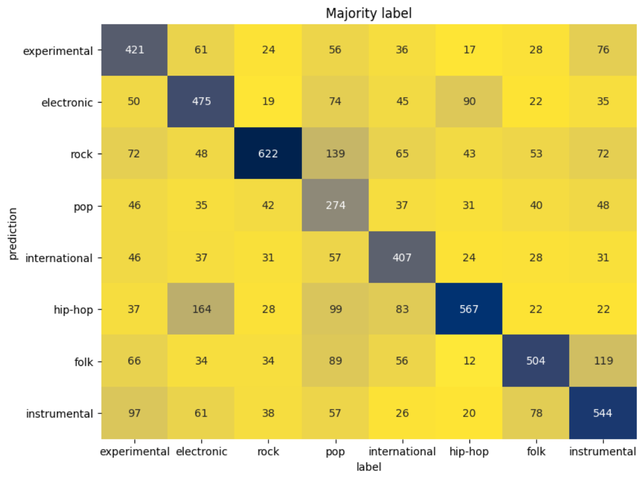
I. *MultiLayer Perceptron*

The same evaluations have been applied to the MultiLayer Perceptron classifier. We proceeded to evaluating all cross-validation folds with the three feature subsets.

Feature sub-set	Accuracy
All	0.660
Spectral	0.613
Temporal	0.456

The accuracies resulted lower compared to the ones obtained with the Random Forest classifier.





J. *Model combining*

The accuracy returned by the MLP Classifier is very small compared to the random forest. However, it is worth to study each classification genre prediction (good or bad). It is possible that the MLP classifier returns some good prediction that we are not able to get using the random forest.

The classification model provided by the SciKit Learn python package, furnish an useful tool which allows to return not just the predicted label, but each probability assignments of each entry to each label. This tool is called ”predict_proba”. This methodology has been applied on our models (Random-Forest and MLP). We ended up with two matrices, one for each model, that we’ll refer to as RF_proba and MLP_proba. The matrices will be structures as follows:

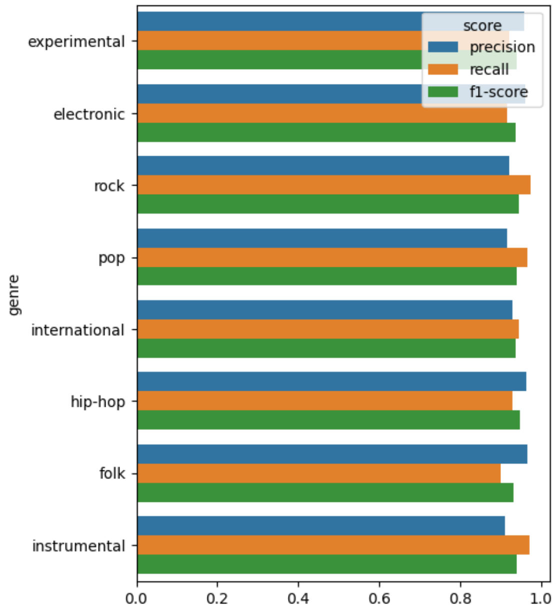
$$\begin{bmatrix} entry & genre1\_prob & \ldots & genre8\_prob \\ 1 & p & \ldots & p \\ 2 & p & \ldots & p \\ \ldots & \ldots & \ldots & \ldots \\ n & p & \ldots & p \end{bmatrix}$$

The model combination happens when we sum the two matrices of probabilities, extracted from both model. We end up with a matrix containing, for each entry, the sum of the probabilities of each label.

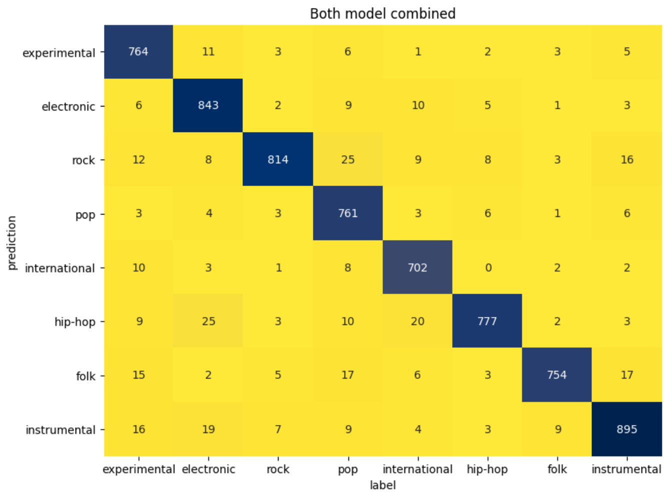
$$\begin{bmatrix} entry & genre1\_prob & ... & genre8\_prob \\ 1 & (p\_rf + p\_mlp) & ... & (p\_rf + p\_mlp) \\ 2 & (p\_rf + p\_mlp) & ... & (p\_rf + p\_mlp) \\ ... & ... & ... & ... \\ n & (p\_rf + p\_mlp) & ... & (p\_rf + p\_mlp) \end{bmatrix}$$

The final label for each dataset entry is lately evaluated as the maximum sum along the rows of this resulting matrix. After evaluating the majority label on each audio file, we obtained an overall accuracy of 94%. This result increased our random forest accuracy by 10%.

Precision, recall and f1-score are described as following:



And finally we can visualize the confusion matrix obtained:



	experimental	electronic	rock	pop	international	hip-hop	folk	instrumental
experimental	764	11	3	6	1	2	3	5
electronic	6	843	2	9	10	5	1	3
rock	12	8	814	25	9	8	3	16
pop	3	4	3	761	3	6	1	6
international	10	3	1	8	702	0	2	2
hip-hop	9	25	3	10	20	777	2	3
folk	15	2	5	17	6	3	754	17
instrumental	16	19	7	9	4	3	9	895