Matheus Henrique de Souza

Electronic Music Genre Classification

Belo Horizonte, Minas Gerais
2020

Matheus Henrique de Souza

Electronic Music Genre Classification

Projeto Orientado em Computação

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Supervisor: Adriano César Machado Pereira

Belo Horizonte, Minas Gerais
2020

Contents

1 Introduction

Automatic Music Genre Classification is a task of great importance due to both its practical and theoretical applications. It is fundamental for music indexing and retrieval, very useful, for instance, to the market of digital music. Besides that, since it heavily depends on Digital Signal Processing (DSP) oriented modeling, it should offer insights about DSP itself. Moreover, it is also concerned to the subjective aspects of music experience that sometimes are so subtle that, understanding how to better approach and emulate them, may lead us to an improved comprehension of human cognition. What makes this task challenging is, of course, processing the huge amounts of data that is generated by a single song file — making it very important to be able to get good results from small amounts of training data –, but also extracting and interpreting these subjective aspects of music from signals.

Electronic music emerged in the beginning of the 20th century, as the experimentation with electronic circuits led to the first electronic musical instruments. Analog synthesizers, electronic keyboards, the theremin, sequencers and, more recently, music production softwares and digital synthesizers would change the history of music production of all genres forever. According to an article by Billboard, in 2015 the electronic dance music market was worth 6.9 billion USD. Another article from 2018, also by Billboard, reports it grew to $7.1 bi. It is a huge market without any doubts, and it represents, for many people, a complete lifestyle.

Electronic Music is very diverse and covers many genres, each one of them with very distinct characteristics. It is easy to distinguish between some of them, and, for some others, its is not so easy. When it comes to subgenres of a main genre this becomes much more difficult since the differences become very subtle. For two different people the same track may be classified as belonging to completely different subgenres.

Given the relevance of the subject, the hereby proposed project takes on the duty to implement and evaluate a model to classify electronic music according to its genre using Machine Learning and Digital Signal Processing techniques and concepts. The main objective here is to build a music genre classification model capable of achieving satisfactory results using a small amount of training data.

The results yielded by the implemented model should also provide insights about the quality of the chosen features and shed some light on which of the subjective aspects of music better describe it. At the end, it is expected to have a robust classification model capable of being used as a practical tool for automatic classifying new songs into a collection, and it is also expected from this model to be a starting point for further research on the matter.

The remaining of this report is structured as follows: Section 2 briefly describes some

related works concerning the subject, Section 4 presents the methodology used to implement and evaluate the model and Section 5 explains the experiments and shows the achieved results. Finally, in the Section 6, the results are analysed and future works are discussed.

2 Related works

In 2006 McKay and Fujinaga produced an article compiling some counterarguments against a trend at that time in which researchers were suggesting that research in automatic genre classification should be abandoned in favour of more general similarity research, all due to the fact that works about automatic genre classification had been producing poor performance gains over the years before. In 2011, however a survey on audio-based music classification, conducted by Fu et al. cited Music Information Retrieval and Music Classification as "an emerging research area that receives growing attention from both the research community and the music industry". Music services like *Spotify* and *Deezer* were emerging at that time, and also, research in Machine Learning and AI was more mature, what explains this turn of table.

A paper from 2017, by Lee et al. achieves great results on music auto-tagging — which involves classification — using an end-to-end approach that learns hierarchical representations from raw data using deep convolutional neural networks. On the same year Lee et al. proposed a very similar model for music auto-tagging using deep convolutinal neural networks using only raw soundwaves and spectral features.

More recently, in 2019, Silva, Nunes, and Neto proposed a model for automatic music classification based on features extraction from MP3 audio files, using some of the features described in the section below and the classic Machine Learning algorithms K-Means (Kanungo et al. 2002) and SVM (Srivastava and Bhambhu 2010). A very interesting research by Farrokhmanesh and Hamzeh 2018 proposes a model that utilizes music classification as an approach for malware detection. Another related research, by Bhattacharjee, Prasanna, and Guha, from 2016 — republished this year —, achieves good results on speech and music classification using spectral features.

Many other projects about music classification can also be found on websites like *Kaggle*, resulting from challenges, personal projects and educative articles on DSP and Machine Learning. It is a widely explored subject and undoubtedly a very important one given the relevance of not only music, but audio in general in the everyday life.

3 Theoretical background

This section briefly explains some base concepts that are used along the conduction of this project.

3.1 Digital Signal Processing and Sound Analysis concepts

Digital Signal Processing (DSP) is concerned, as the name suggests, with treating, understanding and transforming signals. These signals can be a wide variety of things: music, noise, images, earthquakes, electricity among many others. This project will be dealing, of course, with the first two examples. Along this project song files are read, transformed, visualized and interpreted by models.

A song file is typically read as a **waveform**. A waveform represents the wave form of a sound, i.e., the displacement of an air molecule moving in space in the presence of the sound wave. The higher the amplitude of the wave, the higher the sound volume. The period of the wave indicates its pitch. The higher the period, the higher the pitch. Figure 1 below shows two waveforms: the first one represents the first 500 samples taken from a sine wave oscillating at 440 Hz — what produces the the A note —, and the second one shows the waveform for the 2 first seconds of a song, that consists of a kickdrum kicking at 134 beats per minute.

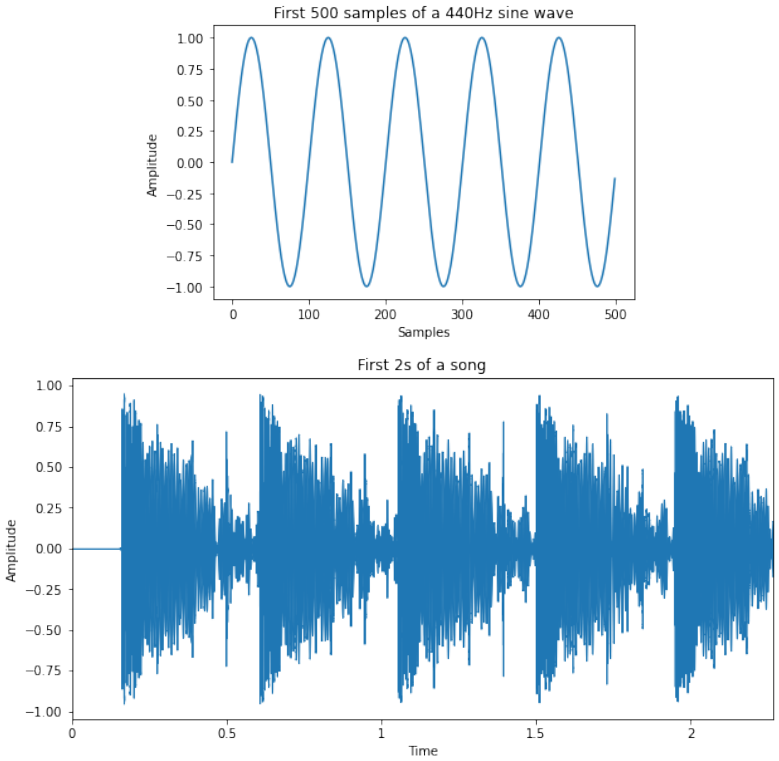


Figure 1 – Waveforms with linear amplitude.

Using Fourier Transforms — more specifically, Short-time Fourier transform (STFT) —, that decomposes a periodic function into its periodic constituents, it is possible to create **spectrograms** to analyse each of the sound's frequencies. Spectrograms are a representation of the spectrum of frequencies of a signal as it varies with time. Figure 2 below shows the spectrograms for the same waveforms presented in Figure 1.

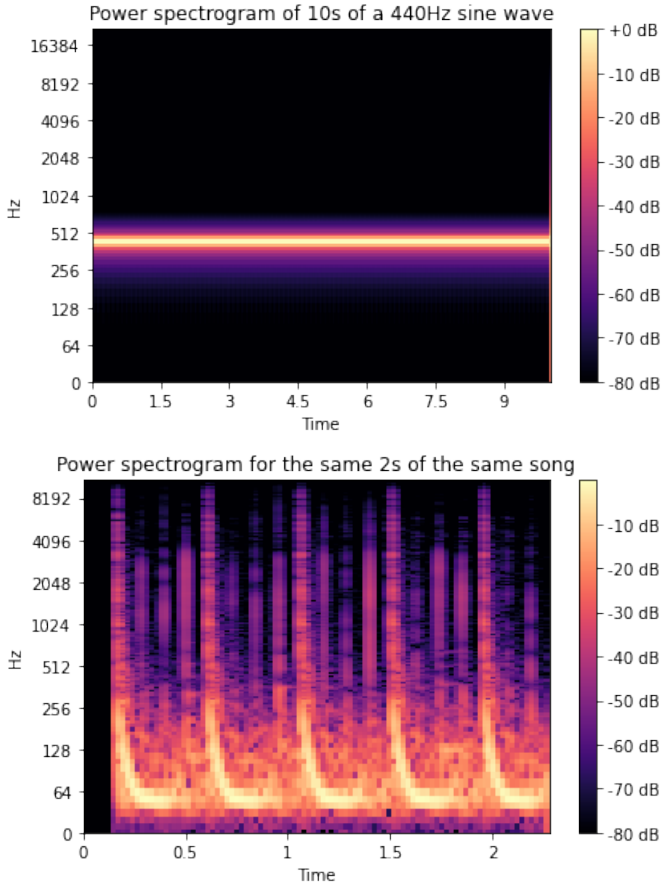


Figure 2 – Spectrograms for the same waveforms above.

The Mel Scale and the Melspectrogram

It was shown by some studies (Stevens, Volkmann, and Newman 1937) (Fletcher and Munson 1937) (Fletcher 1938) (Stevens and Volkmann 1940) that humans do not perceive sound frequencies on a linear scale. Our ears are more sensible to variations of audio in lower frequencies than in higher frequencies. Therefore, Stevens, Volkmann, and Newman proposed a scale in which the difference between a pair of unit pitches always sounds the same to the human ear. This was called the Melscale. By taking a spectrogram of frequencies fitting the Melscale, we create a Melspectrogram.

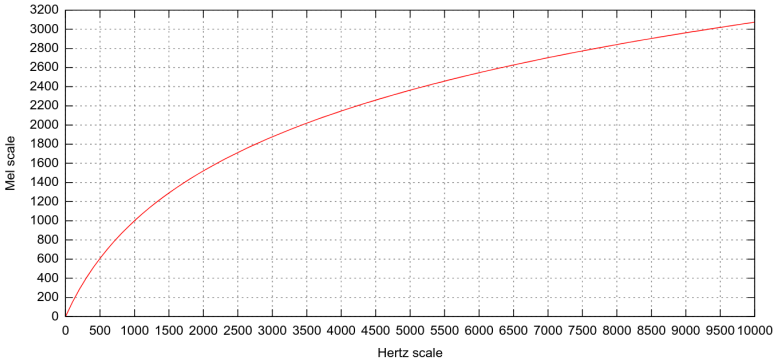


Figure 3 – Hertz scale vs Mel scale.

3.2 Audio Features Extraction

It is possible to extract many features that characterizes the sound from its waveform and spectrogram. This project works with the features provided as functions in the LibROSA Python Package. Each one of them are briefly explained below, but first, it is necessary to understand two simple but key concepts when dealing with digital audio files.

3.2.1 Sampling concepts

Sound is an analog phenomena. We perceive it through our ears as a "continous" sensation. However, in order to represent it digitally it is necessary to take *samples* from its signals. The samples are taken regurlarly at fixed time intervals and, the rate with which these samples are taken define the **sample rate**, measured in Hz. The higher the sample rate, the more precise is the representation of the original sound.

Many of the most common DSP operations work with *frames*. Frames are simply windows of n samples that are used for a single analysis on the sound's time series. Thus the number n is the **frame size**.

3.2.2 Audio features

Below we briefly describe each of some of the features that can be extracted from an audio piece and that were used to build the model in this project.

- **Tempo:** The tempo is one of the most fundamental concepts in music, since it describes how fast — or slow — a musical piece is paced. Some genres of music are distinct simply by its characteristic tempo.
- **Chromagrams:** chromagrams are diagrams that show the amount of each of the twelve tones from the Equal-Tempered Scale contained in an audio section. It can be calculated using the old-reliable STFT or more sophisticated methods like the Constant Q Transform (Brown and Puckette 1992) or the CENS (Müller and Ewert 2011).
- **MFCCs:** The mel-frequency cepstrum (MFC) (Xu et al. 2005) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear frequency — the Melscale explained above. Mel-frequency cepstral coefficients (MFCCs) are the coefficients that collectively make up an MFC.
- **RMS:** the root-mean-square (RMS) power value for each of the analysed frames in the spectrogram.

- **Spectral centroid:** The spectral centroid is a measure used to indicate where the center of mass — regarding the power of the frequencies — of the spectrum is located. Perceptually, it has a robust connection with the impression of brightness of a sound.
- **Spectral bandwidth:** the order-p spectral bandwidth, defined as:

$$\left(\sum_k S(k)\left(f(k) - f_c\right)^p\right)^{\frac{1}{p}} \tag{3.1}$$

  where $S(k)$ is the spectral magnitude at frequency bin k, $f(k)$ is the frequency at bin k , and f_c is the spectral centroid. When $p = 2$, this is like a weighted standard deviation.
- **Spectral contrast:** To compute the spectral contrast, 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.
- **Spectral flatness:** spectral flatness (or tonality coefficient) is a measure to quantify how much noise-like a sound is, as opposed to being tone-like. A high spectral flatness (closer to 1) indicates that the spectrum is similar to white noise.
- **Spectral rolloff:** the roll-off frequency is defined for each frame as the center frequency for a spectrogram bin such that at least r per cent of the energy of the spectrum in this frame is contained in this bin and the bins below.
- **Zero crossing rate:** the zero-crossing rate is the frequency with wich the wave crosses the x axis, i.e., changes from positive to negative and vice-versa.

3.3 Classification algorithms

There are many algorithms that can be used in classification taks. In this project we will be working with three of them, briefly explained below.

3.3.1 Naive Bayes

Naive Bayes (Hand and Yu 2007) is a supervised Machine Learning algorithm inspired by the Bayes' theorem. Basically, it works by counting events of interest among all events and applying the Bayes Theorem to these counts.

Given its simplicity for a decent performance, the Naive Bayes algorithm is often used as a baseline model for classification problems (Saad 2014).

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

Figure 4 – Bayes Theorem main equation

3.3.2 Support Vector Machine

The Support Vector Machine (Srivastava and Bhambhu 2010) model works by finding a hyperplane in an N-dimensional space (where N is the number of features) that distinctly classifies the data points. By minimizing a cost function, it aims to minimize the "margins" created by the hyperplane, as shown below:

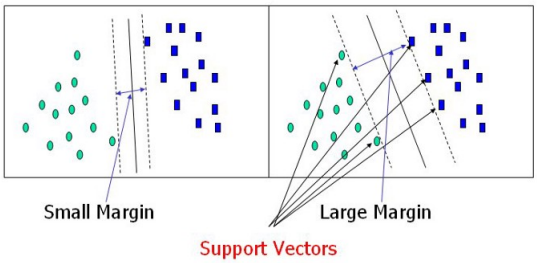
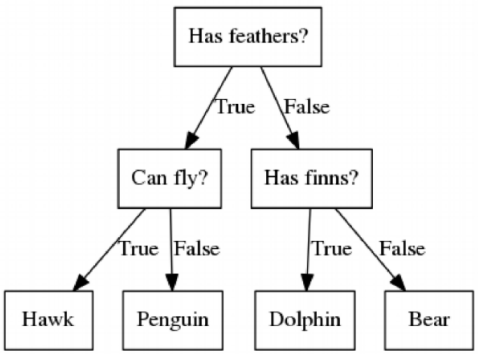


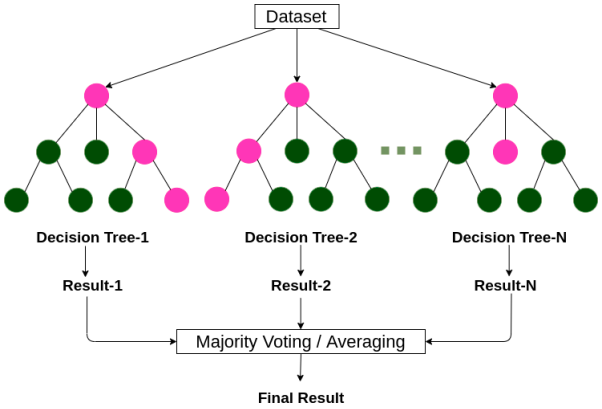
Figure 5 – SVM Margins

3.3.3 Random Forest

A Decision Tree is constructed by separating data through a series of decisions regarding an individual feature. It is illustrated as a tree flowchart in which each branching step means to split the data according to one feature. Random Forests (Tin Kam Ho 1995) are an ensemble of decision trees, i.e., it takes a large number of uncorrelated trees — each of them making its classifications based on a different order of features — and combine their predictions to try to outperform their individual results.



(a) An example of a decision tree being used to classify an animal



(b) Illustration of a Random Forest

Figure 6 – A decision tree and a random forest

3.4 Classification evaluation metrics

In order to evaluate the performance of a classifier model it is necessary to read the results according to some metrics of interest. For classification models we have four main metrics, explained below.

3.4.1 Precision and recall

Precision measures the proportion of positive identifications that were actually correct. Precision is defined as

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{3.2}$$

Recall measures what proportion of actual positives — from all positives — was predicted correctly. It is defined as

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3.3}$$

The closer these metrics get to 1, the closer the model gets to be perfect.

3.4.2 F1-Score

The F1-score is the harmonic mean of the precision and recall and is defined as

$$\text{F1-score} = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.4}$$

Since it is an harmonic mean, the F1-score tends to the smaller value between the precision and the recall, and can be taken as a balanced metric between the two.

3.4.2.1 ROC Curve

The ROC Curve is a curve that summarizes the true positive rate and the false positive rate of the classifier. The y axis measures the true positive rate, called **sensitivity** which is the same as the *recall*, explained above. The x axis measures the false positive rate, defined as

$$\text{False Positive Rate} = (1 - \text{Sensitivity}) = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}} \tag{3.5}$$

The higher the area — limited to 1 — under the ROC Curve, the better the model.

3.4.3 Macro and micro averaged scores

To evaluate a classifier, it is also important to define what macro and micro averaged scores are. For a multiclass classifier, a **macro-averaged** score is the plain average of the calculated score for all individual classes. The **micro-averaged** score means to take all individual true positives, false positives, and false negatives of the system — i.e., for each class — and to apply them to the equation of the score and calculate it.

4 Methodology

This section explains how the hereby described project was conducted. The following subsections explain each of the steps taken to build the implemented model. Figure 7 below graphically summarizes the adopted methodology.

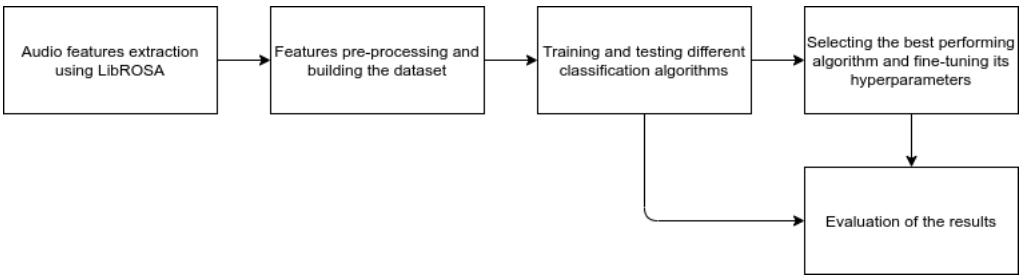


Figure 7 – Illustration of the proposed methodology

4.1 Selection of files

From a local collection of audio files consisting of approximately 7000 audio files manually classified into about 30 genres of electronic music, were selected 1480 of these files, from 6 of these genres — about 20% of the entire collection. The decision for such choice concerns two main reasons. First of all, as aforementioned, we aim for a model capable of achieving good results with a small amount of training input, due to the fact that, a practical application of the model would only be succesful if the tool using it is able to quickly learn how to classify new songs added to the user collection. The second reason is concerned with the huge amount of time taken to process individual song files and the number of features generated by them.

The considered genres and the number of files for each of them were as follows: *dubstep* - 295, *edm* - 221, *house* - 275, *techno* - 350, *trance* - 250, *trap* - 125.

4.2 Preprocessing the raw data and extracting audio features using LibROSA API

The LibROSA library is capable of extracting a rich set of features from audio files. After selecting some instances from pre-classified local files, some of the features supported by the library were extracted from these files to build a training dataset. However, after that it was necessary to preprocess these files. The first operation was to convert all files to the WAV format and then all of them were sampled to 30s of their duration, after their first 30s as illustrated below:

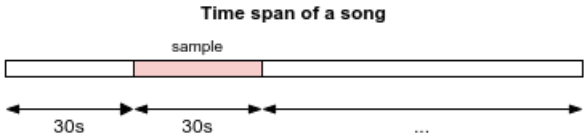


Figure 8 – Audio file sampling

The decision to sample the files like this was based on the knowledge that for most songs the first 30s are usually an introduction section and the core idea, which represents the best the genre of the song, comes after it. Also the 30s after the first 30s is usually a section that is neither the "peak" nor the "valley" of a song regarding its musical ideas, so it was determined to be the best portion that describes the entire song according to its genre.

4.3 Determining which features to use

For the current study we choose to use the features already described in Section 3.2, namely: *Tempo*, *Chroma vector*, *MFCCs*, *RMS*, *Spectral centroid*, *Spectral bandwidth*, *Spectral contrast*, *Spectral flatness*, *Spectral rolloff*, *Zero crossing rate*.

This set of features were chosen because they cover each of the broad subjective aspects of a song and, therefore, was considered to be a descriptive set for the task of music genre classification. It is also necessary to mention that for the *RMS*, *spectral centroids*, *spectral bandwidth*, *spectral contrast*, *spectral flatness*, *spectral rolloff* and *zero crossing rate*, since these features vary numerically over the time span of the audio piece, what was considered for them was their average, median and mode, so, in the numerical dataset, for each of these features we have three numerical values. Thus, we avoid dealing with an extremely large number of columns without losing their meaningfulness.

4.3.1 Number of columns in the dataset

The *MFCCs frame size* is of 2048 samples, the sample rate of the files is of $22050Hz$, i.e., 22050 samples per second. Since all files have $30s$, and we are using the first 13 MFCCs, this feature contributes with $\left\lceil \frac{30 \times 22050 \times 13}{2048} \right\rceil = 4199$ columns.

The *tempo* feature is a single value for the entire song, as well as the *spectral centroid*; the *chroma vector* is an array of 12 components; the *RMS*, *spectral bandwidth*, *spectral contrast*, *spectral flatness*, *spectral rolloff* features have, each, 3 components (mean, median and mode). Therefore, we have $1 + 1 + 12 + 6 \times 3 + 4199 = 4231$ columns in total.

4.3.2 Oversampling

Since the number of instances in each class are unbalanced, we oversampled the minority classes so all classes have the same number of instances of the majority class, i.e.,

all classes have 350 instances.

4.3.3 Stratified train-test split

In order to create a training and a testing dataset, the original dataset was randomly split in two: 30% of the data used for testing and 70% for training the models. The train-test split was made in way that preserves the proportion of instances in each class, i.e., for each class 70% of its samples are used for testing and 30% for training, resulting in balanced training and testing datasets.

4.4 Training and testing different models

With the training dataset ready to go, we could apply to them some classification models to try to find the one that best suits to the data. We chose to apply and evaluate three models — namely Naive Bayes, Support Vector Machine and Random Forest —, each of them working according different principles. They are briefly described in Section 3.3.

4.5 Commiting to the model with the best performance and fine-tuning its hyper-parameters

After evaluating the first results for each model, we select the one with the best performance to tune it's hyper parameters in order to try to achieve better results. Thus, the final model will use the best performing algorithm with the fine-tuned hyper parameters.

5 Experiments and results

Finally we proceed to the evaluation of the results given by each of the models explained in the last section. For each of them, were generated the confusion matrix, the values of macro-averaged precision, recall and the f1-score for the train-test split described above; the ROC curve and the values of macro-averaged precision, recall and the f1-score along a cross-validation of 5 different random stratified splits of the same data.

5.1 Results for Naive Bayes

Below we present the results obtained using the Naive Bayes algorithm.

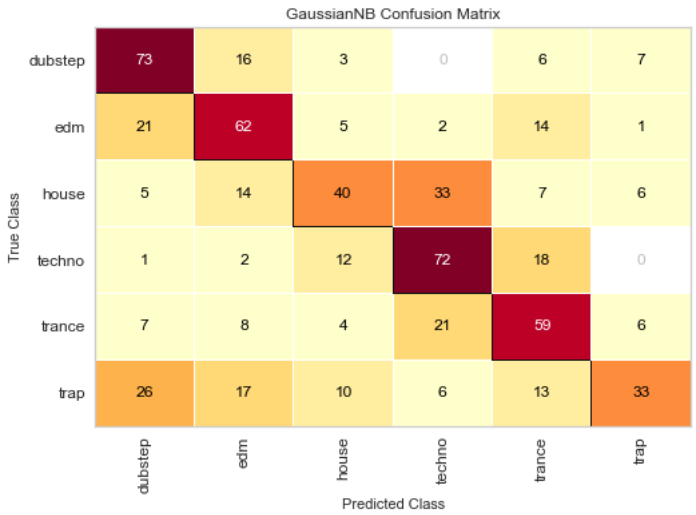


Figure 9 – Confusion Matrix for the Naive Bayes model

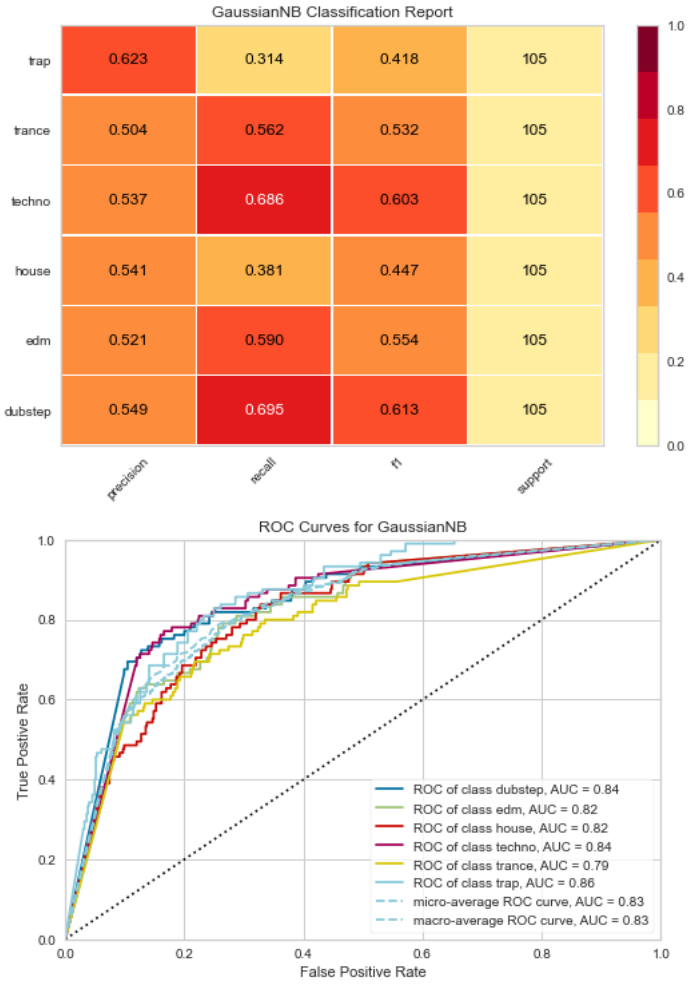


Figure 10 – Classification report and ROC curves for the Naive Bayes model

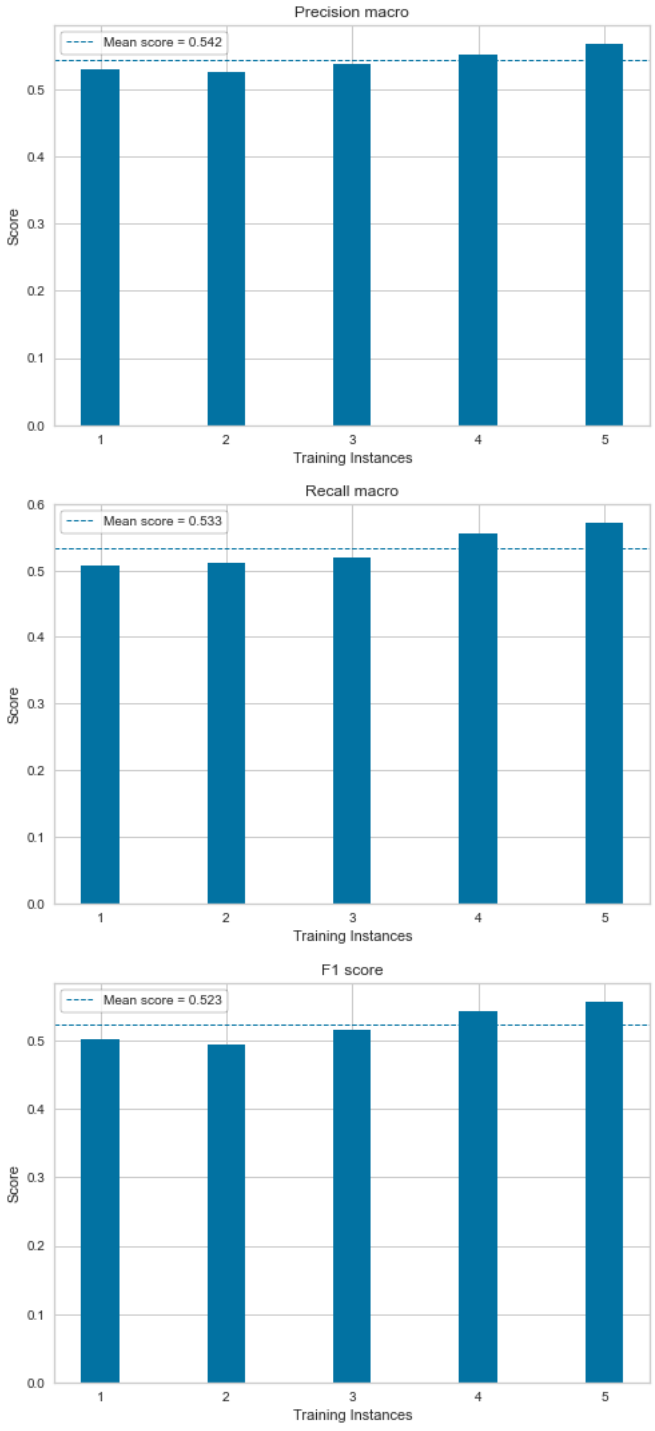


Figure 11 – Macro-averaged precision, recall and f1 score for the Naive Bayes model along 5 cross validation folds

The Naive Bayes algorithm obtained an average precision of 0.542, an average recall of 0.533 and an average f1-score of 0.523, the ROC AUC was of 0.83. As expected the Naive Bayes model didn't yeld impressive results.

5.2 Results for Support Vector Machine

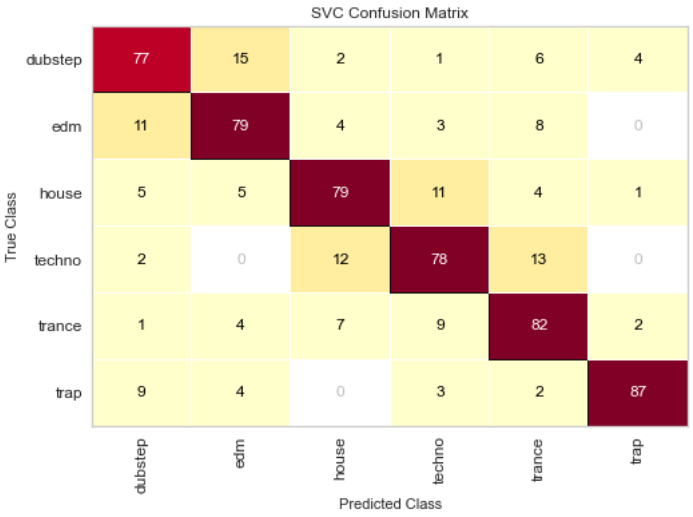


Figure 12 – Confusion Matrix for the SVM model

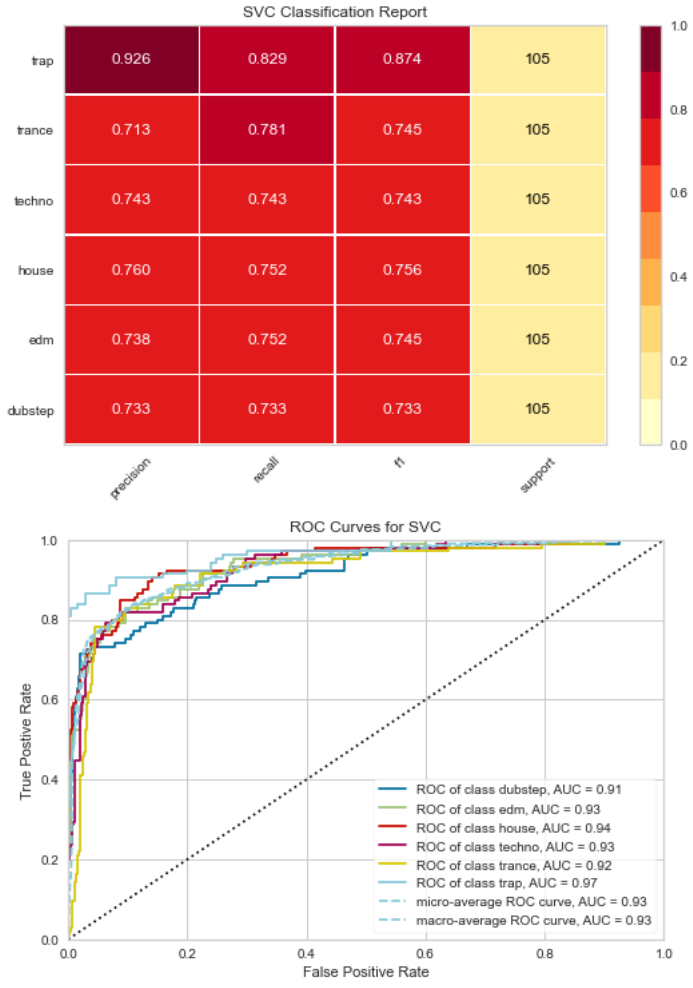


Figure 13 – Classification report and ROC curves for the SVM model

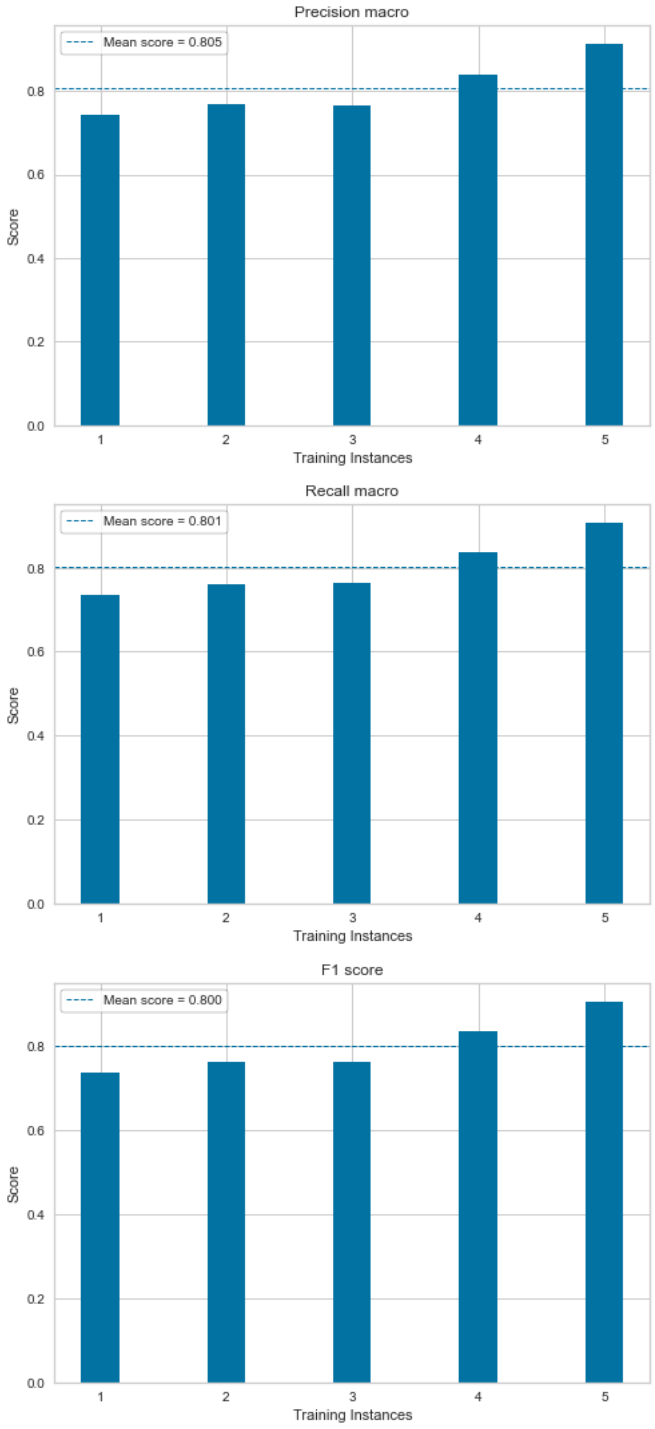


Figure 14 – Macro-averaged precision, recall and f1 score for the SVM model along 5 cross validation folds

Using the SVM algorithm an average precision of 0.805, an average recall of 0.801 and an average f1-score of 0.800 was obtained. The ROC AUC was of 0.93. These were surprisingly good results, and were a great improvement compared to the results yielded by the Naive Bayes algorithm.

5.3 Results for Random Forest

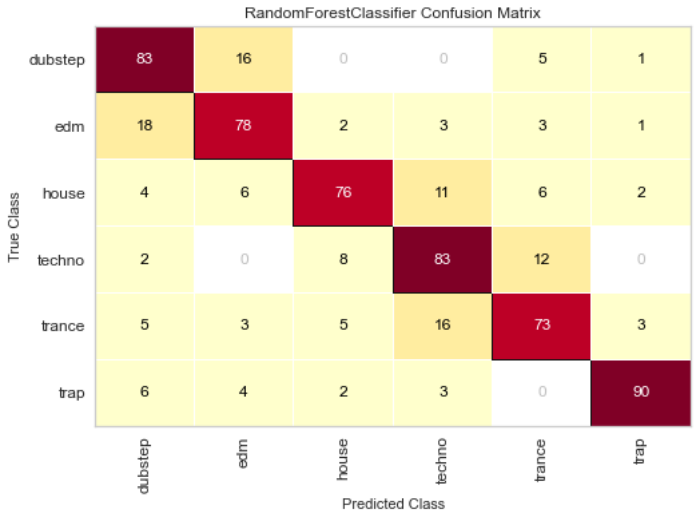


Figure 15 – Confusion Matrix for the Random Forest model

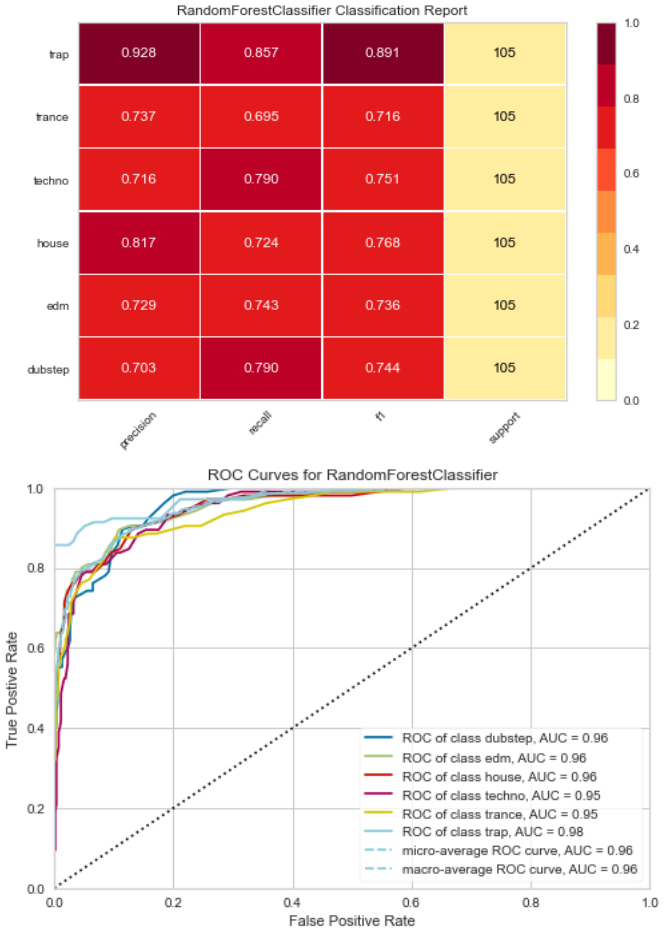


Figure 16 – Classification report and ROC curves for the Random Forest model

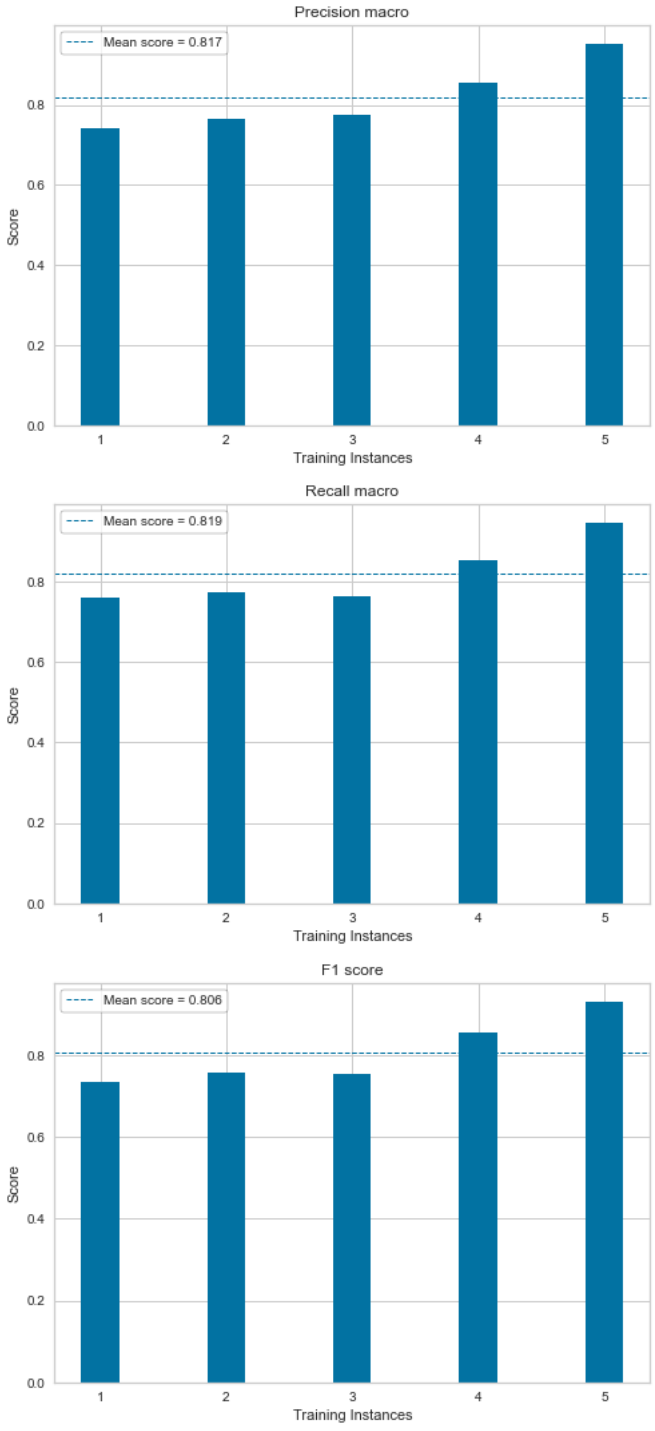


Figure 17 – Macro-averaged precision, recall and f1 score for the Random Forest model along 5 cross validation folds

For the Random Forest algorithm the results were: an average precision of 0.817, an average recall of 0.819 and an average f1-score of 0.806, with a ROC AUC of 0.96. Clearly, those were the best results from the three.

5.4 Results after tuning the hyper-parameters of the best performing model

As we can see above, the best performing model between the three, using the default hyperparameters, was the Random Forest algorithm. We then select it and try to find, from a set of hyperparameters, the ones that yeld the best results, so we can overcome the baseline performance. In the table below we present each of these hyperparameters, the set of tried values and a brief explanation of their meaning.

Hyperparameter	**Meaning**	**Tried values**
criterion	The function that measures the quality of a split	gini, entropy
n_estimators	Number of trees in random forest	200, 400, 600, 800, 1000, 1200, 1500
max_features	Number of features to consider at every split	auto, sqrt
max_depth	Maximum number of levels in tree	0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100
min_samples_split	Minimum number of samples required to split a node	2, 5, 10
min_samples_leaf	Minimum number of samples required at each leaf node	1, 2, 4
bootstrap	Method of selecting samples for training each tree	yes, no

Table 1 – Random Forest hyperparameters to be fine-tuned.

After trying 3 folds for 100 random combinations of these hyperparameters, we found out that the best combination was: *criterion:* entropy, *n_estimators:* 800, *max_features:* auto, *max_depth:* 10, *min_samples_split:* 2, *min_samples_leaf:* 2, *bootstrap:* no.

5.4.1 Improved performance

Using the best combination of hyperparameters yielded, described above, the Random Forest algorithm obtained the following results:

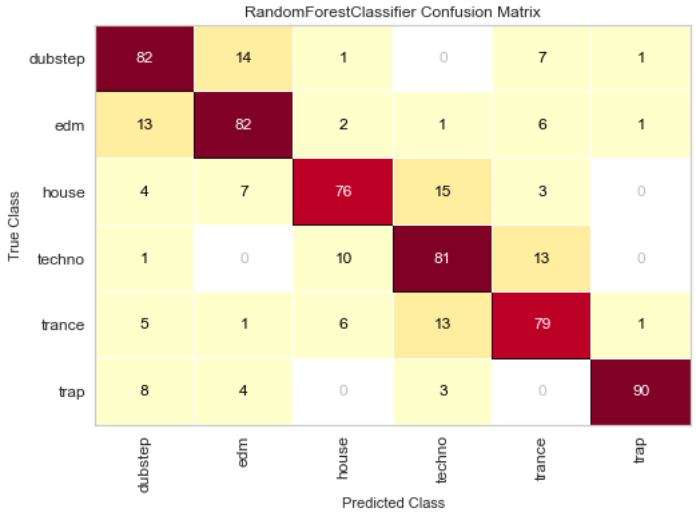


Figure 18 – Confusion Matrix for the improved Random Forest model

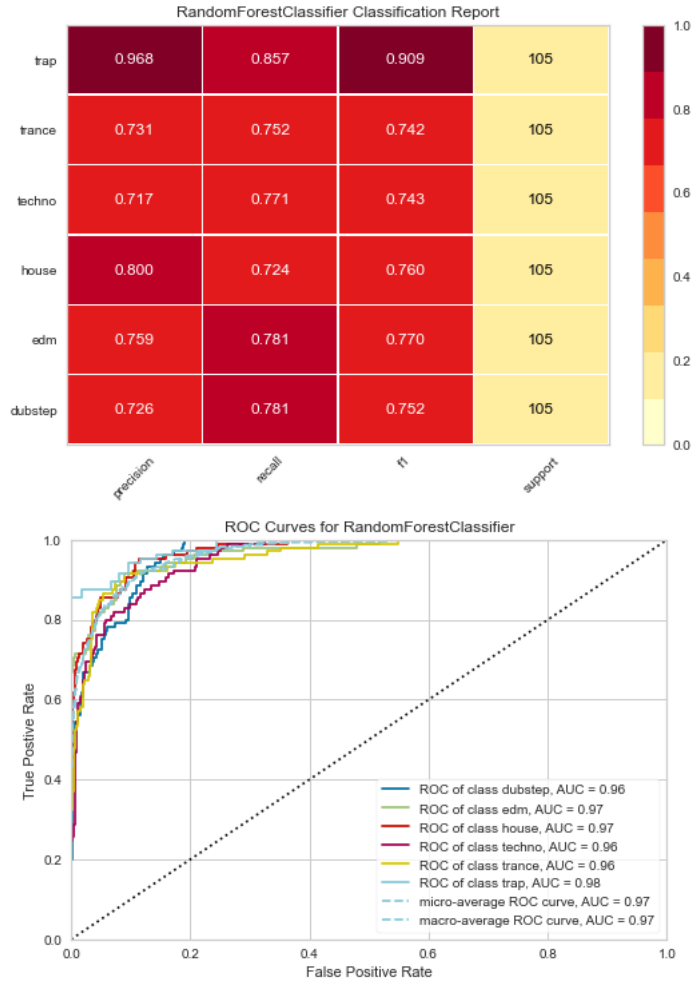


Figure 19 – Classification report and ROC curves for the improved Random Forest model

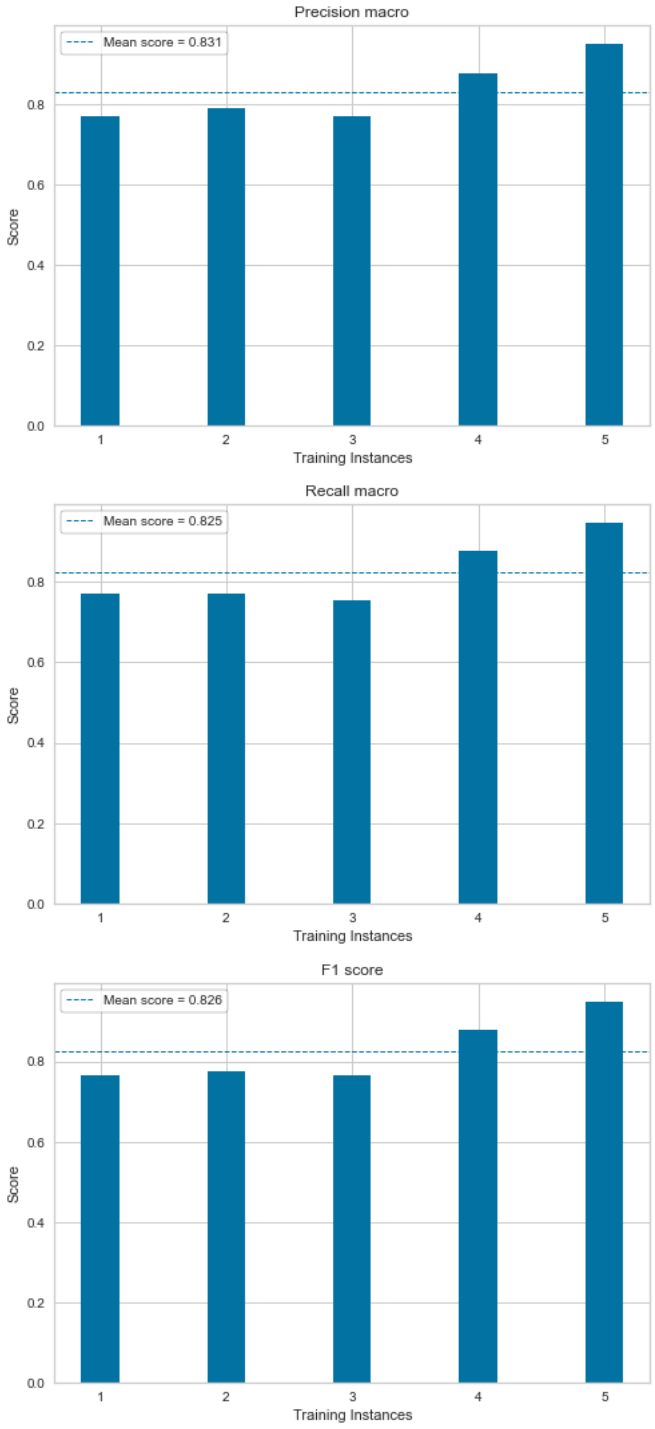


Figure 20 – Macro-averaged precision, recall and f1 score for the improved Random Forest model along 5 cross validation folds

As we can see, the average precision and the average recall had satisfactory improvements, from 0.817 to 0.831; and from 0.819 to 0.825, respectively. The average f1-score went from 0.806 to 0.826. The average ROC AUC went from 0.96 to 0.97.

5.4.1.1 20 top features

Finally, we present the 20 top features for our fine-tuned final model.

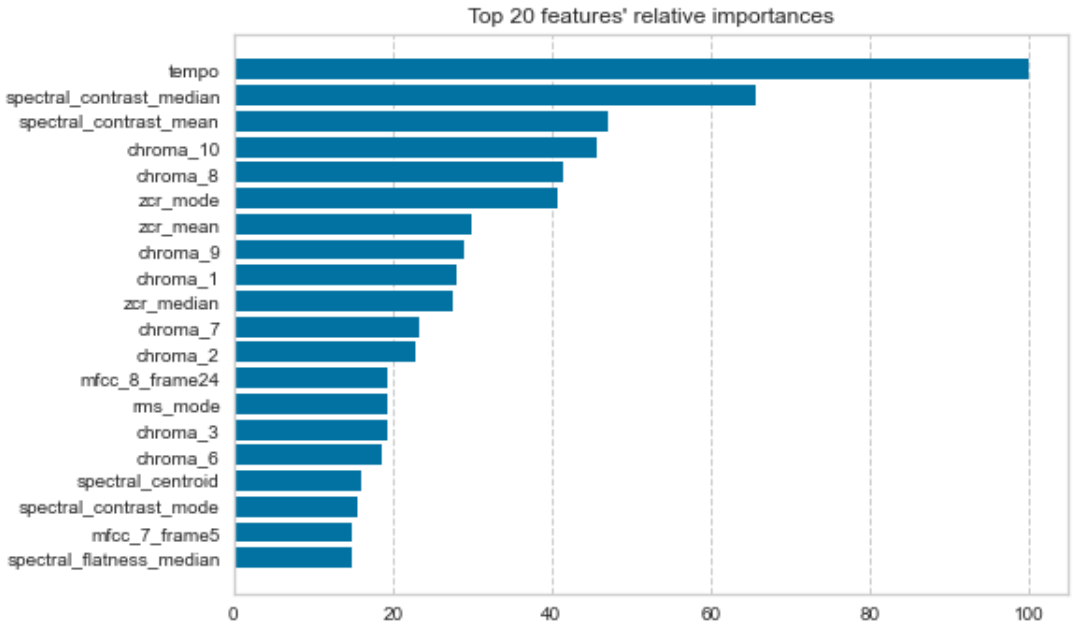


Figure 21 – 20 top features for the improved Random Forest.

Surprisingly, the tempo feature was the most relevant feature for the model. This was unexpected due to the fact that many genres share the same characteristic tempo and, therefore, this feature shouldn't be that relevant in distinguishing between them.

The spectral contrast, which measures broad and narrow band noise, was the second most important feature. Since this feature tells a lot about the texture of sounds, it was expected for it to be that important.

Finally, what we can observe in Figure 21 is the great importance of chroma features, what raises an evidence that tonal characteristics of a song are very important to define a genre.

6 Conclusion and future works

In the present study we have implemented and evaluated a music genre classification model using spectral features extracted from music audio files. The tests were made using three classification algorithms and, at the end, one of them — namely, the Random Forest — was selected and its hyperparameters were fine-tuned in order to improve its results.

The results were very satisfactory since the final model performed very well according to the used evaluation metrics — namely, precision, recall, f1-score and ROC AUC. Given the small ammount of samples used in the training and testing datasets and the achieved results, it was shown that it is possible to develop good automatic music genre classification models capable of classifying new songs into a music collection adequately using not much training data, what is a very interesting result for practical applications. The good results yielded also showed the relevance of spectral features when it comes to describing a piece of audio.

Furthermore, the results regarding the most important features for the final model provided many insights about which characteristics of a song better describe it. It was shown that, at least for the set of samples used in the tests for the presented study, the tempo of a song is extremely important, as well as the spectral contrast and tonal characteristics. The MFCCs, widely used in machine leaning problems involving audio, on the other hand, were not as important as those simple features, what shows that complex features are not always the best ones.

Giving the promising results yielded by the present study, it is intended to develop an application to auto-tag and organize song files according to their genre using the classification model developed in this project. The implemented model would be very useful if applied to an automation tool and would help a lot in the decision of which genre to attribute to a song that has characteristics of multiple genres. Also, it is intended to extend the research and improve the developed model so it is capable of dealing with a harder problem that is classifying subgenres, since these are much less distinct. Thus a more robust model would be created, providing better practical results and better insights about all the relevant concepts to the task of music genre classification.

Bibliography

Al-Shoshan, Abdullah I. (2006). "Speech and Music Classification and Separation: A Review". In: *Journal of King Saud University - Engineering Sciences* 19.1, pp. 95 –132. ISSN: 1018-3639. DOI: https://doi.org/10.1016/S1018-3639(18)30850-X. URL: http://www.sciencedirect.com/science/article/pii/S101836391830850X.

Bhattacharjee, M., S. R. M. Prasanna, and P. Guha (2020). "Speech/Music Classification Using Features From Spectral Peaks". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28, pp. 1549–1559.

Billboard (2015). *Global EDM Market Hits $6.9 Billion*. Billboard. URL: https://www.billboard.com/articles/business/6575901/global-edm-market-hits-69-billion (visited on 05/31/2020).

— (2018). *Report: global EDM industry now worth $7.1b, down 2% but has sustainable wide-scale appeal*. Billboard. URL: https://www.billboard.com/articles/business/6575901/global-edm-market-hits-69-billion (visited on 05/31/2020).

Brown, Judith and Miller Puckette (Nov. 1992). ""An efficient algorithm for the calculation of a constant Q transform"". In: *Journal of the Acoustical Society of America* 92, p. 2698. DOI: 10.1121/1.404385.

Deezer. Deezer. URL: https://www.deezer.com/ (visited on 06/09/2020).

Equal-Tempered Scale. *Equal Temperament*. Georgia State University. URL: http://hyperphysics.phy-astr.gsu.edu/hbase/Music/et.html (visited on 06/09/2020).

Farrokhmanesh, Mehrdad and Ali Hamzeh (May 2018). "Music classification as a new approach for malware detection". In: *Journal of Computer Virology and Hacking Techniques*, pp. 1–20. DOI: 10.1007/s11416-018-0321-2.

Fletcher, Harvey (1938). "Loudness, Masking and Their Relation to the Hearing Process and the Problem of Noise Measurement". In: *The Journal of the Acoustical Society of America* 9.4, pp. 275–293. DOI: 10.1121/1.1915935. eprint: https://doi.org/10.1121/1.1915935. URL: https://doi.org/10.1121/1.1915935.

Fletcher, Harvey and W. A. Munson (1937). "Relation Between Loudness and Masking". In: *The Journal of the Acoustical Society of America* 9.1, pp. 78–78. DOI: 10.1121/1.1902030. eprint: https://doi.org/10.1121/1.1902030. URL: https://doi.org/10.1121/1.1902030.

Fu, Z. et al. (2011). "A Survey of Audio-Based Music Classification and Annotation". In: *IEEE Transactions on Multimedia* 13.2, pp. 303–319.

Hand, David and Keming Yu (May 2007). "Idiot's Bayes: Not So Stupid after All?" In: *International Statistical Review* 69, pp. 385 –398. DOI: 10.1111/j.1751-5823.2001.tb00465.x.

Kaggle. Kaggle. URL: https://www.kaggle.com/ (visited on 06/09/2020).

Kanungo, T. et al. (2002). "An efficient k-means clustering algorithm: analysis and implementation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7, pp. 881–892.

Lee, Jongpil et al. (2017a). *Sample-level Deep Convolutional Neural Networks for Music Auto-tagging Using Raw Waveforms*. arXiv: 1703.01789 [cs.SD].

— (2017b). "Sample-level Deep Convolutional Neural Networks for Music Auto-tagging Using Raw Waveforms". In: *CoRR* abs/1703.01789. arXiv: 1703.01789. URL: http://arxiv.org/abs/1703.01789.

LibROSA Python Package (2020). *LibROSA*. URL: https://librosa.github.io/librosa/ (visited on 06/09/2020).

McKay, Cory and Ichiro Fujinaga (Jan. 2006). "Musical Genre Classification: Is It Worth Pursuing and How Can It be Improved?" In:

Müller, Meinard. *Fourier Tempogram*. AudioLabs Erlangen. URL: https://www.audiolabs-erlangen.de/resources/MIR/FMP/C6/C6S2_TempogramFourier.html.

Müller, Meinard and Sebastian Ewert (Jan. 2011). "Chroma Toolbox: Matlab Implementations for Extracting Variants of Chroma-Based Audio Features." In: pp. 215–220.

Nam, J. et al. (2019). "Deep Learning for Audio-Based Music Classification and Tagging: Teaching Computers to Distinguish Rock from Bach". In: *IEEE Signal Processing Magazine* 36.1, pp. 41–51. DOI: 10.1109/MSP.2018.2874383.

Saad, Farag (2014). "Baseline Evaluation: An Empirical Study of the Performance of Machine Learning Algorithms in Short Snippet Sentiment Analysis". In: *Proceedings of the 14th International Conference on Knowledge Technologies and Data-Driven Business*. i-KNOW '14. Graz, Austria: Association for Computing Machinery. ISBN: 9781450327695. DOI: 10.1145/2637748.2638420. URL: https://doi.org/10.1145/2637748.2638420.

Silva, Angelo C. Mendes da, Mauricio A. Nunes, and Raul Fonseca Neto (2019). *A Music Classification Model based on Metric Learning and Feature Extraction from MP3 Audio Files*. arXiv: 1905.12804 [cs.SD].

Spotify. Spotify. URL: https://www.spotify.com/ (visited on 06/09/2020).

Spotify Web API (2020). Spotify. URL: https://developer.spotify.com/documentation/web-api/ (visited on 06/08/2020).

Srivastava, Durgesh and L. Bhambhu (Feb. 2010). "Data classification using support vector machine". In: *Journal of Theoretical and Applied Information Technology* 12, pp. 1–7.

Stevens, S. S. and J. Volkmann (1940). "The Relation of Pitch to Frequency: A Revised Scale". In: *The American Journal of Psychology* 53.3, pp. 329–353. ISSN: 00029556. URL: http://www.jstor.org/stable/1417526.

Stevens, S. S., J. Volkmann, and E. B. Newman (1937). "A Scale for the Measurement of the Psychological Magnitude Pitch". In: *The Journal of the Acoustical Society of*

America 8.3, pp. 185–190. DOI: 10.1121/1.1915893. eprint: https://doi.org/10.1121/1.1915893. URL: https://doi.org/10.1121/1.1915893.

STFT (2020). *The Short-Time Fourier Transform*. Stanford. URL: https://ccrma.stanford.edu/~jos/sasp/Short_Time_Fourier_Transform.html (visited on 06/09/2020).

Tin Kam Ho (1995). "Random decision forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1, 278–282 vol.1. DOI: 10.1109/ICDAR.1995.598994.

Tonnetz. *Tonnetz*. Imaginary. URL: https://imaginary.org/de/node/1523.

Xu, Min et al. (2005). "HMM-Based Audio Keyword Generation". In: *Advances in Multimedia Information Processing - PCM 2004*. Ed. by Kiyoharu Aizawa, Yuichi Nakamura, and Shin'ichi Satoh. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 566–574. ISBN: 978-3-540-30543-9.