Homework 1

April 26, 2021

Group 8
Person codes: 10805780 – 10582571 – 10782704 – 10531049

Github Repository: www.github.com/enricovalente23/CMLS-Homework-1-Group-8.git

Presentation of the work

Our task for this homework was to prepare a classifier that predicts the musical genre of a music piece. To do this, we used the GTZAN Genre Collection* , a database of songs divided by genre, and prepared supervised learning models. Out of the 10 genres in the database, we were supposed to classify only 4 genres from the available ones: **Classical**, **Disco**, **Jazz**, **Country**.

The general pipeline we followed during our work was composed of these steps:

- **Collect data** from the folders of the original dataset;
- Splitting the dataset in **training set** and **testing set**;
- **Extracting the useful features.** We aggregated all of them in a single result per audio track for the SVM model, and used them separately for 2D Convolutional Neural Network.
- Applying **feature selection methods**;
- Applying **preprocessing** to input data;
- Choosing **classification methods**.

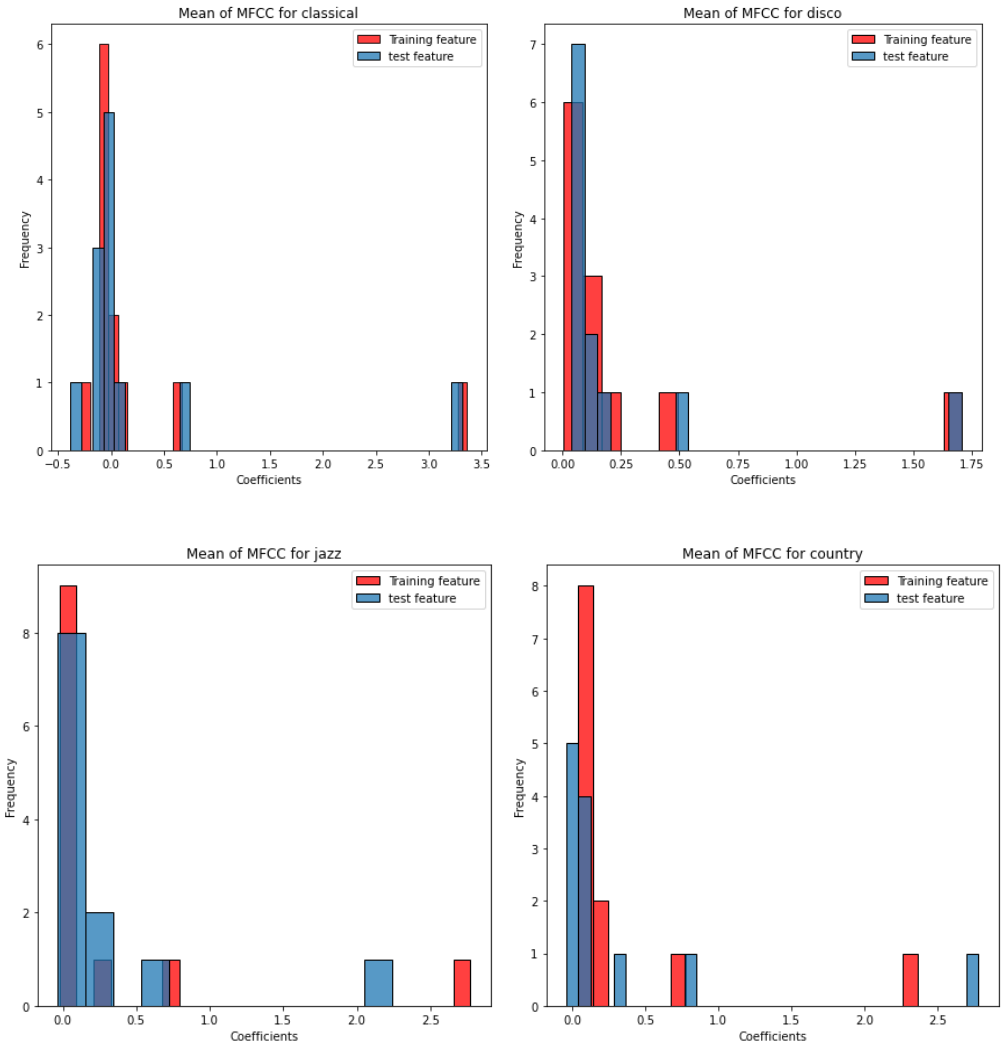
To **extract features**, we mainly used **Librosa***.

In order to **define and train** our models, we found several classifiers, regressors, feature selection algorithms implemented in the **scikit-learn** library for SVM and preprocessing for 2D Convolutional Networks and, **tensorflow** for creating the 2D Convolutional Networks.

Data collection

The ***genre*** is a high-level subjective descriptor, so we needed to build a relationship between low-level features and high-level features to fill the semantic gap.

We splitted the samples for every genre in two subfolders: **train** (66% of the data) and **test** (34% of the data). We simply split the audio tracks into a training set and a testing set without adding any constraints because we ensured that our dataset was balanced, as we can see in the following plots.



After that, we split each file into smaller audio segments using the segment_size variable and created a function called `complete_array` to make sure that every audio array was 30 seconds long instead of 29.999 seconds (by padding zeros at the end of the shorter arrays to ensure consistency for the database).

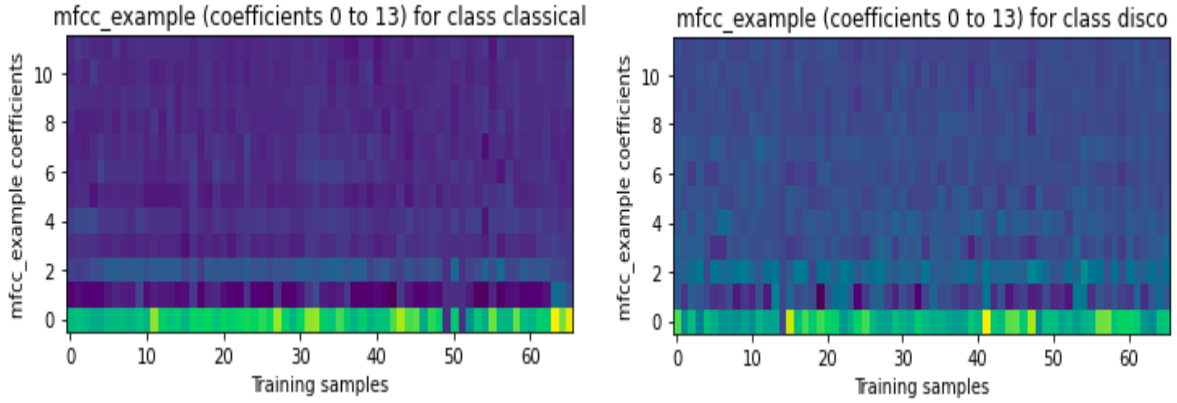
Feature Selection

We selected the following features in our analysis:

a. MFCC
b. Chroma
c. Tempo
d. Spectral Centroid Mean
e. Spectral Centroid Standard Deviation
f. Zero Crossing Rate
g. Spectral Roll-Off

a) The **Mel Frequency Cepstrum** (MFC) 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 non-linear mel scale of frequency. In our function **`compute_mfcc`**: we first compute the

STFT of the audio signal in order to get its spectrogram, then find the weights of the mel filters and we apply it to spectrogram; finally take its logarithm and we apply the Discrete Cosine Transform to obtain the final MFC Coefficients.



Looking at MFC averages for input, we can say that MFC Coefficients are differentiable between different classes. For example, for classical, the intensity is concentrated greatly at the first MFCC coefficient while for disco, the distribution is more even. This means that the model will benefit from having MFCC as a feature to distinguish these two genres.

b) For computing the **Chromagram** we used Librosa's function `chroma_stft` (**`librosa.feature.chroma_stft`**). Chroma-based features are a powerful tool for analyzing music whose tuning approximates to the equal-tempered scale. One main property of Chroma features is that they capture harmonic and melodic characteristics of music, while being invariant to changes in timbre & instrumentation which is very relevant in our case since there are great differences in instrumentation across genres.

c) To compute the **Tempo**, we used a dynamic programming beat tracker function from Librosa called **`librosa.beat.beat_track`**. Beats are detected in three stages, following the method of measuring onset strength, estimate tempo from onset correlation and then picking peaks in onset strength approximately consistent with estimated tempo.

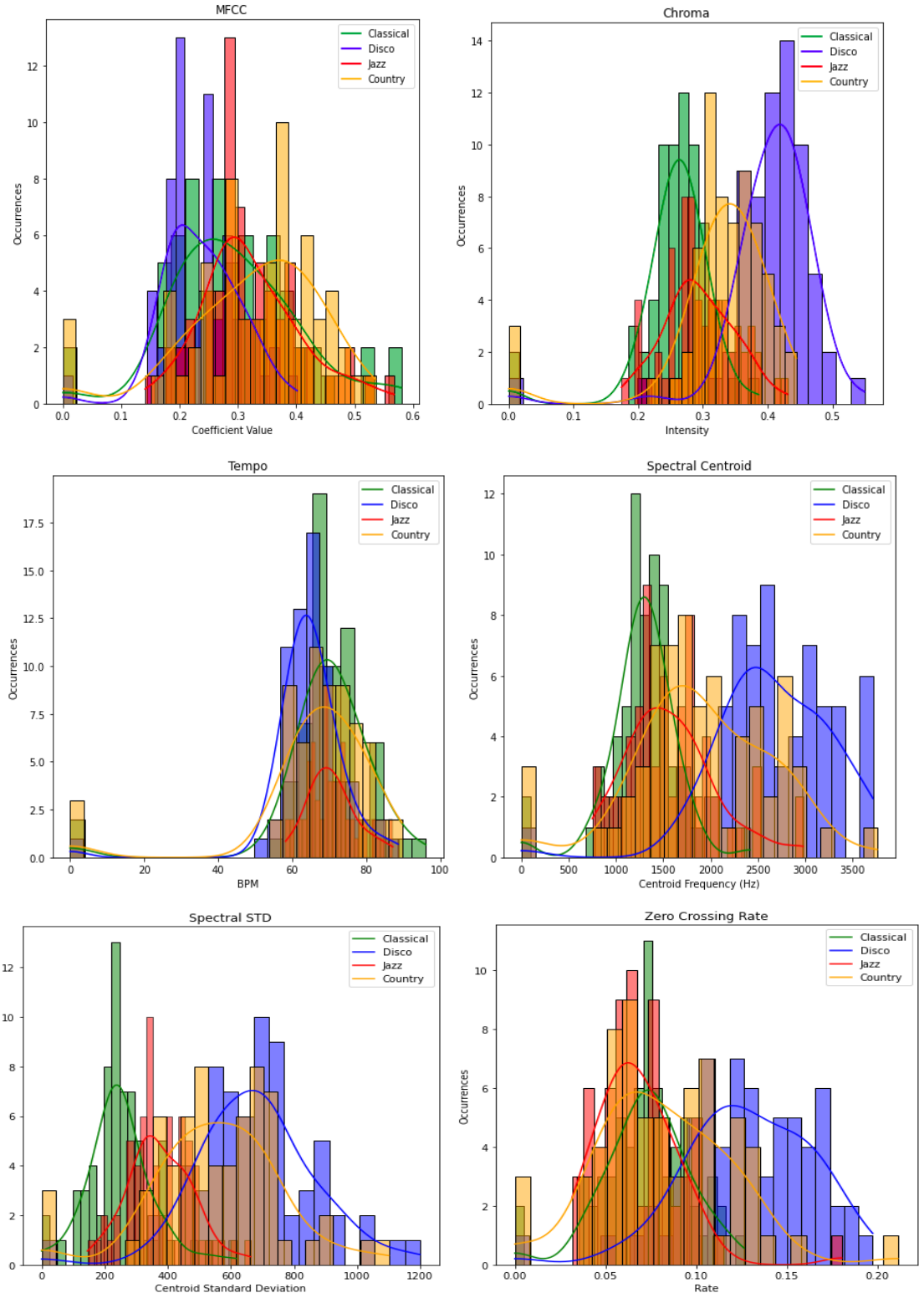
d) In order to calculate the **Spectral Centroid Mean** and, we made use of Librosa's `spectral_centroid` (**`librosa.feature.spectral_centroid`**). Each frame of a magnitude spectrogram is normalized and treated as a distribution over frequency bins, from which the mean (centroid) is extracted per frame.

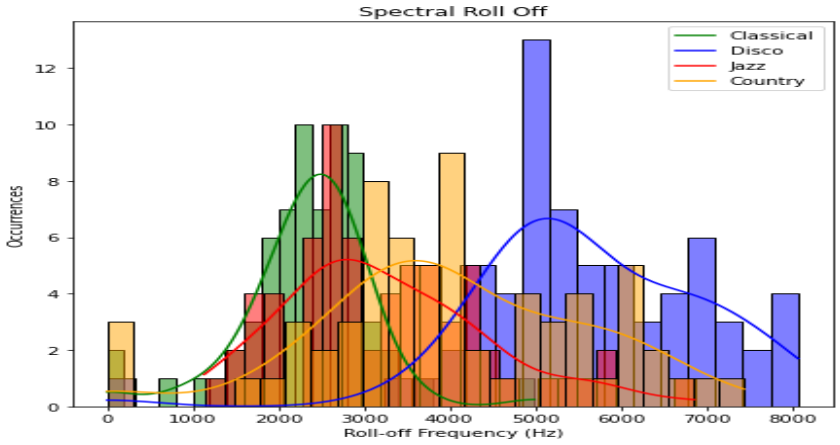
e) The **Spectral Centroid Standard Deviation** was simply calculated through the numpy's `np.std` function.

f) The **Zero Crossing Rate** is computed counting the number of times that the audio waveform crosses the zero axis. We implemented it using Librosa's feature `zero_crossing_rate` (**`librosa.feature.zero_crossing_rate`**).

g) The **Spectral Roll-Off Frequency** is defined for each frame as the center frequency for a spectrogram bin such that at least the roll-off percentage (0.85%) of the energy of the spectrum in this frame is contained in this bin and the bins below. We used Librosa's `spectral_rolloff` function (**`librosa.feature.spectral_rolloff`**).

Our comparison between the different genres with respect to the same feature shows that temporal modulations of features are important for the classification of audio and music:





Looking at the histogram of our results, we can say that most of our features are separated for different genres. For example the mean of MFCC coefficients and chromagram are so well separated that they end up performing the best in the SVM. Zero-crossing-rate and Tempo however, are not so clearly separated because we can see there is so much overlap in the same region. Because of this, we tried excluding these two features into the Combination matrix. The results of this experimentation is discussed in the results section.

We thought that chromagram might actually make the results worse since the tonality is irrelevant for genre classification and this can mislead the model. This is because Chromagram detects the semitones intensity in time, so a simple difference in tonality could return totally different results as some notes may not be used at all depending on the key and the scale used in the musical piece. A solution might be to choose all the pieces in the same key for all training and testing or take attention to the distribution of the keys in each genre. Regardless this is probably a small issue.

Preprocessing stage

Before we move onto creating our models and classifying our dataset, it is important to tweak the dataset so that it is consistent and better suited for training. We did two preprocessing steps:

a) **Normalization**
b) **Principal Component Analysis (PCA)**

a) The normalization ensures that the distributions of features are not different for each feature. If they were different, the model would need to learn the specific weight adjustment for each feature which adds unnecessary complexity to our model. By normalizing the features, we ensure that the model is able to adjust the weights for each class more consistently and achieves high model accuracy faster.

We used the following scheme for normalization:

```
x_test_0_normalized=(x_test_0 - feat_min) / (feat_max - feat_min).
```

b) We used PCA to simplify our feature set and decrease the number of features so that the model is better fitted. PCA algorithm tries to find a lower dimensional subspace that "best" approximates the original feature vector in a least square sense. It is obvious that some features of the audio are interrelated. For example, the mean of Spectral Centroid and the Standard Deviation of Spectral Centroid are highly correlated. By applying PCA, we can reduce and “combine” these two features into one, which almost makes up for not having two different features. This way, the unnecessary linearity residing in the database is gone and the model is able to distinguish each class easier and achieves high model accuracy faster.

Classification Methods

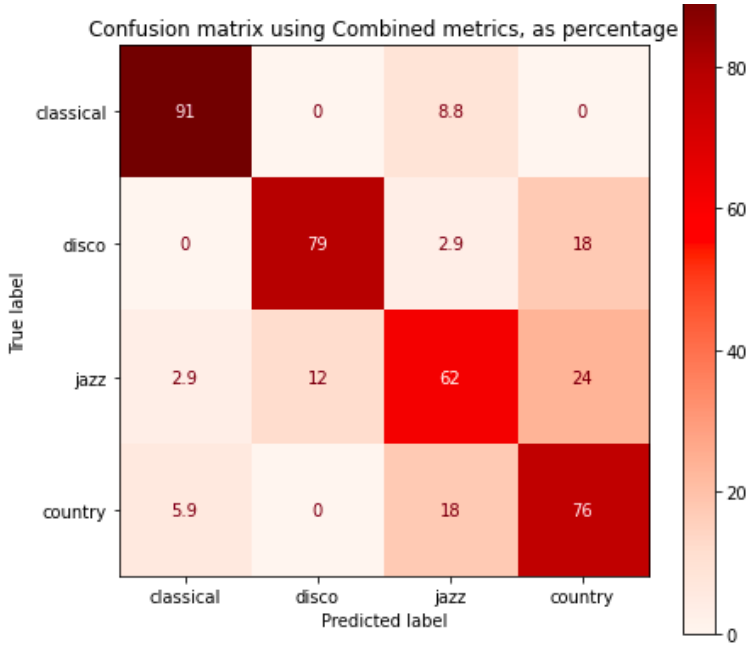
Support Vector Machine is a binary classifier that learns the boundary between items belonging to different classes. SVM’s are organized as a set of supervised learning methods used for classification, regression and outliers detection. Some of their advantages are that they are effective in high dimensional space and in cases in which the number of dimensions is greater than the number of samples, this is very helpful in our case because we have a lot of features but not many files.

Another supervised learning method we used is deep learning network, in particular Convolutional Neural Networks (CNN). These networks work by taking a 2D image and applying moving adaptive filters that change their weights by backpropagation. These filters try to observe the patterns inside the images like edges or color gradients. Each “neuron” has activation layers that are connected to other neurons that detect even more complex patterns. By adjusting the weights over time, the network tries to get the best accuracy relative to the validation dataset given. To use CNN, we had to slice each 30 seconds audio file into smaller audio bits that are connected to the same output tag. We extracted features out of the slices to create “audio images”, that are stored inside a 4D array (genre*file*2D images of slice vs feature). We experimented with slices of 5 seconds and 10 seconds but found out that the best accuracy is achieved by using 2 seconds slices.

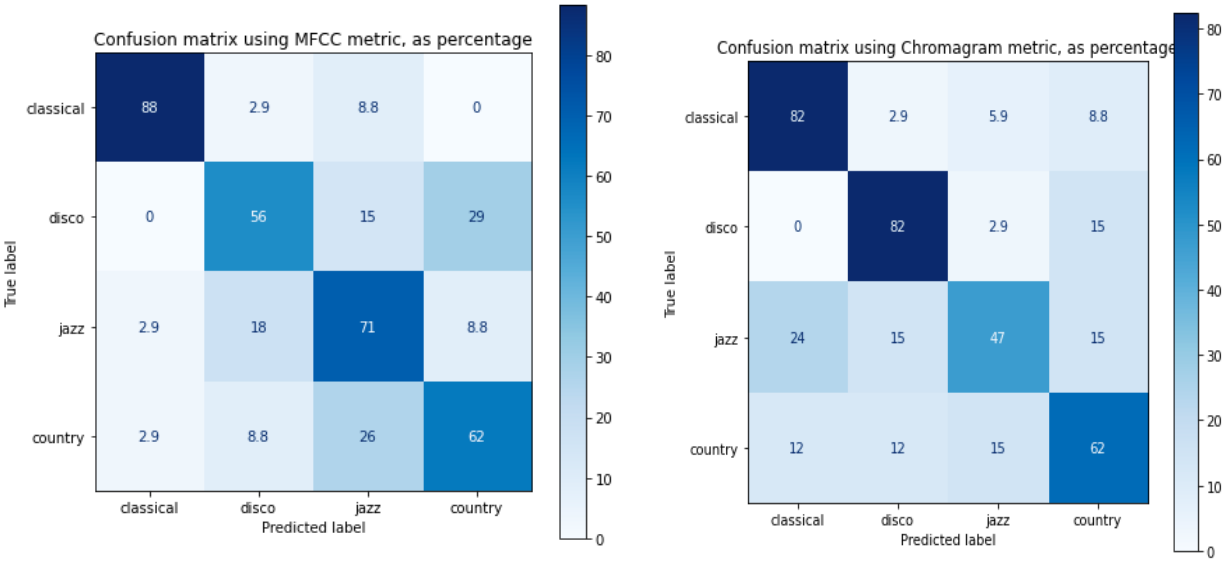
Lots of architectures have been tested with very big filter sizes and many filters, but they usually overfit the data, so the current configuration gave the best results. We also tried changing the learning rate and early stopping algorithms but none of them significantly helped.

Results

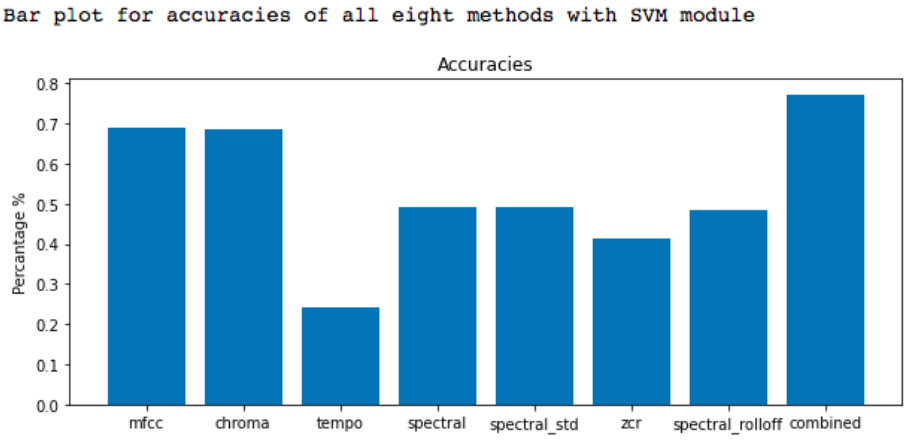
The SVM model with the combined features gave the best results. Its confusion matrix is:



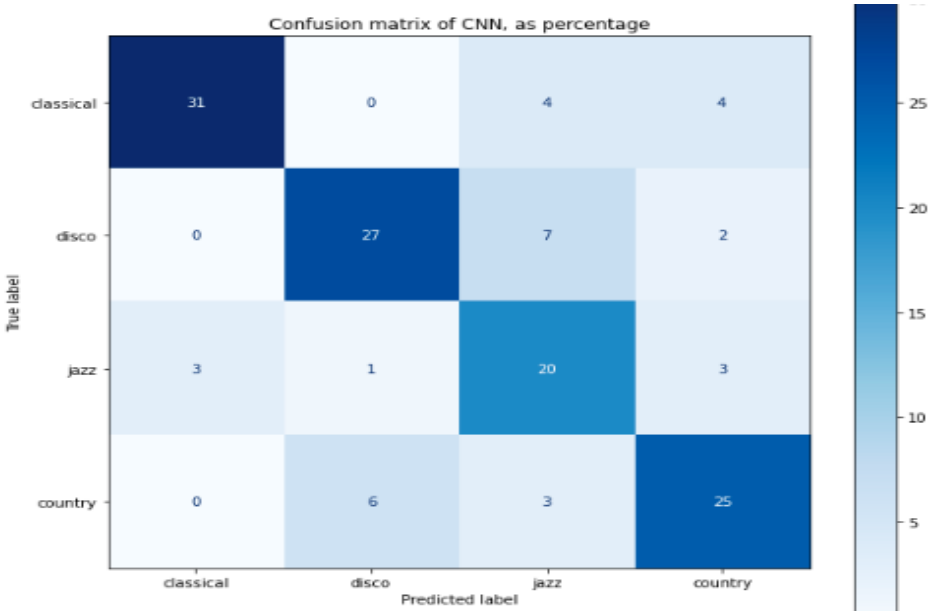
With 78% precision and 77% recall overal. The other two best features were MFCC and Chromagram given as:



As we can see the overall accuracy of the MFCC was higher but the particular disco class was detected better by Chromagram. This explains why the combined set was the best: it might use different features and combinations to better detect each genre. One interesting thing was that even though the tempo and ZCR's precision was only 46% and 36%, without them, the model performed only slightly better at 79% precision. This means that SVM is good at ignoring bad features overall.



Since the combined features performed the best, we only tried the combined set as an input for the CNN model. The results were good with precision and recall at 76%.



Interestingly, the CNN models performed very badly when we took out the Tempo and ZCR from the combined matrix. This might mean that it is somehow using those two features into good use.

We also tried running the same CNN network with the PCA data at 10 number of PCA components. This failed horribly compared to other models and resulted in 38% accuracy. This might be because the features are so well separated and there is so little redundancy in the feature set that by taking PCA we are only losing information.

Overall, we noticed that the combination of all the analyzed features is the most powerful classification method and SVM and CNN both worked very well for this classification task.