Sound classification of the UrbanSound8K

David Pérez Ruiz
david.perez.ruiz@estudiantat.upc.edu

Gerard Franco Panadés
gerard.franco@estudiantat.upc.edu

Fàtima El Baghdadi
fatima.el.baghdadi@estudiantat.upc.edu

***Abstract*—This document is a report that analyzes the classification of 10 kinds of sounds extracted from the Urban Sound Dataset [1] using Neural Networks (NN) and Support Vector Machines (SVM). In addition, an "interface" is presented that makes it easier (just a few clicks as a matter of fact) to annotate new unlabeled data that were to make its way to the existing dataset.**

I. Introduction

The aim of the project is to develop a classifier capable of distinguishing between the several classes that the Urban Sound Dataset has. Prior to that, an exploratory analysis of the data is presented here along with the extraction of the features from the sound sources. The aim is to provide one same set of features to properly classify among all classes. The dataset has 10 different classes and two methods will be used to show that it can be classified as a multiclassification problem or as a binary problem. For the multiclassfication problem it has used the Neural Net and for the binary problem it has computed the support vector machine. These two methods are presented in the following paper explaining the decisions and the steps that are done. The code[1] implemented is made available in a Github repository. As previously explained, The Urban Sound Dataset has 10 different classes and are labeled (in the metadata files) as:

- 0 = air conditioner
- 1 = car horn
- 2 = children playing
- 3 = dog bark
- 4 = drilling
- 5 = engine idling
- 6 = gun shot
- 7 = jackhammer
- 8 = siren
- 9 = street music

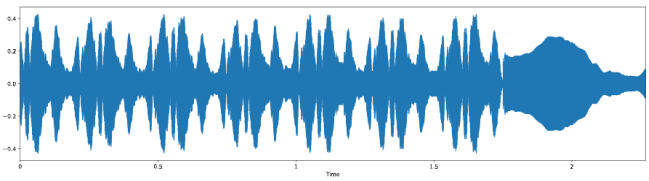

Fig. 1. Plot of the audio array in waveform manner.

II. Candidate features

In order to classify the different sounds presented above, it is necessary to extract the audio features. The Mir Toolbox [2] for Matlab has been used to extract the following features:

1 Zero Crossing Rate (1x1 dimension)
2 Spectral Centroid (1x1 dimension)
3 Spectral Rollof (1x1 dimension)
4 Mel Frequency Cepstrum Coefficient (MFCC) (13x1 dimension)
5 Spectral Flatness (1x1 dimension)
6 Skewness (1x1 dimension)
7 Kurtosis (1x1 dimension)
8 Spectral Entroypy (1x1 dimension)
9 Tempo (1x1 dimension)
10 Tonal centroids (6x1 dimension)

As it can be seen the dimension of the features is 27 features of every observation. Now, it will proceed to explain the meaning of every feature to get a basic idea.

A. Zero Crossing Rate

The zero crossing rate is the rate of sign-changes along a signal, i.e., the rate at which the signal changes from positive to negative or back. This feature has been used heavily in both speech recognition and music information retrieval. It usually has higher values for highly percussive 5 sounds like those in metal and rock.

B. Spectral Centroid

The spectral centroid is a measure of a signal that characterises spectrums. It is also used as a measure of brightness, and indicates where the center of mass of a signal is located using the Fourier transform's frequency and magnitude information. It can be seen an example of a spectral centroid (normalized) respect its waveform in figure 2:

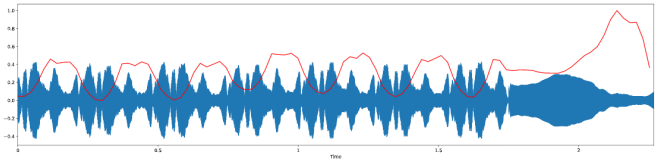

Fig. 2. Plot Spectral Centroid (red) with the waveform (blue).

[1] https://github.com/PittyPaladin/urban_sounds_classification

C. Spectral Rollof

Put in other words: spectral rollof returns the frequency for which 85% of the energy is contained (in this frequency and the ones below) for each frame. For example, if the frequency spectrogram at a frame t has up to 5000Hz bins but with bins up to 4000Hz 85% of the energy is included, then the rollof frequency is 4000Hz.

D. Mel Frequency Cepstrum Coefficient (MFCC)

MFCCs take into account human perception of sound to convert the traditional frequency audible ranges to Mel Scale in a nonlinear manner. There are several mel scale coefficients, and the Mir Toolbox takes 13 as default. They are mostly used in speech recognition, and are quite sensible to noise. They are the current state of the art feature (so to speak) in the speech recognition area. Moreover, it is presented the visual MFCC scaled in figure 3.

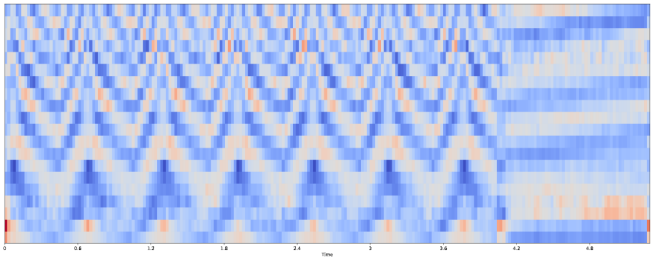

Fig. 3. MFCCs scaled such that each coefficient dimension has zero mean and unit variance

E. Mel Frequency Cepstrum Coefficient (MFCC)

Spectral flatness quantifies how tone-like a sound is, as opposed to being noise-like. Usually, it is measured in decibels.

F. Skewness

Skewness is a measure of the symmetry of the distribution. The skewness can have a positive value, in which case the distribution is said to be positively skewed with a few values much larger than the mean and therefore a long tail to the right.

G. Kurtosis

Kurtosis is a statistical measure that defines how heavily the tails of a distribution differ from the tails of a normal distribution. In other words, kurtosis identifies whether the tails of a given distribution contain extreme values.

Skewness essentially measures the symmetry of the distribution, while kurtosis determines the heaviness of the distribution tails as it can be seen in figure 4.

H. Spectral Entropy

Spectral Entropy is a measure of its spectral power distribution. It means that if the curve is extremely flat, corresponding to a situation of maximum uncertainty, then the entropy is maximal. Reversely, if the curve displays only one very sharp peak, above a flat and low background, then the entropy is minimal, indicating a situation of minimum uncertainty.

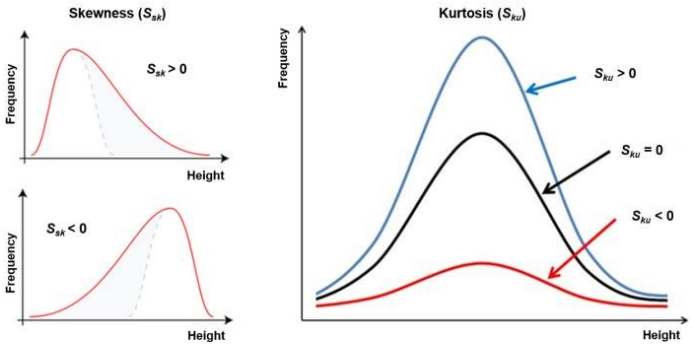

Fig. 4. Skewness and Kurtosis graphically explained [3].

I. Tempo

The feature Tempo can be defined as the pace or speed at which a section of music is played. It conveys a feeling of either intensity or relaxation. Typically, the speed of the music is measured in beats per minute, or BPM. For instance, a tempo notated as 60 BPM would mean that a beat 10 sounds exactly once per second. A 120 BPM tempo would be twice as fast, with two beats per second.

J. Tonal Centroids

Tonal centroids computes the tonal centroid features (tonnetz) as it can be seen in figure 5:

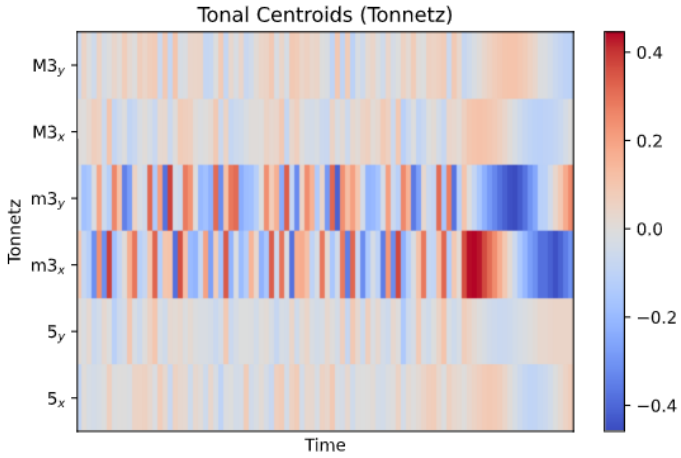

Fig. 5. Tonal Centroids of the audio selected in fig. 1.

III. Reduction of the features

Once it has extracted the features (that takes 9 hours approximately), the second step is their normalization. Moreover, as there are features of the dataset that are 'NaN', a function has been used to delete these observations.

As there are a lot of features, the first step is plotting the correlations among pairs of features: (Note: it has plotted only 7 features to explain some details, but there are 27 features)

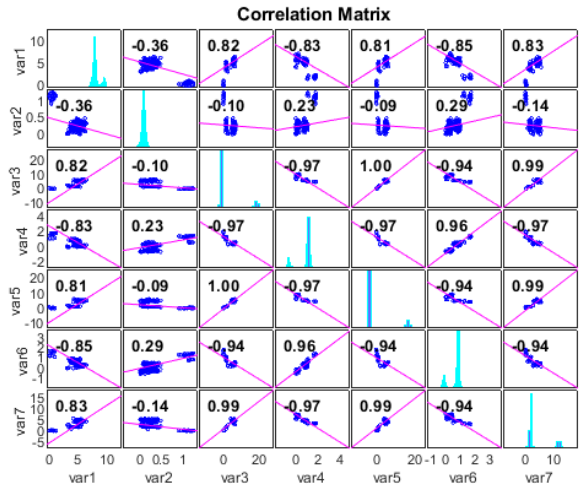

Fig. 6. Correlation matrix with 7 features of the original dataset.

As it can be appreciated in figure 6, there are features that are very correlated (for example var3 respect var1, var4, var5, var6 and var7) and maybe there isn't necessary using all the features. So, the next step is determining how many features are needed to explain the data. It is used the function covariance and the eigenvalues of the features to see which nº of features are needed to explain at least the 95% of the data.

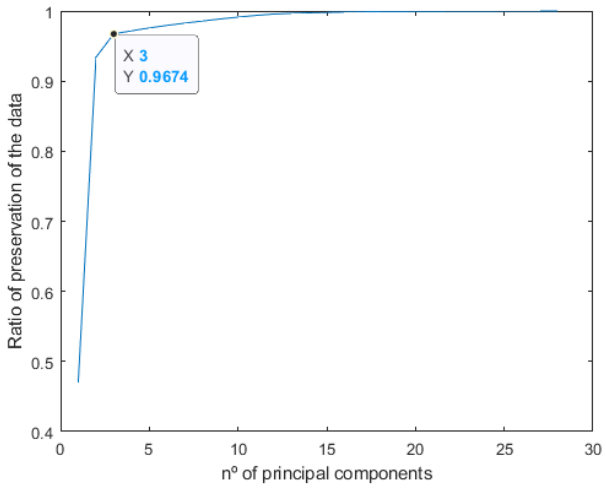

Fig. 7. nº of components that explain the data.

As it can be perceived, in figure 7, 3 components are needed to explain the 97% of the data. With this being said, the Principal Component Analysis (PCA) is applied to reduce the features to just 3 principal components (figure 8).

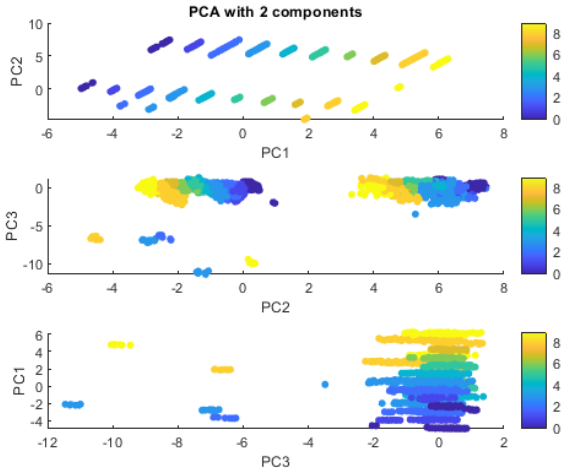

Fig. 8. Plots of the 3 components of PCA respect the type of class.

Also, it has computed the correlation matrix, in figure 9, to verify that the components are orthogonal and how is the data.

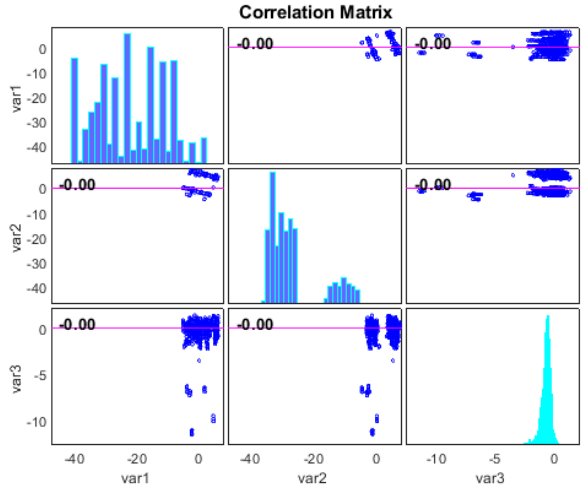

Fig. 9. Correlation matrix applying PCA.

IV. First Classifier: NN

The first classifier to train the model and predict is NN. The 80% of the data is to train the model because there are a lot of classes. The configuration of the hidden-Layers that gives better performance is: [10, 10, 10] (figure 10) .

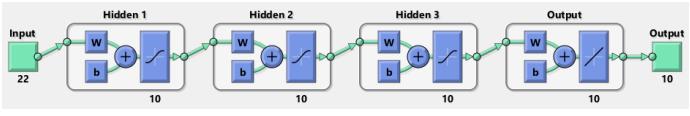

Fig. 10. Configuration of the network.

It has used the different types of training functions and it selects only 3 methods to explain them:

1 Levenberg-Marquardt

2 Bayesian Regularization
3 BFGS Quasi-Newton

The 3 methods update weights and bias values but using different ways, and some are faster than the others. The nº of epochs to train the models are 1000 (the number of rounds of optimization that are applied during training, as bigger as better to minimize the error in the training procedure) and it is used the mean squared error (MSE) to measure the network's performance. It has applied the NN to the training set using all the features and applying PCA:

Training functions	1	2	3
Accuracy(%) all features	100	100	24.9
Accuracy(%) applying PCA	66.1	100	77.8

As it can be seen, the second method has a very good performance, the first method decreases its accuracy applying PCA, and in the third method the accuracy increases. These examples have been chosen to illustrate the variations that occur when PCA is applied. As a first decision the second method is the best, but computationally is very expensive (duration 2 hours) instead of the first case that also has a good performance. In addition, it is presented the figure 11 that show the difference between the predictions using the method Levenberg-Marquardt applying PCA and the real target along the nº of observations.

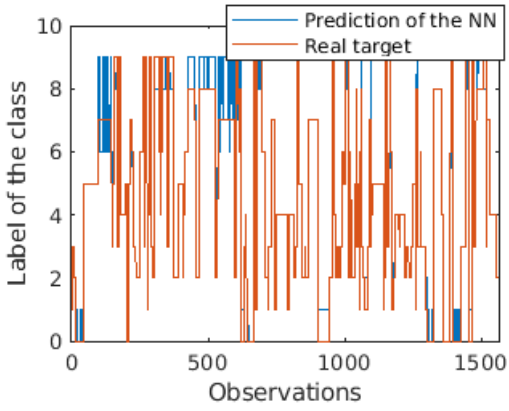

Fig. 11. Results between the prediction and real target using Levenberg-Marquardt.

The other functions did not have a good performance with the default settings, but modifying the nº of epochs, the learning rate, etc. it has obtained better results, but not as good as the methods commented. A remarkable example is the method "Gradient descent with momentum backpropagation" where using the default options (epochs = 1000, learning rate = 0.01 and momentum constant = 0.9) the accuracy was 18.82%. A momentum constant of value 0.2 has been considered, as this parameter helps slide through such minimum considering that if it is closer to 1, it indicates that is insensitive to the local gradient; therefore, does not learn properly. Another important fact is the learning rate: if is too large, can cause the model to converge too quickly to a suboptimal solution, so it has reduced the value of the learning to 0.0001. Finally, it has increased the value of epochs to 50000, because the accuracy was still lower. Despite that, tuning all these parameters and using several combinations, the value of the accuracy was 36.48%. So, it can be noted that there are cases when even trying to increase the accuracy is not possible to increase it a lot.

Note: to train the model and to eliminate rounding problems and make prediction easier to understand, the "ind2vec" function has been used to transform classes into vectors. For example: if the class is 2, then a vector of equal dimension of the existing number of classes (10 in this case) is created, with a value 1 in position 2 and value 0 in all other position. In this case, having the class 0, one has been added in all classes and after training and prediction one is subtracted to obtain the original classes. To finish with the analysis of this part, it will be calculated the confusion matrix (figure 12) when it is applied Levenberg-Marquardt with the reduced data that is related also with the figure 11: As it can be seen, it confuses

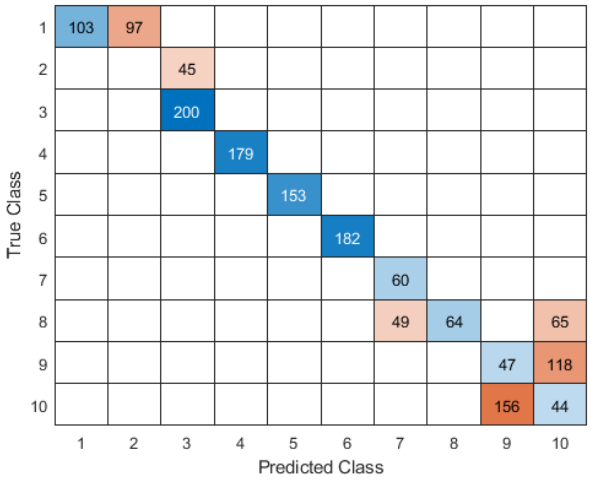

Fig. 12. Plot of the confusion matrix of NN applied to the validation data.

class 1 (air conditioner) with class 2 (car horn), the class 2 with class 3 (children playing), class 8 (jackhammer), class 7 (gun shot) and 10 (street music). Finally, it confuses 9 with 10 (siren with street music) and vice versa.
These confusions are possible because the sound "is similar" so the NN is confused in the classification. The key is trying to obtain the purest classes possible to classify them correctly.

V. Second Classifier: SVM

Another useful classifier is SVM, it finds a hyperplane in an N-dimensional space (N: the number of features) that distinctly classifies the data points. As the present problem is a multiclasification because there are 10 classes, it has used the method of "one vs all the others". This method consists in that the class that is going to train (for example: class 0) is assigned to its target +1 and all other classes (that are not 0) are assigned to their target -1. Having 10 classes, 10 models

are created to train each type of class and to predict the classes of the data test.

As the software that is using is Matlab, it is used the function 'OptimizeHyperparameters' that optimises automatically the parameters. As there are 10 models, it is computed 10 different models with its optimized parameters. Some important parameters that are optimized are:

- Alpha: contains the estimated Lagrange multipliers.
- Beta's: are the coefficients values of the SVM model: for example $y_i = f(x, \beta) = \beta_0 + \beta_1 X_1 + \beta_2 X_2$
- Bias: is the distance to the origin of the hyperplane solution.
- Box constraint: controls the maximum penalty imposed on margin observations, and aids in preventing overfitting (regularization). If the box constraint is big, then the SVM classifier assigns fewer support vectors. However, increasing the box constraint can lead to longer training times. The dimension depends on the number of observations of the dataset.
- Kernel function that in our case is the Radial Basis Function (RBF).
- Kernel scale that scale all elements of the predictor data on which the model is trained.
- Solver Optimization routine used to train the SVM classifier. In our case, always is Sequential Minimal 'SMO'.

Now it will show some graphs (figures 13, 14) obtained in the training and prediction of class = 2 to see how support vector machine works. In the figures 13 and 14 it can be seen

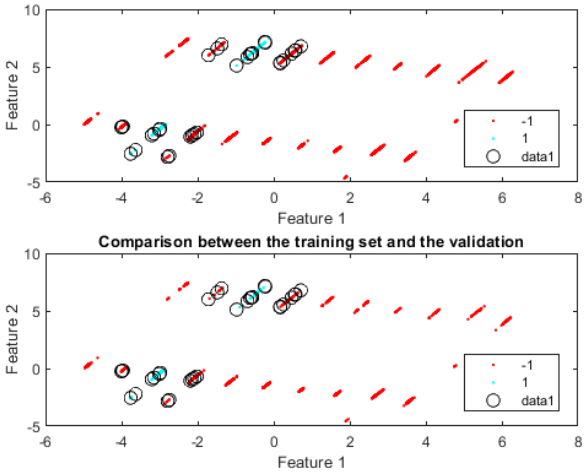

Fig. 13. Plot of the first and second features and support vectors for the training (above) and testing (below).

that the class 2 is coloured with cyan and rest of the classes are coloured with red. The support vector machine indicates the limits of the "circle" that contains the class. The results obtained have sense because once it crosses the other side of the support vector it can be found that there is another class. The function that is optimized in this case is plotted in figure 15.

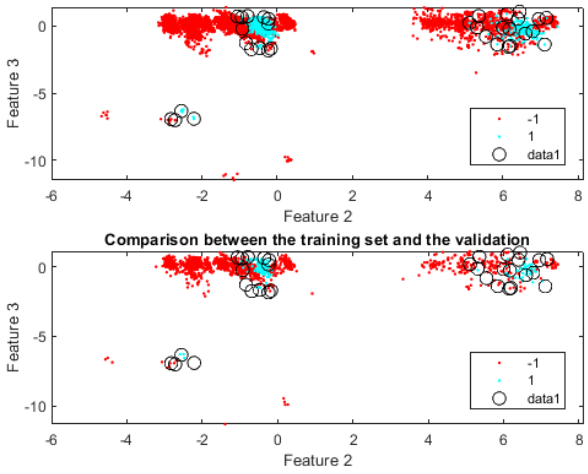

Fig. 14. Plot of the second and third features and support vectors for the training (above) and testing (below).

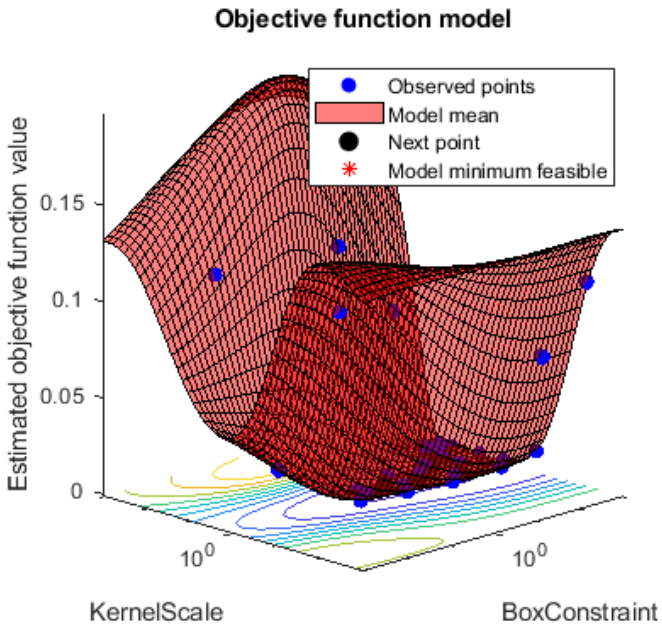

Fig. 15. Plot of the objective function when the class is 2.

The accuracy obtained with SVM is 99.88%, which is a very good performance for the validation data. Now, it will compute the confusion matrix to see in which cases it fails: As it can be observed in figure 16, all the classes are well classified except two observations that don't belong to any class (the class 11 exists when a observation doesn't belong to any class).

Note: In the training that has been tested initially, using the same observation in different models gave different classes, so it has assigned the class where the scores were higher.

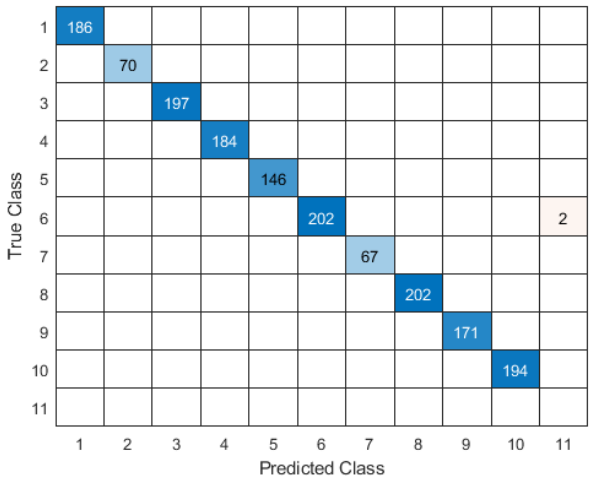

Fig. 16. Plot of the confusion matrix of the 10 different models applied to the validation data.

VI. Interface to classify the new sounds

As mentioned, an interface is created to annotate new incoming data to the existing dataset. The goal of making a graphical user interface (GUI) is to make it easier and less time consuming for the user to store a label for an audio track inside a `.csv` metadata file like the one given in the Urban Sound dataset. Up to this point, if a new track is added it has to be stored in one of the folders. Then its name and label (among other data) is stored in the aforementioned metadata file manually. That is certainly time consuming, specially taking into account that the two classifiers explained in this report are supervised, and require a large amount of data to achieve satisfying results.

The interface presented here is made up of 10 buttons, a play/pause button and a volume and playback slider. Once the user opens the app, an audio track needs to be selected. After that, a metadata dictionary is selected (it has to be already created and in `.csv` format). Once this is done, the user can play the audio. Once the audio is identified, the button corresponding to its class is pressed and automatically a label for that audio will be appended to the previously selected metadata file. Anyway, all steps are detailed in the Help section of the app. A glimpse of how the app looks is presented in figure 17.

As a sidenote, this interface has been created with the framework wxPython, which is a cross-platform GUI toolkit for the Python language. This framework has been chosen due to its capability to create native user interfaces that run with little or no modifications on Windows, Macs and Linux or other unix-like systems.

VII. Conclusions

The objectives of this project have been analysing a dataset, extracting the features and selecting the most important ones to work with. As it has been proved, a lot of features increase the computation time and can also give a worse performance. Another goal is applying the methods studied during the course and prove how they change the performance tuning their different parameters. As it has been explained previously, tuning the parameters of the NN, like cost functions, epochs, learning rate, constant of gamma, etc. makes the performance change. The best results that have been obtained are using the Levenberg-Marquardt and Bayesian Regularization functions with 1000 epochs. With respect to the SVM classifier, it has been observed that it produces very good performance using the method "one vs all" alongside a built-in Matlab function that auto-tunes the hyperparameters. The computational cost is lower using the SVM with respect to the NN, however the accuracy using NN is 100 with respect to the 99,8% of the SVM.

Finally, instead of applying another classifier, an interface has been created to obtain new annotated sounds that could potentially make its way to the dataset. It is not only important to classify labeled data but also to be able to create quick and easily annotated data in the first place.

References

[1] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *22nd ACM International Conference on Multimedia (ACM-MM'14)*, Orlando, FL, USA, Nov. 2014, pp. 1041–1044.

[2] O. Lartillot, P. Toiviainen, and T. Eerola, "A matlab toolbox for music information retrieval," vol. 4, 01 2007, pp. 261–268.

[3] A. Bonyár, "Application of localization factor for the detection of tin oxidation with afm," 10 2015.

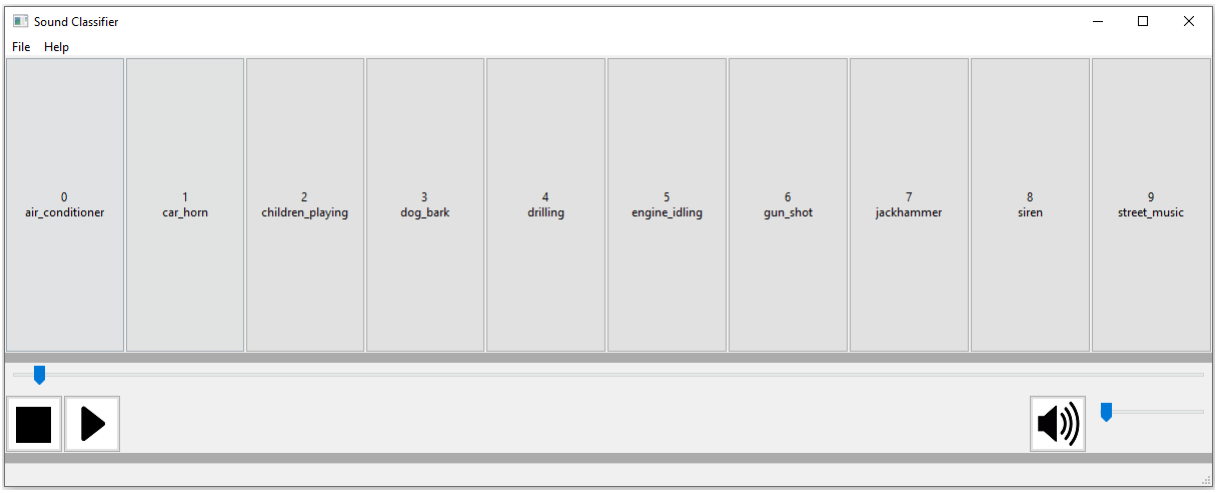

Fig. 17. Snapshot of the interface created to ease the data labelling process.