Machine Learning Approach for Urban Sound Classification

José Castanheira, *Mec. 76545;* Pedro Santos, *Mec. 76532*

***Abstract*—The cities are getting smarter each day that passes by having a lot of data collection sensors to supply information and one of them are audio sensors. This paper presents two machine learning techniques to classify sound between ten different classes of sounds. The machine learning models used were Artificial Neural Networks and K Neighbors Classifier. Firstly, it is made an overview of the work done so far in applying classification to this dataset. Then, it was tested some algorithms, and analyze the parameters that provided its best performance. It was also analyzed the impact of using preprocessing algorithms by comparing the results of using and not using them. In the end it was concluded that the best model was the K Neighbors Classifier, with an accuracy of 93.06% on the test set. This value was equally good as some of the best models used in previous work done by other researchers.**

I. Introduction

The main goal of this paper is to present two developed learning algorithms used to classify urban sounds. For each algorithm the best hyper-parameters are studied in order to achieve the best model performance. After, it is concluded which algorithm performed best.

The dataset is provided in the form of ".wav" files, each one of them with a duration of four seconds. Features are then extracted from this files in order to train the algorithms for the classification. The sounds are divided into 10 different categories: air conditioner; car horn; children playing; dog bark; drilling; engine idling; gun shot; jackhammer; siren; street music.

The models that were used were Artificial Neural Networks and K Neighbors Classifier.

II. Data Description and Visualization

The dataset compresses a total of 8732 labeled sound clips and it is retrieved from UrbanSound8K. In Table I it is possible to see the different types of sounds that are presented in the dataset, the given id and the number of sounds of each type. Figure 1 shows the same distribution in histogram form.

The features extracted from each clip are the following:

- Mel-Frequency Cepstral Coefficients (MFCC)
- Tonnetz
- Spectral Contrast
- Chromagram from a waveform
- Mel-scaled spectrogram

These features are obtained from the audio files through the use of the python library called Librosa, which contains appropriated methods to extract them.

The MFCC are coefficients that collectively make up a mel-frequency cepstrum, which 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 mel scale of frequency.[1]

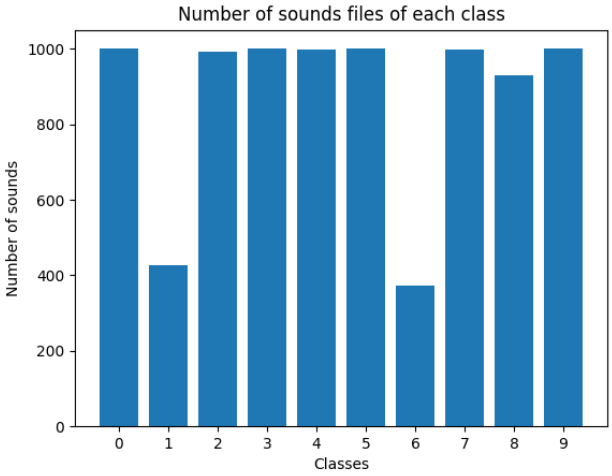

Fig. 1. Distribution of sounds for each class

	id	Number Of sounds
air_conditioner	0	1000
car_horn	1	426
children_playing	2	993
dog_bark	3	1000
drilling	4	999
engine_idling	5	1000
gun_shot	6	374
jackhammer	7	999
siren	8	929
street_music	9	1000

TABLE I
THE GIVEN IDS FOR EACH CLASS AND THE NUMBER OF SOUNDS OF EACH CLASS IN THE DATASET

The tonnetz feature, also called tonal centroid feature, is a conceptual lattice diagram representing tonal space.[2]

The Spectral contrast is defined as the decibel difference between peaks and valleys in the spectrum.[3]

The chromagram, or chroma features, are a powerful tool for analyzing music whose pitches can be meaningfully categorized (often into twelve categories) and whose tuning approximates to the equal-tempered scale.[4]

The Mel-scaled spectrogram is a perceptual scale of pitches judged by listeners to be equal in distance from one another.[11]

Using librosa it is possible to plot some of the sounds properties, for example the waveforms of the sounds. By doing this, it is much easier to understand some of the features that will be extracted and also see for each class the general

differences between the waves. Figure 2, 3 and 4 show the wave form of the sound produced by an air conditioner, a car horn, and children playing, respectively.

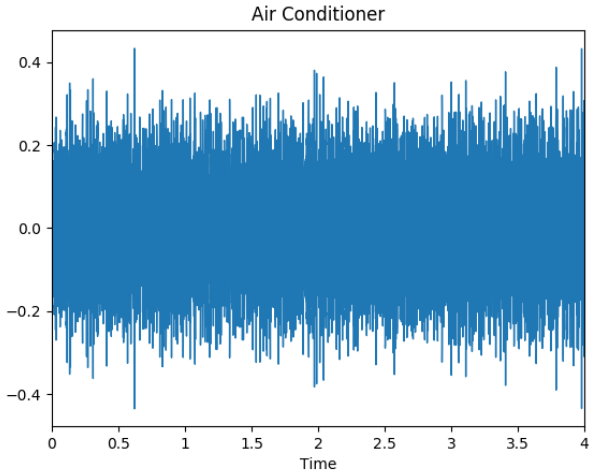

Fig. 2. Waveform of one sound of the class Air Conditioner

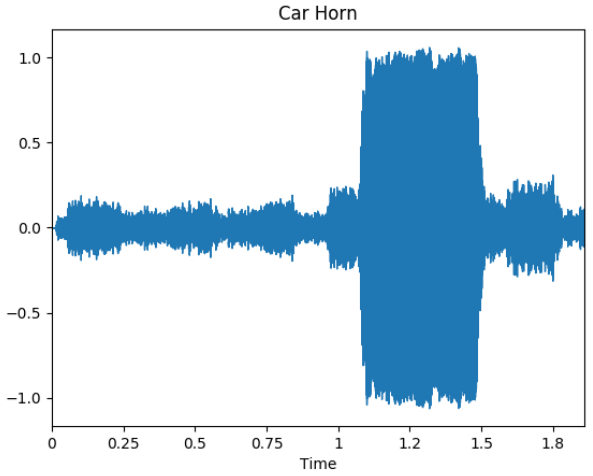

Fig. 3. Waveform of one sound of the class Car Horn

60% of the dataset was used for training while 20% was used for the cross validation. The rest, as it is obvious, was used for testing. This approach was common to all learning methods. The test set was only used to test the best model.

III. Previous Related Work

Justin Salamon, Christopher Jacoby and Juan Pablo Bello used this same dataset with the purpose of learning about the characteristics of the dataset itself [5]. With that in mind, their goal was not to find the optimal parameters of the learning algorithms used, that would yield the optimal performance. In their work, they applied 10-fold cross validation in the executed experiments. The learning methods used were: J48, a variant of ID3 (Iterative Dichotomiser 3) for decision tree induction; k-NN, with parameter k equal to 5; random forest

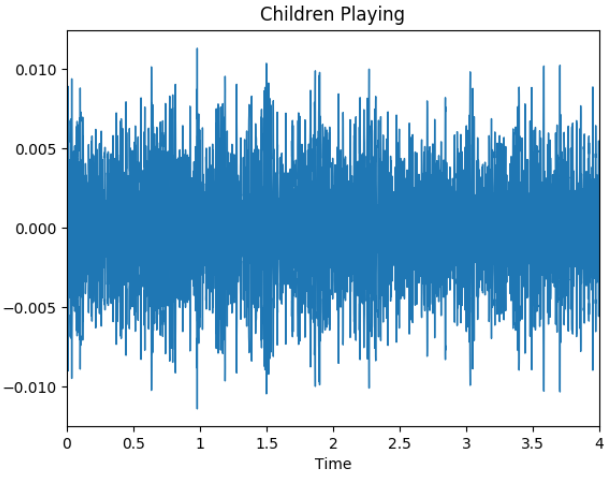

Fig. 4. Waveform of one sound of the class Children Playing

with 500 trees; support vector machine using radial basis function kernel, and a baseline majority vote classifier named ZeroR. The best performing models were the SVM and the random forest (approximately 70% accuracy), and the worst was the ZeroR (accuracy around 10%). The confusion matrix for the SVM model showed that the classifier tended to confuse 3 pairs of sounds: air conditioners and idling engines; jackhammers and drills; children and street music.

Another approach to this problem (using the same dataset) is to represent the sounds as images, and use an image classification neural network [6]. The idea is to create image representations (spectogram, MFCC and CRP) and use two types of convolutional neural networks - AlexNet and GoogLeNet - to classify those images. Convolution recurrent neural networks (CRNN) is a combination of convolutional neural networks (CNN) and recurrent neural networks (RNN) [6]. The parameters and network hyper-parameters taken into consideration were the sampling rate, frame length, frame overlap percentage, training period, base learning rate, solver type, learning rate change policy, and gamma. The optimal performance achieved was 93% with GoogLeNet and 92% with AlexNet. These values were attained using 8 kHz sampling rate, frame length of 50ms, frame over lap percentage equal to 50%, 50 epochs, learning rate 0.01, Stochastic Gradient Descent as the solver type, Exponential Decay for the learning rate change policy, and gamma equal to 0.95.

In another paper, Justin Salamon and Juan Pablo Bello explored the application of the spherical k-means for feature learning from the audio signals [8]. As they state in their work, the main difference between this algorithm and variants of the widely-used k-means clustering algorithm is that the centroids are constrained to have unit L2 norm (the benefits of this are presented in their paper). The spherical k-means algorithm purpose is to learn a codebook of representative codewords from the training data, and after encoding new samples against this codebook, use the resulting code vector for training and testing some classifier. In their work, the classifier used was a random forest. Again, as in their paper,

mentioned in the first paragraph, their objective is not to attain maximum performance, but to focus on the feature learning stage. They tested the model with three different values of k (500, 1000, 2000). Generally, the model performed better using k equal to 2000 (around 75% accuracy).

In [7], Salamon and Bello also applied other deep learning models for classification using the present dataset. They purposed a deep convolutional neural network composed by three convolutional layers interleaved with two pooling operations, followed by two fully connected (dense) layers. Data augmentation techniques, such as time stretching (slow down or speed up the audio samples), pitch shifting (raise or lower the pitch of the audio sample), were also tested in conjunction with the neural network. The performance of the model was about 73% without augmentation techniques on the dataset. The original dataset is not large/varied enough for the convolutional model to outperform the "shallow" SKM approach (k equal to 2000) [7]. However, when used in conjunction with data augmentation, the model performance improved until 79%, while the SKM didn't yield a significant change in performance. Increasing the capacity of the SKM model (by increasing k from 2000 to 4000) didn't also yield any improvement in classification accuracy. This indicates that the superior performance of the proposed CNN is not only due to the augmented training set, but rather thanks to the combination of an augmented training set with the increased capacity and representational power of the deep learning model [7].

Salamon and Bello tested the scattering transform as an alternative to the mel-spectogram [9]. The scattering transform can be viewed as an extension of the mel-spectrogram that computes modulation spectrum coefficients of multiple orders through cascades of wavelet convolutions and modulus operators [9]. Again, the model used was the spherical k-means. They proved that slightly better results could be achieved using this technique, while at the same time reducing the amount of training data.

IV. Handle the 8732 sounds

The amount of time that takes to extract the features of the 8732 sound clips is around 25 minutes in a CPU i7-4710HQ. In that conditions it would be very hard to test different algorithms and parameters. The solution adopted was to serialize all the data. By doing this it is only necessary to extract the features one time (and serialize the data) and then just use that same data in each run by deserializing it.

V. Preprocessing the Data

Before training the models the data was shuffle to ensure that the training set, cross validation set and test set are representative of the overall distribution of the data. Another technique of preprocessing that was used was StandardScaler, however before using this technique the algorithms were tested first without it, to be possible to compare the results between applying and not applying preprocessing. This class belongs to the sklearn preprocessing package.

More information about the StandardScaler can be found in ⟨http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html⟩ This algorithm will transform the data in such way that its distribution will have a mean value 0 and standard deviation of 1.

VI. Neural Network

An Artificial Neural Network is a model inspired on the human brain, composed of several layers, each with a variable number of hidden nodes. In this section, different neural network architectures were tested to see how well the model behaves. In all experiments, the learning rate used was equal to 1e-5 and the solver used was the LBFGS (except when stated otherwise).

A. One hidden Layer

Firstly, the performance on a single hidden layer network was tested. The number of neurons used was: 25, 50, 100, 150, 200, 500, 1000. After 1000 hidden nodes, the computation time was considerably big. Figure 5 shows graphically the performance of the neural network. The best performance was attained when using 1000 hidden nodes, with an accuracy of 77.18%.

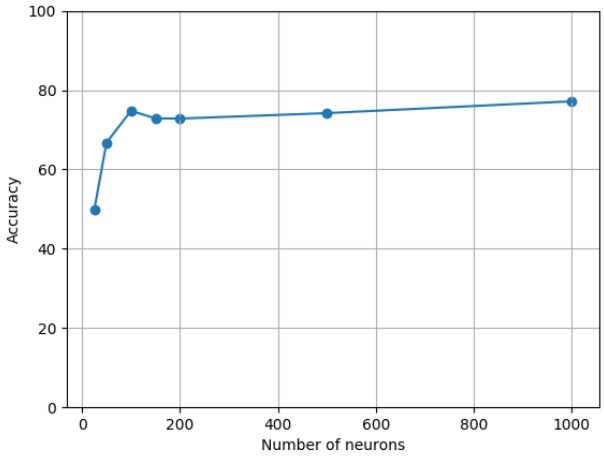

Fig. 5. Performance of a one hidden layer Artificial Neural Network as a function of the number of hidden nodes.

Table II shows the confusion matrix for the neural network with 1000 hidden nodes.

B. Influence of the learning rate

From the previous subsection, one can conclude that 1000 hidden nodes provide the optimal performance. From this point on, the influence of the learning rate was studied. In order to do so, we tested the model with 1000 hidden nodes for the following values of the learning rate: 1e-7, 1e-6, 1e-5, 1-4, 1e-3, 1e-2, 1e-1. Table III shows the different values of accuracy attained, as a function of the different learning rates. We can conclude that the learning rate introduces small variations in the performance of the model - the difference between the best and worst performance for the different learning rates is about 0.84%.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	176	0	4	0	2	2	0	6	2	6
1	4	69	1	3	4	0	0	0	2	6
2	12	0	133	18	7	5	2	3	5	32
3	5	2	16	146	6	3	2	1	10	12
4	1	6	6	8	132	4	0	7	0	9
5	15	0	6	0	3	166	1	0	2	4
6	0	0	4	9	0	0	65	0	1	3
7	2	0	0	1	12	1	0	154	1	1
8	8	1	12	6	0	6	1	1	171	2
9	8	4	25	6	13	4	0	10	0	134

TABLE II
ANN WITH 1000 HIDDEN NODES CONFUSION MATRIX.

	Accuracy
1e-7	76.38
1e-6	76.55
1e-5	76.61
1e-4	77.19
1e-3	76.95
1e-2	76.83
1e-1	76.34

TABLE III
NEURAL NETWORK ACCURACY FOR DIFFERENT VALUES OF LEARNING RATE.

C. Two Hidden Layers

The second step was to test a neural network with more than one hidden layer. A two hidden layer neural network was used, with the number of hidden nodes in each layer varying between 50, 100, 150, 200, and 500. Figure 6 shows a scatter plot of the performance of neural network for each value of its hyper-parameters. The optimal value of accuracy was found at 100 hidden nodes on the first hidden layer, and 150 on the second one, with an accuracy of 82.74%.

It is important to note that already at this stage better results are being achieved than the ones attained at some of the previous work done before (see section III).

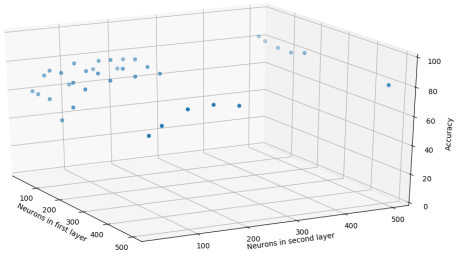

Fig. 6. Performance of a two hidden layer Artificial Neural Network as a function of the number of hidden nodes.

D. Multiple Layers

Using two hidden layers offered a clear advantage over the version with just one hidden layer. At this stage, the aim is to identify exactly how the increase in the number of hidden layers affects the performance of the model. The number of hidden nodes was set to 100 (to avoid long computations). The model was tested from 1 hidden layer to 10 hidden layers. In figure 7 it is visible how the model operates while varying the number of hidden layers. With 4 hidden layers, the model attained the maximum performance of 80.73% accuracy, while with the maximum number of hidden layers - 10, in this case - it had the worst performance, of 72.36% accuracy.

With this subsection, it is reinforced the importance of the number of hidden layers when using neural networks to apply classification on this dataset.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	194	0	1	1	0	0	0	1	1	0
1	0	79	0	2	3	1	0	0	0	4
2	4	0	186	7	2	1	1	1	1	14
3	4	1	8	166	9	0	1	0	8	6
4	0	2	0	4	159	1	0	6	0	1
5	2	0	0	0	1	192	0	0	0	2
6	2	0	0	2	0	1	77	0	0	0
7	0	0	0	3	6	0	1	163	0	0
8	3	0	5	4	0	0	0	3	190	3
9	1	3	12	4	3	1	0	5	1	173

TABLE IV
NEURAL NETWORK (WITH PREPROCESSED DATA) CONFUSION MATRIX.

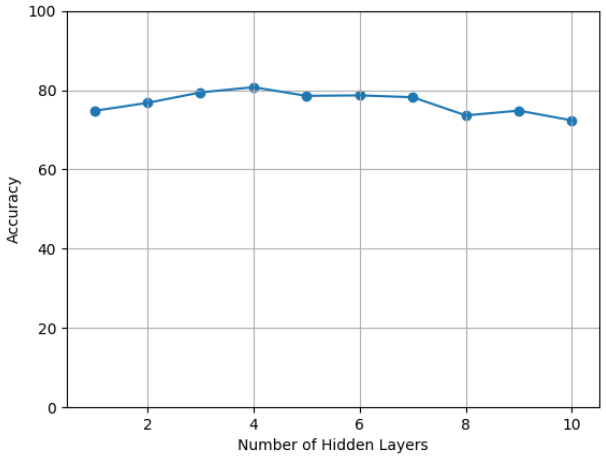

Fig. 7. Performance of Artificial Neural Network as a function of the number of hidden layers. Number of hidden nodes in each layer is equal to 100.

E. Applying preprocessing

When applying the preprocessing mentioned in section V, significant improvements were detected. Repeating subsection A (one hidden layer) yield the performance visible in figure 8. Although the number of nodes required to achieve the best performance is still the same (1000), the overall accuracy attained was clearly better, with a value of 90.54%. Table IV shows the confusion matrix of this model.

However, it is important to note that with 500 nodes the accuracy achieved was 90.31%, practically the same as the one with 1000 nodes. In this case we can conclude that the best model is the one that uses 500 nodes, as it lowers the computations required without yielding any significant performance drop.

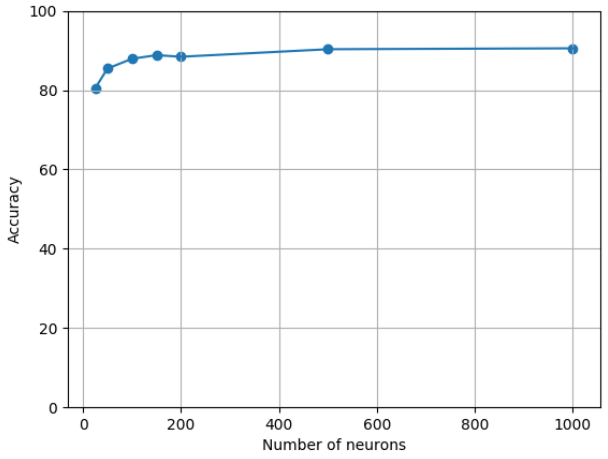

Fig. 8. Performance of Artificial Neural Network with one hidden layer. Uses preprocessed dataset.

The effect of the learning rate when using preprocessed data didn't increase significantly relatively to the one seen before. The best value attained was with learning rate equal to 1e-1, with an accuracy of 91.51%. The worst was with learning rate equal to 1e-3 (instead of 1e-4, as previous), with accuracy 90.37%. We conclude that preprocessing the data affects the optimal learning rate.

The experiment of using a two layer neural network with a variable number of hidden nodes was performed again. Figure 9 shows the performance of the model using the various architectures. Now, the best architecture was the one with 500 nodes on the first layer and 200 on the second one. The accuracy achieved was 89.97%. One important thing to note here is that, previously, the 2 layer neural network performed better than the network with just one hidden layer. Here, the roles are inverted.

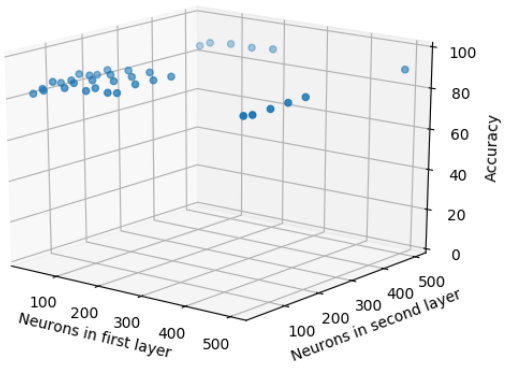

Fig. 9. Performance of Artificial Neural Network with two hidden layers. Uses preprocessed dataset.

Lastly, it was evaluated the model behavior as a function of the number of hidden layers (same experience as in sub subsection D). Figure 10 shows the evolution of accuracy with the increase in number of hidden layers. Optimal performance is still achieved with 4 hidden layers - 88.59%.

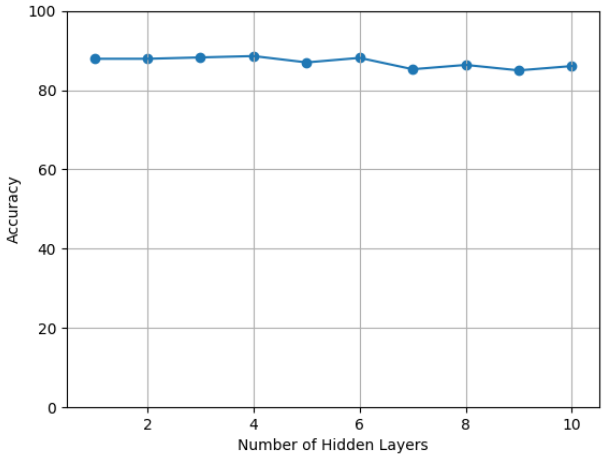

Fig. 10. Performance of Artificial Neural Network as a function of the number of hidden layers. Number of hidden nodes in each layer is equal to 100. Uses preprocessed dataset.

VII. K Neighbors Classifier

K Neighbors Classifiers is a simple algorithm to understand and for that reason it is interesting to try to use it and see how good are the results . If it is reasonably good with the default parameters and if it is an easy algorithm to understand how it can be manipulated to improve the results, it will generate in the end at least a good result without spending to much time.

K Neighbors Classifiers consists in plotting the data on an n-dimensional space where 'n' is the number of data-attributes and in this specific case, n is equal to 10. Then, to calculate and predict the label of an unclassified data, the point is plotted on this n-dimensional space and the labels of nearest 'k' data points are noted. The class that occurs more times among the k nearest data points is considered as the class of the new data-point.

A. Running the algorithm with default values of library

The first approach was to test the algorithm using the default values of the library sklearn. The default values of the parameters are presented in the next list. The explication of each parameter can be found at ⟨http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier⟩

- n_neighbors=5
- leaf_size=30
- weights='uniform'
- p=2
- metric='minkowski'
- n_jobs=1
- algorithm='auto'
- metric_params=None

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	186	1	3	0	0	3	2	2	0	1
1	10	50	7	4	3	6	1	6	2	0
2	11	1	171	4	2	5	1	4	7	11
3	6	2	25	129	6	2	3	6	8	16
4	5	0	6	0	143	0	1	11	2	5
5	1	0	11	2	0	179	2	1	0	1
6	2	0	7	8	0	3	59	1	0	2
7	4	0	0	0	2	0	0	167	0	0
8	11	0	12	7	1	4	0	2	166	5
9	12	4	31	7	7	7	2	7	3	124

TABLE V
K NEAREST NEIGHBOR WITH DEFAULT PARAMETERS CONFUSION MATRIX.

The accuracy obtained by running the algorithm without changing the parameters was 78.78%. The confusion matrix is presented in the next table V.

Using the confusion matrix it was developed a function to calculate the miss predictions of each class. The result is presented in the histogram in the figure 11.

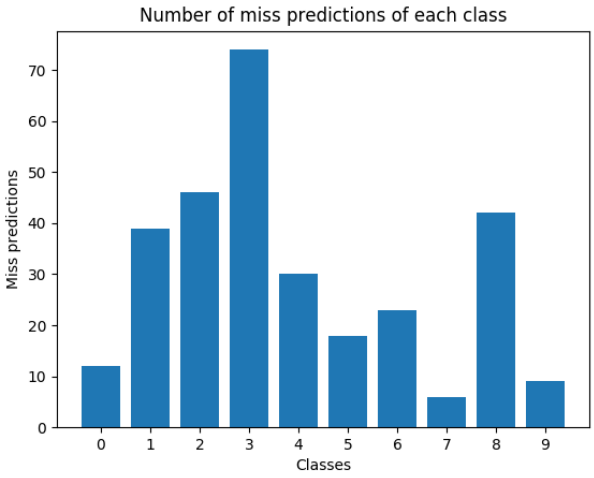

Fig. 11. Miss predictions with default parameters

Analyzing the results, it is possible to see that the class with more miss predictions is the class number 3 that corresponds to the sound of dog bark, with 74 miss predictions. On the other hand the class which has the less miss predictions is the class number 7 that corresponds to the sound of the jackhammer, with 6 miss predictions.

B. Finding the best parameters

Now, after analyzing the results with the default parameters, the next step is to try find the parameters that will generate the optimal accuracy.

After analyzing the parameters it was concluded that the only parameters that can change the accuracy is the number of neighbors and p. The other parameters will only influence the memory used and the speed. So, first of all it was decided to only manipulate the number of neighbors. The number of neighbors that were tested was [1,2,3,4,5,6,7,8,9,10,11] and the results are in the figure **??**.

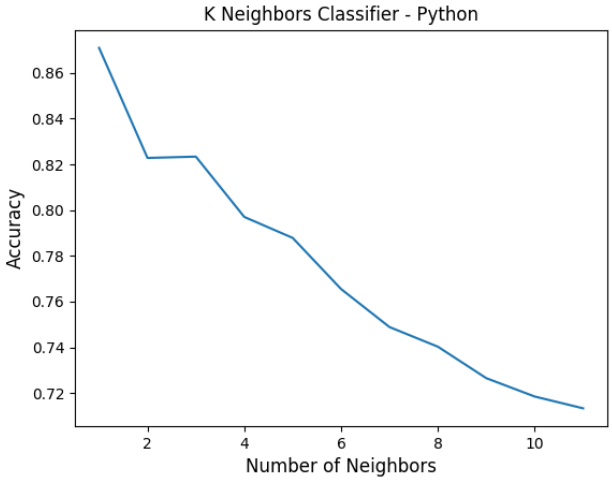

Fig. 12. Accuracy for each number of neighbors

Neighbor	p	accuracy
1	1	91.57
2	1	87.61
3	1	87.32
4	1	85.26
5	1	83.08

TABLE VI
THE BEST P AND THE THE CORRESPONDING ACCURACY FOR EACH VALUE OF NEIGHBOR

It is possible to conclude by analyzing figure 12 that the accuracy decreases as the number of neighbors increases, so the best accuracy is obtained by using only one neighbor. Using k = 1, the accuracy obtained is 87.09%.

It is now time to manipulate the parameter p. The parameter p can have 3 different values: 1, 2, or another arbitrary number. When p is equal to 1 the manhattan distance will be used to calculate the nearest neighbor, when p is equal to 2 the euclidean distance will be used, and when it is another value the minkowski distance will be used. In the following list it is presented the results by running the algorithm with the different possible values for p, and with k equal to 1. By changing p, it was concluded that the best p was 1 (manhattan distance) which corresponds to an accuracy of 91.57 %.

- manhattan_distance = 91.57 %
- euclidean_distance = 87.09 %
- minkowski_distance = 84.00 %

The last attempt to get a better accuracy was to combine the different possible values of p and k. To accomplish that goal, all combinations with k between 1 and 5 and the different values of p were tested.

- neighbors=[1,2,3,4,5]
- p=[1,2,3]

To analyze the results it was calculate for each value of neighbors the best accuracy and which p was used to get that accuracy. The results of this test are presented in the table VI

By analyzing the results of the table VI it is concluded that independently of the value used for the number of neighbors the best value for p is 1 (manhattan distance). Also, it is

Neighbors	p	accuracy
1	1	91.57 %
2	1	87.61 %
3	1	87.32 %
4	1	85.26 %
5	1	83.08 %
1	2	87.10 %
2	2	82.28 %
3	2	82.34 %
4	2	79.70 %
5	2	78.78 %
1	3	84.00 %
2	3	79.47 %
3	3	79.24 %
4	3	77.24 %
5	3	75.00 %

TABLE VII
ALL THE RESULTS BETWEEN NEIGHBORS AND P

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	193	1	2	0	0	2	2	2	0	1
1	3	73	1	1	3	2	0	2	1	3
2	2	0	199	4	2	5	1	4	7	11
3	4	0	13	168	2	1	4	1	7	3
4	3	0	1	0	161	0	0	5	1	2
5	0	0	2	0	0	195	0	0	0	0
6	0	0	2	3	0	0	74	1	0	2
7	0	0	0	0	3	0	0	169	0	1
8	1	0	2	3	0	3	0	0	195	4
9	3	1	11	4	7	1	1	4	2	170

TABLE VIII
K NEAREST NEIGHBOR CONFUSION MATRIX USING THE BEST PARAMETERS.

concluded that by changing p to 1 the increase of the number of neighbors will continue to reduce the accuracy like it was concluded before for the default parameter that corresponds to running the algorithm with p equal to 2 (euclidean distance).

Analyzing the table VII it is possible to conclude:

- that no matter the used p, with the increase of neighbors the accuracy diminish.
- for the same number of neighbors, by setting p to 1, better accuracy is always attained.
- for the same p, with the increase of neighbors the accuracy gets lower (there is only one case that this doesn't happen {p=2,neighbor=2,3})

C. Best parameters

The conclusion is that the parameters that will produce the optimal accuracy is p equal to 1 and k equal to 1. By using this parameters, an accuracy of 91.57% was achieved. In the table VIII is presented the confusion matrix using the best parameters.

In figure 13 it is represented the sum of miss predictions in each class using the best parameters and it is interesting to compare with figure 11 that is the miss predictions with default parameters. It is possible to see that the proportion between the misses of each class is very similar, but it was possible to reduce considerable the number of miss predictions of every class. To have a better idea of how much the miss predictions diminish while using the best parameters, note that when using the default values the sum of the miss predictions

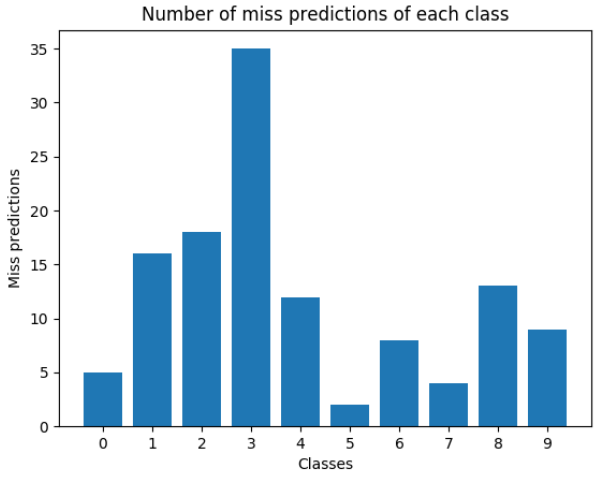

Fig. 13. Miss predictions of best parameters

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	195	0	0	3	0	0	0	0	0	1
1	1	73	5	2	2	0	1	0	0	5
2	1	0	202	6	0	0	0	0	2	6
3	0	0	19	168	1	1	2	0	8	4
4	0	2	4	0	157	0	1	5	1	3
5	0	0	0	0	1	195	0	0	1	0
6	0	0	2	1	0	0	77	0	0	2
7	0	0	1	0	4	0	0	168	0	0
8	0	0	4	2	0	0	0	0	201	1
9	0	0	19	1	0	0	2	2	0	180

TABLE IX
K NEAREST NEIGHBOR CONFUSION MATRIX USING THE BEST PARAMETERS AND PREPROCESSING.

is 299 and with the best parameters it is only 122, being the total of elements 1744.

D. Using Preprocessing

All the tests performed before were done without any preprocessing other than shuffling the data. Now, the purpose it to try to improve the accuracy by doing some more complex data preprocessing.

E. Using StandardScaler in the best solution

The first thing that was tested was to train the model with the best parameters found in the last section, however doing before the preprocessing. The accuracy obtained was 92.66%. The accuracy with the preprocessing increased by 1.09% in relation to our previous best accuracy. Table IX presents the confusion matrix by using preprocessing.

The histogram in figure 14 shows the miss predictions with preprocessing and like expected it is very similar with the histogram in figure 13. The same happens with the confusion matrix in the table IX.

F. Testing if the best parameters changes with the preprocessing

In this subsection the experiments consists in calculating for each value of neighbors the best accuracy and which

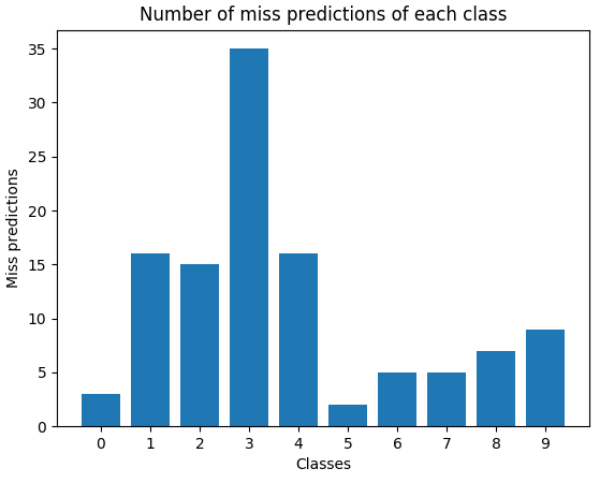

Fig. 14. Miss predictions of best parameters and with preprocessing

Neighbors	p	accuracy
1	1	92.66 %
2	1	88.36 %
3	1	88.47 %
4	1	86.00 %
5	1	85.21 %
1	2	92.60 %
2	2	87.61 %
3	2	88.02 %
4	2	85.37 %
5	2	85.09 %
1	3	91.91 %
2	3	86.35 %
3	3	86.70 %
4	3	84.63 %
5	3	84.40 %

TABLE X
ALL THE RESULTS BETWEEN NEIGHBORS AND P USING PREPROCESSING

p was used to get that accuracy. Again, before this, data preprocessing will be applied and the main goal is to see if the best parameters change with the preprocessing.

It is concluded that the best parameters using the preprocessing doesn't change, so the parameters that will generate a better accuracy are still p equal to 1 and also the number of neighbors equal to 1. Also, it is concluded that using this preprocessing the parameter p is less important because, when comparing the accuracies that have the same number of neighbors but different p, the difference between them is very small. That makes sense because, as the preprocessing is standardizing the data, the distance used to see the closest neighbor will be less important. Another important point is that the increase of the accuracy when not using the optimal parameters (table VII) is much higher than in the case of when using the best parameters. To show that it is a good idea to see the difference between the worst and best results in each case. Without preprocessing the difference between the best accuracy and the worst is 16.57 %, while with preprocessing the difference is 8.26 %.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	206	0	0	3	0	0	1	1	0	0
1	0	68	11	2	0	0	0	1	0	10
2	0	0	192	0	0	0	0	0	3	3
3	0	0	24	157	0	1	1	0	7	3
4	0	0	4	1	190	0	0	1	0	1
5	1	0	2	0	0	198	0	0	0	1
6	0	0	4	0	0	0	70	0	0	1
7	0	0	0	0	7	0	0	187	0	2
8	0	0	3	0	0	0	0	0	168	1
9	1	0	15	2	0	0	0	0	3	187

TABLE XI
K NEAREST NEIGHBOR CONFUSION MATRIX USING THE TEST SET.

Label	**Precision**	**Recall**	**FScore**	**Support**
0	99.04	97.63	98.32	211
1	100	73.91	85.00	92
2	75.29	96.97	84.76	198
3	95.15	81.35	87.70	193
4	96.45	96.45	96.45	197
5	99.50	98.02	98.75	202
6	97.22	93.33	95.24	75
7	98.42	95.41	96.89	196
8	92.81	97.68	95.18	172
9	89.47	89.90	89.69	208

TABLE XII
K NEAREST NEIGHBOR PRECISION, RECALL, FSCORE AND SUPPORT .

VIII. MODEL SELECTION

After testing the two algorithms and analyze the results it was concluded that the algorithm that can produce higher accuracy is K Neighbors Classifier.

The next step is using the test set. The accuracy obtained with the test set was 93.06%. The confusion matrix is presented in the table XI.

In the table XII is presented for each class the precision, recall, fscore and support values. This values were calculated using the functions of sklearn.

The Fscore is defined as the harmonic mean of precision and recall. The three class with higher fscore are:

- id=5, label=engine_idling - 98.75 %
- id=0, label=air_conditioner - 98.32 %
- id=7, label=jackhammer - 96.89 %

The three class with lower fscore are:

- id=2, label=children_playing - 84.76 %
- id=1, label=car_horn - 85.00 %
- id=3, label=dock_bark - 87.70 %

IX. CONCLUSION

The results achieved, both with Artificial Neural Networks and K Neighbors Classifier proved to be better using the standard scaller, which means that the dataset can be made better than it originally is. The neural networks, although being a more complex model, were slower to train and failed to work as well as the K Neighbors Classifier. Overall the results were good, even when compared to the work done before by other researchers. Note in particular than the performance for the K Neighbors Classifier was as good as the one obtained when using GoogLeNet, a form of deep learning.

Other important aspect to note is using the StandardScaler preprocessing in K Neighbors Classifier it will reduce the

impact of the different possible parameters that can be used to train the model and also improve the accuracy of the model and in a even more accentuated way if the parameters are not the best ones. Regarding the conjunction of StandardScaler and Artificial Neural Networks, what is observed is that StandardScaler changes the architecture that provides the best accuracy.

X. Work Division

José Castanheira: Classification Models, report writing.
Pedro Santos: Classification Models, report writing.

References

[1] Mel-frequency Cepstrum. https://en.wikipedia.org/wiki/Mel-frequency_cepstrum.
[2] Tonnetz. https://en.wikipedia.org/wiki/Tonnetz.
[3] Jun Yang, Fa-Long Luo, Arye Nehorai. *Spectral contrast enhancement: Algorithms and comparisons* In Speech Communication 39, 2003.
[4] Chroma Feature. https://en.wikipedia.org/wiki/Chroma_feature.
[5] Justin Salamon, Christopher Jacoby, Juan Pablo Bello. *A Dataset and Taxonomy for Urban Sound Research*.
[6] Venkatesh Boddapatia, Andrej Petefb, Jim Rasmussonb, Lars Lundberga. *Classifying environmental sounds using image recognition networks*. In International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2017, 6-8 September 2017, Marseille, France
[7] Justin Salamon and Juan Pablo Bello. *Deep Convulutional Neural Networks and Data Augmentation for Environmental Sound Classification*. In IEEE Signal Processing Letters, Vol. 24, No. 3, March 2017
[8] Justin Salamon, Juan Pablo Bello. *Unsupervised Feature Learning For Urban Sound Classification*. In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)
[9] Justin Salamon, Juan Pablo Bello. *Feature Learning with Deep Scattering For Urban Sound Analysis*. In 2015 23rd European Signal Processing Conference (EUSIPCO)
[10] Tutorial Urban Sound Classification http://aqibsaeed.github.io/2016-09-03-urban-sound-classification-part-1/.
[11] Mel-scale spectrogram. https://en.wikipedia.org/wiki/Mel_scale.