Data Science Lab: Process and methods
Politecnico di Torino

Project report
Student ID: 279123

1 Data exploration (max. 400 words)

This project is about the analysis of different audio files in order to compute the "speaker recognition task", that is about reading the file and select who's speaking from a set of possible choices. For the analysis, I have used *Librosa* n.d., that is a Python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems.

1.1 Data set structure

The data set is provided by *Dataset* n.d., a free web site that provides audio-books. It's cardinality is 30.281 elements, divided in two splits: 24.449 in the development set, that has been used as training set; 5.832 in the evaluation set.

Elements belong to 10 different classes, each one represent a different speaker. The dataset is well balanced: the figure 1 shows that the mean of elements for classes is around 2500, with a very tiny deviation.

Looking at the data description, we expect to have files 0.5sec long. A further analysis returned that all the elements but 7 have a length greater or equal than 0.45 second. Having in mind the final goal, it's clear that shortest elements can be considered "errors" and they can be dropped out from the dataset.

1.2 Outliers detection

By using the "describe" method available for Pandas Dataframes, it is possible to check the distribution of numerical attributes by figure out interesting statistic like mean, standard deviation and percentiles. Because features are been obtained by many manipulations, it's difficult to make an analysis of their distribution in the space: while with other kind of features (e.g. age of a person etc,) it is possible to identify if records have strange values, in this case it is not possible. In order to identify outliers we need a more specific method.

So by using Librosa features, files are converted in digital representation and some features has been extracted; specifically, I've extracted the following: *Mel-frequency cepstrum* n.d. (first 40), Chromogram (12 elements), Spectral centroids, Spectral bandwith, Zero crossing *Feature extraction* n.d.

Looking at the features, it's easy to see that there are different files for which all the features had zero as value. That elements are distributed among different classes and they don't seem to have

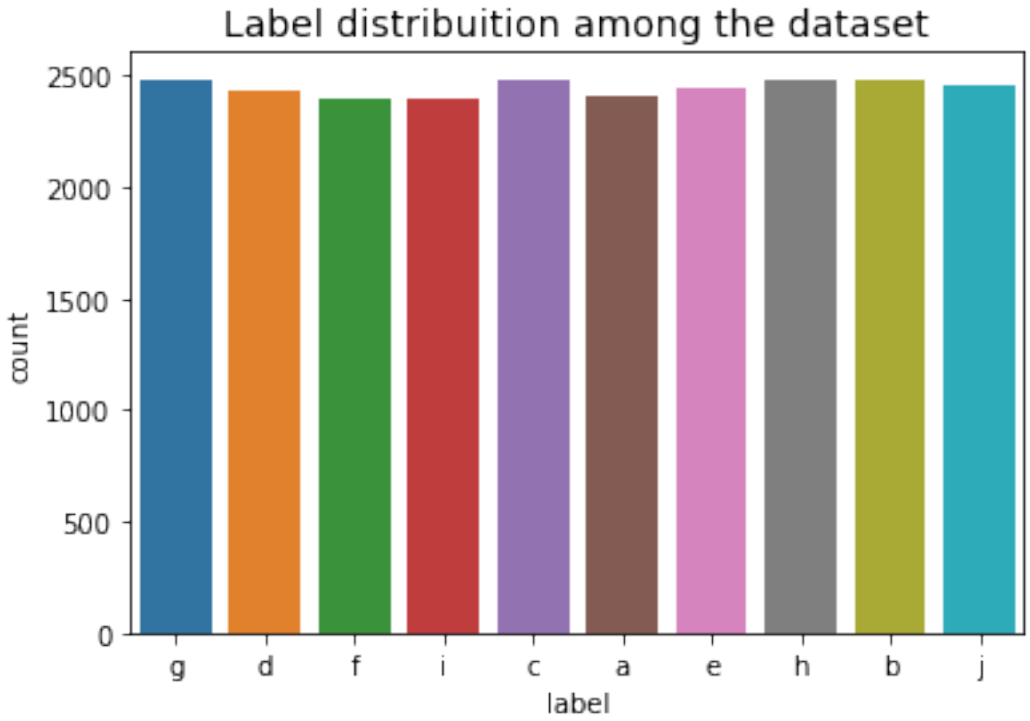


Figure 1: The elements are uniformly distributed among classes

a regular pattern, so I decided to drop out them because I noticed that there are audio files in the training dataset that are only noise..

At this point, the dataset is composed of 55 features; in order to identify all the outliers, I have tried different approaches, by using standard methods provided by Sklearn:

- **IsolationForest**: it is a tree-based anomaly detection algorithm
- **LocalOutlierFactor**: tries to identify outliers by locate those examples that are far from the other examples in the feature space
- **EllipticEnvelope**: starting from normally distributed data, it applies a statistical analysis to detect outliers.

Best performances are provided by *LocalOutlierFactor*, so all the further analysis are done by using this method.

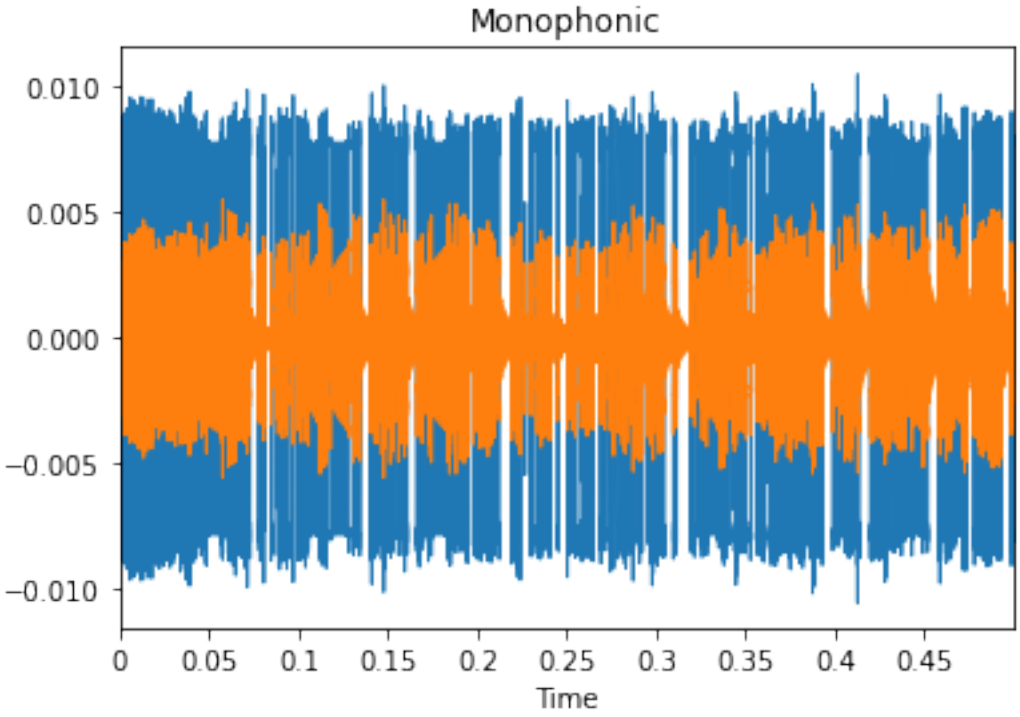


Figure 2: Frequencies audio wave plot before and after noise reduction

2 Preprocessing (max. 400 words)

The preprocessing phase is dived into three parts:

2.1 Noise Reduction

Audio collected in different and non-ideal conditions can have a strong noise component that, by affecting the features-extraction phase, can confuse the classification algorithm.

The noise reduction goal is achieved by creating a band-pass filter that allows to exclude from the audio all the frequencies that aren't in a pre-defined range. In order to define that range, I have used the maximum and the minimum value, into the dataset, of the spectral-centroid features. It is calculated as the weighted mean of the frequencies present in the signal, determined using a Fourier transform, with their magnitudes as the weights. So, the idea is to find the most dense range of frequencies available in the signal and keep only the data that are inside this range. It is an element-wise method and each file is analysed without considering others. Another approach could be to find a global range for the entire dataset and create the bandwidth using this, but due to the fact that the elements were collected under different conditions, it seems more correct to do an analysis separate. The Figure 2 shows how this band-pass filter works: it's easy to see that upper and lower frequencies has been cleaned by the method.

2.2 Standardization

An important step before applying dimensionality reduction is to standardize features, so all the features are in the same space and it has sens to calculate distances and variance among elements.

The standard score of a sample x is calculated as:

$$z = (x - u)/s \tag{1}$$

where x is the original data, u is the mean of feature in the entire dataset and s is the correspondent standard deviation.

2.3 Dimensionality reduction

The dataset so is composed of 57 columns. But for each algorithms that uses the concept of distance for the classification task, the dimensionality is a big problem. In order to solve this issue, I have applied two different approaches:

- **Features Selection**: it is a technique where we choose those features in our dataset that contribute most to the target variable. In this project, it computes the score based on the ANOVA test, a statistical method that, for each feature, compares the linear model with and without the named feature to define how much it affects the classification task. It the feature is considered "important", it is added to the model; else, it is discarded. This method allows me to take all the features available in Librosa and to decide which I want to keep in a deterministic way. Otherwise, I'd had to choose a priori which feature will be representative and which won't. This step allows the dataset to be reduced from 57 to 45 columns

- **PCA**: Principal Components Analysis is a technique of dimensionality reduction, based on the idea of preserve the highest variance among the data but reducing the number of features. Each PC is an orthogonal linear transformation of the original data. Figure 3 shows how the variance is distributed among all the principal components in the dataset. It's cleat that the first 30 PC perfectly fit all the variance in the dataset, so we can truncate the PC matrix at this point.

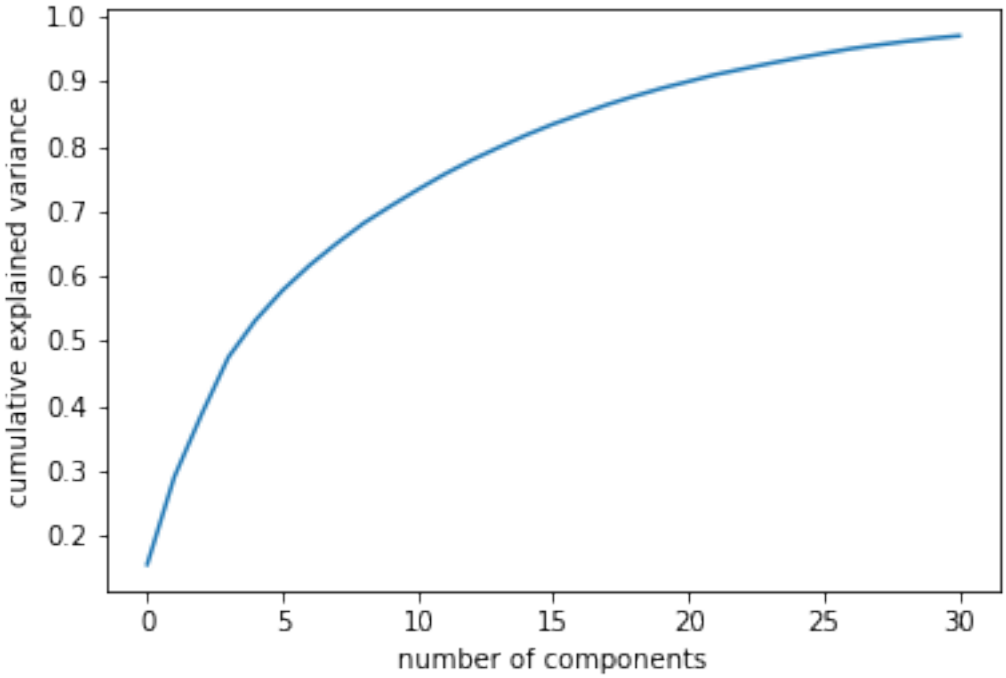


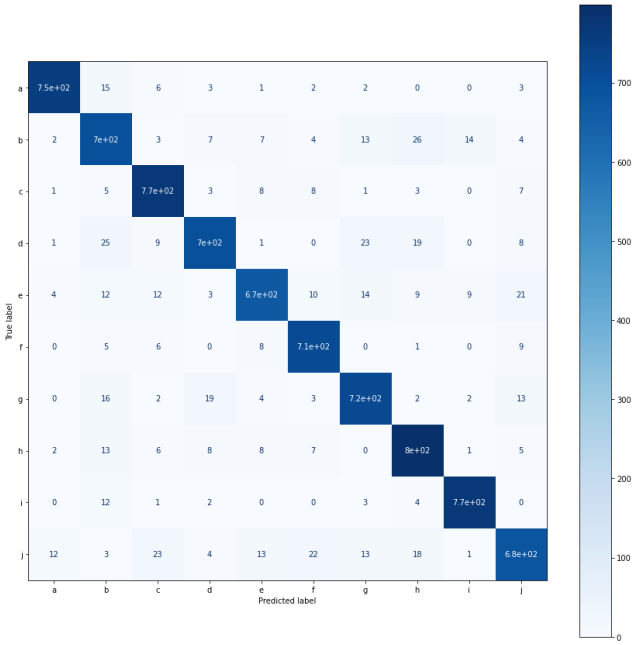
Figure 3: Variance among different features

3 Algorithm choice (max. 400 words)

In order to create a learning algorithm able to classify instances, we need to create a training set S , sampled from the entire dataset D , and to train our model trying to minimize the training error. This approach is based on the Empirical Risk Minimization paradigm. Then, let's analyse different algorithms:

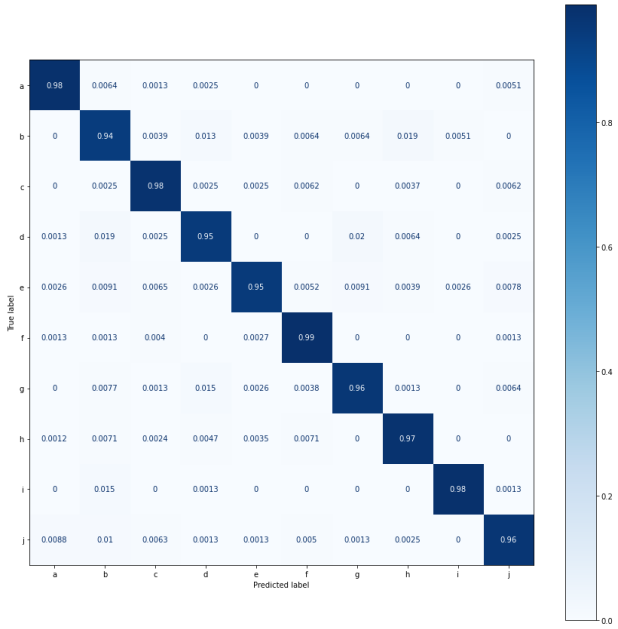
- **SVM**: it finds the optimal hyperplane between the points of two classes such that the distance of the nearest points to the decision boundary is maximized.
- **Random Forest**:it is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees.
- **KNN**: the core idea for KNN is that the features that are used to describe the domain points determines their classification label. So, elements from the same classes need to have similar features.
- **Logistic Regression**: it is a method that is based on the "logistic function", which maps from R to $[0, 1]$, where the final value is seen as the probability for each element to belong to a specific class.
- **Naive Bayes**: it computes, by the Bayes rules, for each element and each class, the probability that the element belongs to the class. Highest probability defines the assigned class.

In Figure 4 we can see the confusion matrix for all the mentioned algorithms: the best performances are given by the Support Vector Machine method.

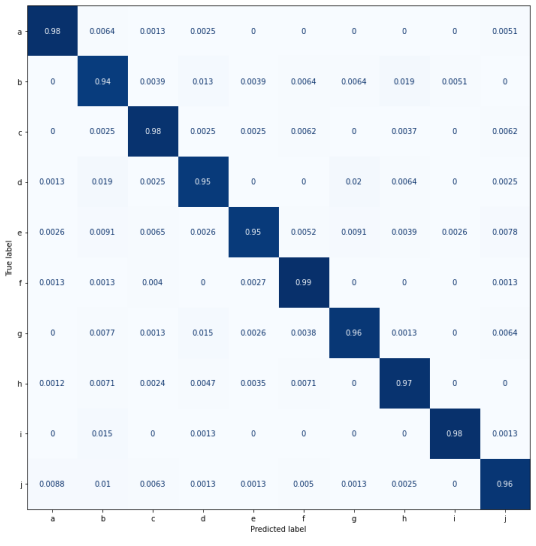


Random Forest

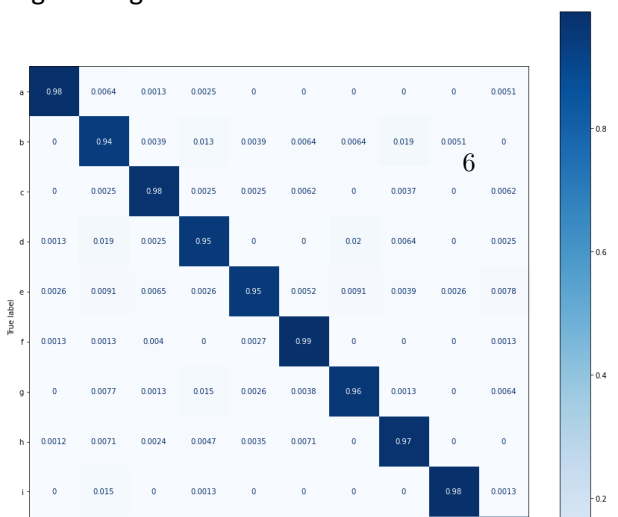
SVM



Logistic Regression



KNN



	C	gamma	kernel	Accuracy
0	0.1	0.01	rbf	0.918841
1	0.1	0.01	poly	0.515432
2	0.1	0.01	linear	0.930902
3	0.1	0.10	rbf	0.802127
4	0.1	0.10	poly	0.959379
5	0.1	0.10	linear	0.930902
6	1.0	0.01	rbf	0.954060
7	1.0	0.01	poly	0.855982
8	1.0	0.01	linear	0.928305
9	1.0	0.10	rbf	0.949831
10	1.0	0.10	poly	0.953599
11	1.0	0.10	linear	0.928305
12	10.0	0.01	rbf	0.964530
13	10.0	0.01	poly	0.944261
14	10.0	0.01	linear	0.926588
15	10.0	0.10	rbf	0.951799
16	10.0	0.10	poly	0.951589
17	10.0	0.10	linear	0.926588
18	100.0	0.01	rbf	0.962268
19	100.0	0.01	poly	0.959420
20	100.0	0.01	linear	0.925625
21	100.0	0.10	rbf	0.951799
22	100.0	0.10	poly	0.951589
23	100.0	0.10	linear	0.925625

Figure 5: Tuning phase results

4 Tuning and validation (max. 400 words)

After having selected the method, it is important to define the best combination of hyper-parameters in order to achieve best results. A good approach, in this case, can be to use a K_Fold Cross-Validation model: the original dataset is divided in k partitions having $\frac{n}{k}$ elements, where n is the size of the original dataset. At each step, the i^{th} partition is used as validation set and the others $k-1$ are concatenated and used as training set. The model is trained on the training set, and validate on the validation set and the final score for the model is the average mean squared error: $CV_{(k)} = \frac{1}{k} \cdot \sum_{i=1}^{k} MSE_i$, where $MSE_i = |Y_i - \hat{Y}_i|^2$ being $\hat{Y}_i$ the predictor of Y_i using all the other observations.

The k-fold procedure is done many times and, at each iteration, the model uses a different set of parameters from the grid and, finally, the model providing the best score is selected as the best one and the entire dataset is re-trained by using those hyperparameters. In the categorical case, as the one analyzed here, the error is computed as: $CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} Err_i$, where $Err_i = I(y_i \neq \hat{y}_i)$

The Figure 5 shows the results with respect to f1_score metric: the best model is the one with $C = 100, gamma = 0.01$ and the RBF Kernel.

References

[1] *Dataset.* URL: https://librivox.org/.

[2] *Feature extraction.* URL: https://librosa.org/doc/latest/feature.html.

[3] *Librosa.* URL: https://librosa.org/doc/latest/index.html.

[4] *Mel-frequency cepstrum.* URL: https://en.wikipedia.org/wiki/Mel-frequency_cepstrum.