Data Science in Biomedicine

ECG: a Machine Learning approach

Authors:
Pablo Carrera Flórez de Quiñones
Jose Llanes Jurado

Professor:
Jose David Martín Guerrero

Vniversitat de València

Contents

1 Introduction

An electrocardiogram (ECG) is one of the most powerful tools to inform about the functionality of the cardiovascular system [1]. It could be defined as the graphical representation of the heart activity during a period of time and is used to determine different cardiovascular diseases and metabolic alterations. The way is a EGC is studied can be viewed in two approaches, the pattern recognition, as it is usually done, and the understanding of the exact electrical vectors recorded by an ECG as they relate to cardiac electrophysiology, which is more difficult due to the quantitative aspect of the process.

The focus for studying this at large scales is on the analysis of huge datasets of ECG from different patients to classify the different health issues found in them. Commonly, this classification is done by humans, which convert the task in a really time consuming labour and makes it really prone to errors. Categorizing and detecting different waveforms and morphologies in the signal is one of the main task to succeed in the classification. So at the end of the day it is up to subjectivity of the doctor to classify a ECG signal into one category or another.

In order to automatize this process and avoid the errors provided by the human biases, some computational approaches are done. In the last years, more specifically, many machine learning approaches have been suggested [2]. The usual way to face this classification problem it to analyze a one, or more, massive datasets of ECG signals labeled by pri,or.

Most of these approaches involve a preprocessing step for preparing the signal. And then, these handcrafted features, which are mostly statistical summarizations of signal windows, are extracted from these signals and used in further analysis for the final classification task. For this final classification step, conventional machine learning approaches for ECG analysis have been developed in terms of support vector machines, multi-layer perceptrons and decision trees.

These handcrafted features provide an acceptable representation of the signal, but based on recent machine learning studies, automated feature extraction and representation methods are proven to be more scalable and are capable of making more accurate predictions, this allow the model to learn the features that are best suited to the specific task that it is dedicated to carry out [3, 4, 5]. This approach provides a more accurate representation of ECG signal, and using them, the models can compete with a human cardiologist in analyzing the signal [6].

This work is organized as follows: after this introduction, in the section 2 we make a presentation and exploration of the dataset we are considering for our study; in the section 3 we are applying different supervised methods in order to discover hidden patters of the data without the use of labels; then in section 4 we are using the labels to implement many supervised methods to get a taste of the accuracy that we can obtain and to establish a benchmark for more complicated models; in section 5 we are applying some deep neuronal network architectures in order to study how the accuracy of the simpler models can be improved; finally in section 6 we are doing a recap and a comparison between all the presented methods. All the models exposed are developed by ourselves in Python using libraries such as sklearn and keras.

2 Exploratory Data Analysis

In this work we are using the PhysioNet MIT-BIH Arrhythmia ECG dataset [7, 8] as data source for the labeled ECG records. This is the most extended and commonly used dataset for ECG, so it will provide a valid comparison of our methods with the ones applied by other researchers.

This dataset consists of ECG recordings from 47 different subjects recorded at the sampling rate of 360Hz. Then, since ECG beats are the inputs of the proposed methods, we need a method for preprocessing ECG signals and extracting beats [9]. The steps used for extracting beats from an ECG signal are:

1. Split the continuous ECG signal to 10s windows and select a 10s window from an ECG signal.
2. Normalize the amplitude values to the range of between zero and one.
3. Find the set of all local maximums based on zerocrossings of the first derivative.
4. Find the set of ECG R-peak candidates by applying a threshold of $0:9$ on the normalized value of the local maximums.
5. Find the median of R-R time intervals as the nominal heartbeat period of that window T.
6. For each R-peak, select a signal part with the length equal to $1:2T$.
7. Pad each selected part with zeros to make its length equal to a predefined fixed length.

and finally, each beat is annotated by at least two cardiologists. Since there are to many heart disorders, the annotations in this dataset can be used to create five different beat categories in accordance with Association for the Advancement of Medical Instrumentation EC57 standard [10].

Category	**Label**	**Annotations**
N	0	Normal Left/Right bundle branch block Atrial escape Nodal escape
S	1	Atrial premature Aberrant atrial premature Nodal premature Supra-ventricular premature
V	2	Normal Premature ventricular contraction Ventricular escape
F	3	Fusion of ventricular and normal
Q	4	Paced Fusion of paced and normal Unclassifiable

Table 1: Summary table of the correspondence between the annotation done by cardiologist, the categories created for summing up the heart disorders and the labels proposed for each category for our models.

As can be seen in the table 1, there are five different labels in our final dataset. The first denote healthy patients and the rest are related with patients suffering some king heart issue. As we can see, the categories are stablished following subjective criteria, that may not represent actual features of the signal, so the idea of grouping them in less categories having common insights can be very productive. In order to check the intracategory and intercategory similarities and differences we are ploting some random beats in the Figure 1.

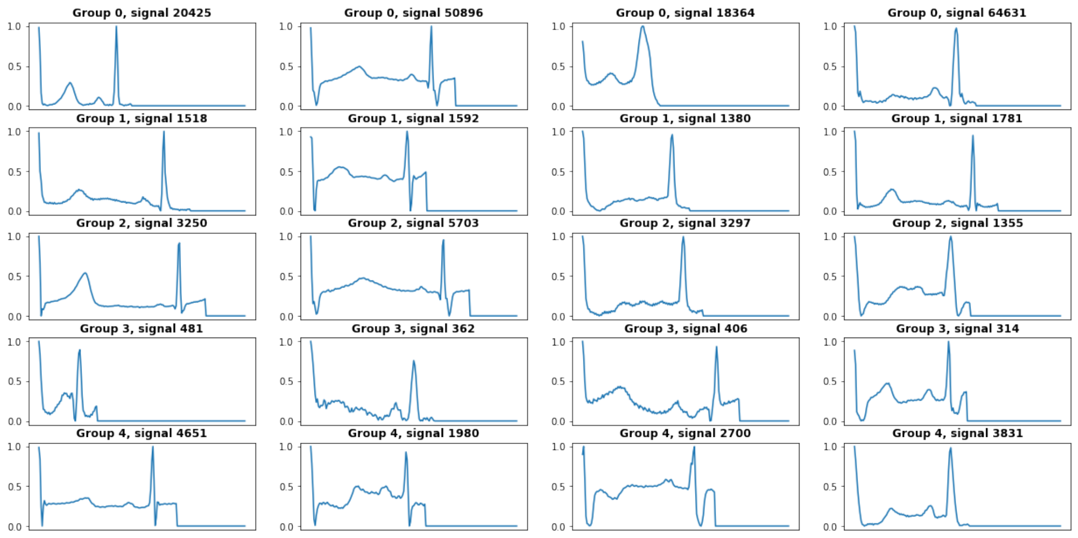


Figure 1: Graphical representation of random beats of the PhysioNet MIT-BIH Arrhythmia ECG dataset. There are 4 random beats for each of the 5 categories.

We can see in Figure 1 that there are huge differences between random signals inside the same category. These differences are that noticeable that it seem difficult even to say it two beats pertain to the same category. This makes us realize even more the necessity of an automatic treatment of the signals and make us think about an alternative approach to the problem. Since the signals are very different between them, we can approach the classification of the beats into 5 categories according to the standart labels of Table 1, or to approach the classification of the beats into 2 categories, healty vs. rest.

In order to see which of the two proposed approaches fits better our problem is important to see the distribution of the signals into the different categories. In order to see this exploration in a qualitative way we are ploting the countplots of the signals in Figure 2, and for complementing it in a quantitative way we are computing the ratios of the categories in Table 2.

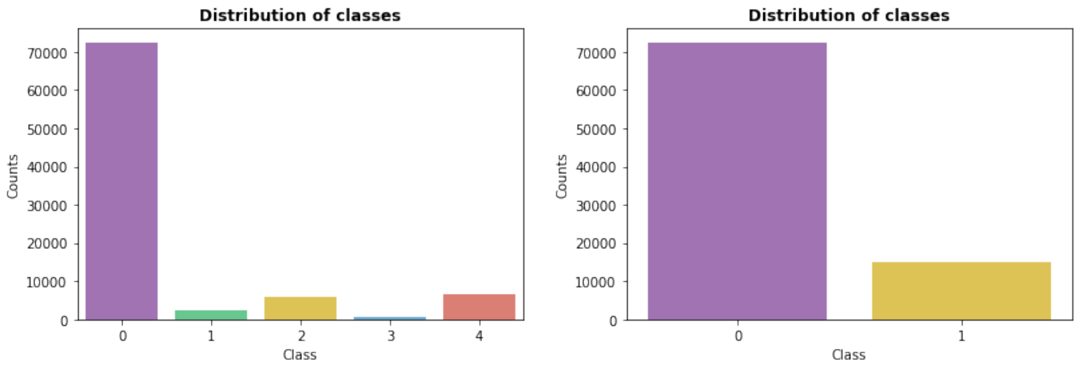


Figure 2: Graphical representation of the distribution of beats into the 5 classes of Table 1 and into the 2 categories of the healthy vs. rest approach.

Category	**Ratio**	**Binarized Category**	**Ratio**
N	0.8277	Healthy	0.8277
S	0.0254	Non-Healthy	0.1723
V	0.0661		
F	0.0073		
Q	0.0735		

Table 2: Summary table of the percentage of every different classes studied in the problem.

As we can see in Figure 2 and Table 2, there is a huge difference between the population of the different categories, that is, we have a imbalanced classes problem in which there is a class which groups the majority of the data, approximately the 82% and the rest are much less populated. There are many ways of facing this issue:

- Use standard methods of Machine Learning: the results that can be obtained from these methods will be highly biased, the accuracy of the models can be high due to the high presence of the majoritarian class. That is not representing the reality behind the problem, we are actually interested in the minoritarian ones, but serves a benchmark for better approaches.
- Use of tree-based models: these models work with ranges of values, so if the minoritarian categories are well localized, these models can reach a good performance without major modifications.
- Use of deep learning models: these models include a huge number of parameter, so they can learn perfectly complex structures capable to distinguish between the minoritarian classes if enough quantity of data is provided.
- Use methodologies for imbalanced classes: there are methodologies developed with the idea to cope with the problem of imbalancing. We are going to consider undersampling, which consist in randomly select sample from the majoritarian category to reduce its size, and oversampling, which consist in randomly duplicate the sample of the minoritarian category to increase its size.

3 Unsupervised algorithms

Once we have the ECG signals preprocessed and decomposed into separated beats we can start to develop Machine Learning algorithms in order to get a model of classification of the data. The first way to face this problem will be the use of unsupervised algorithms in order to obtain natural groupings of the data and see if these groupings reproduce the categories established by the human annotators.

In order to explore the different approaches that we can give to this decoding of the data we have developed many different kinds of algorithms:

- Linear methods: the first step to produce a dimensionality reduction of the data and to generate natural groups is to check the performance of linear methods. We are using Principal Component Analysis (PCA) for this purpose.
- Manifold learning: when the linear methods do not preform well, we can try to implement a set non-linear methods known as Manifold Learning. We are using Isometric Feature Mapping (Isomap) and t-distributed Stochastic Neighbor Embedding (t-SNE) for this purpose.

- Clustering: when the dimensionality of the data cannot be reduced without loosing too much information we can try to develop clustering algorithms in order to get natural groupings of the data. We are using K-means and Hierarchical Clustering for this purpose.

3.1 Linear methods

Identifying a linear manifold embedded in a higher-dimensional space is closely related to the classical statistics problem of linear dimensionality reduction. The recommended way of accomplishing linear dimensionality reduction is to create a reduced set of linear transformations of the input variables. Linear transformations are projection methods, and so the problem is to derive a sequence of low-dimensional projections of the input data that possess some type of optimal properties.

3.1.1 PCA

Principal component analysis (PCA) was introduced as a technique for deriving a reduced set of orthogonal linear projections of a single collection of correlated variables, where the projections are ordered by decreasing variances. Then we can choose the projections which encode the major part of the variance, archiving a huge reduction of the data to consider. We have tried to perform a reduction to 2 and 3 Principal Components, as can be seen in Figure 3.

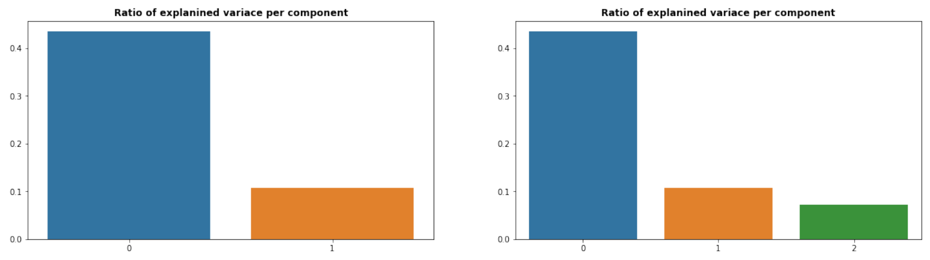


Figure 3: Graphical representation of the ratio of explained variance per component in a PCA reduction to 2 and 3 Principal Components respectively.

We can see in Figure 3, that the reduction is not so bad since with 2 components we reproduce the 54.17% of the variance and with 3 components the 61.32%. However this is not enough to reach the performance of a human annotator, so we cannot get satisfied with this decomposition. But we can still use it to get a 2-dimensional representation of the data in order to see the distribution of the labels in developing of other methods. We make this representation in figure 4

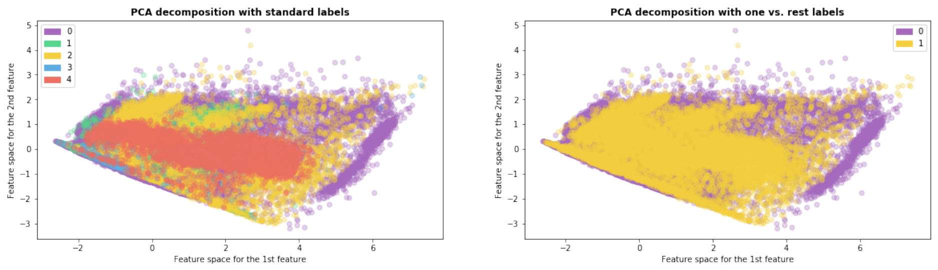


Figure 4: Graphical representation of the labels into a 2-dimensional scatterplot in the 2 first Principal Components of the data.

In Figure 4 we can see that the different minoritarian categories are more or less localized in the 2-dimensional space, but the majoritarian one distributes along all the space. This gives an insight of what we can expect from the problem, the distribution of features between the healthy individuals is so disperse that it blurs the differences with the individuals with hearth issues.

3.2 Manifold Learning

We have seen that we cannot make a linear decomposition such that the data are clearly separated, so we have to pass to the application of non-linear methods. The goal of these algorithms is to recover the full low-dimensional representation of an unknown nonlinear manifold embedded in some high-dimensional space, where it is important to retain the neighborhood structure of the original space. When this manifold is highly nonlinear, as could be the case here, these algorithms outperform the usual linear techniques.

3.2.1 ISOMAP

The Isometric Feature Mapping (Isomap) produces a mapping of the original data to a low-dimensional manifold by preserving the geometrical properties of the data by the conservation of distances between samples. Here the non-linearity comes from the assumption that the Euclidean distances in the original manifold have to be conserved into the geodesic distances of the new manifold. We have tried to perform a reduction to a 2 dimensional manifold in Figure 5.

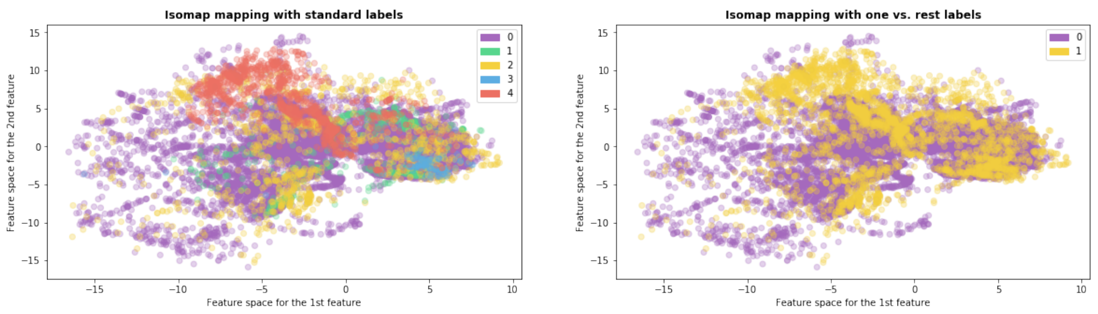


Figure 5: Graphical representation of the labels into a 2-dimensional scatterplot in the 2-dimensional manifold of the data produced by the Isomap algorithm.

We can see in Figure 5 that, as in the case of PCA, the distribution of the healthy individuals, the class 0, distributes along all possible directions while the different minoritarian classes are more or less localized. So we can conclude that this method either provides a satisfactory decomposition of the data.

3.2.2 t-SNE

The t-distributed Stochastic Neighbor Embedding (t-SNE) algorithm comprises two main stages. First, t-SNE constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a high probability of being picked together while dissimilar points have an extremely small probability of being picked together. Second, t-SNE defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the Kullback–Leibler divergence between the two distributions with respect to the locations of the points in the map. e have tried to apply this technique to perform a reduction to a 2 dimensional manifold in Figure 6.

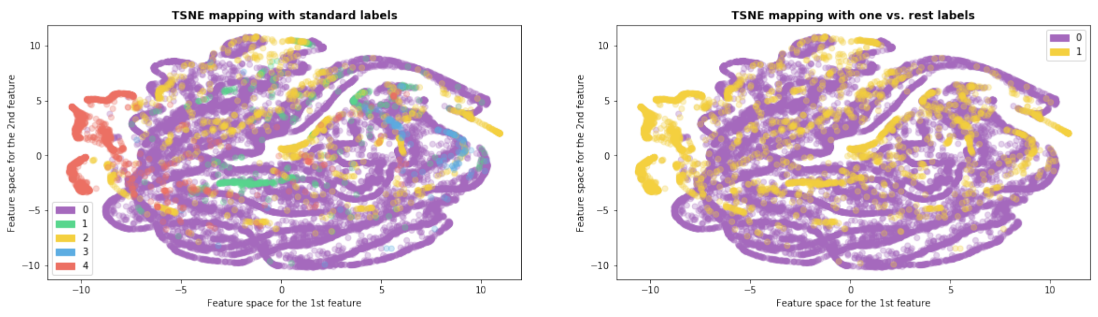


Figure 6: Graphical representation of the labels into a 2-dimensional scatterplot in the 2-dimensional manifold of the data produced by the t-SNE algorithm.

We can see in Figure 6 that, as in the case of PCA and Isomap, the distribution of the healthy individuals, the class 0, distributes along all possible directions while the different minoritarian classes are more or less localized. So we can conclude that this method either provides a satisfactory decomposition of the data and we have to try more sophisticated grouping methods.

3.3 Clustering

Clustering aims at finding groups in a given data in such a way that patterns grouped together are more similar or closer than patterns in other groups. This assumes nothing about the data, and so, incorporate no bias into the classification of the samples. There are many ways of implementing this groupings but in this work we are considering only the two most utilized, the K-means algorithm and the Hierarchical clustering algorithm.

3.3.1 K-means

The K-means algorithm is one of the most popular inside the Machine Learning community by its simplicity. Given a number of clusters it produces a centroid for each cluster representing the geometric centers of them. Then it computes the distances from each point to the centroid and make the groups regarding which centroid is closer them. Then the centroids are recomputed and the points are reassigned. This iterative procedure continues until the clusters remain static.

As we have seen, one of the biggest problem of this algorithm is that the number of clusters has to be introduced by hand before performing the analysis. The standard way to cope with this is the elbow curve method. The elbow curve method consists in trying different numbers of clusters and see how the inertia of the model, the sum of the distances to the center of the clusters, reduces by incorporating many clusters. Then, we choose as optimum number of clusters the one that produces not a significant improvement from the precedent. In Figure 7 we have represented the results of applying this method to our data.

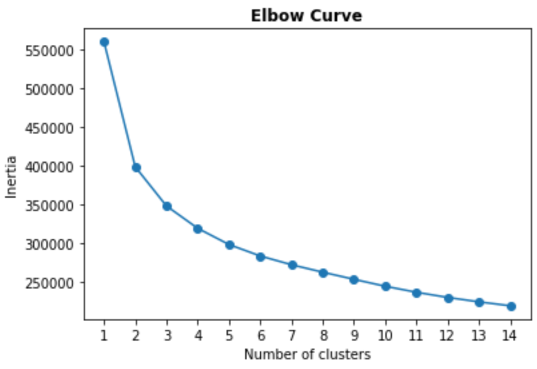


Figure 7: Elbow curve associated with the application of K-means algorithm to our data.

We can see in Figure 7 that 5 is the optimum number of clusters, reproducing the classification done by the human cardiologists. We can see also that making clustering with only 2 groups is far from being optimum, but we are developing this idea just for comparison. In figure 8 we represent the results of the application of K-means algorithm with 5 and 2 clusters.

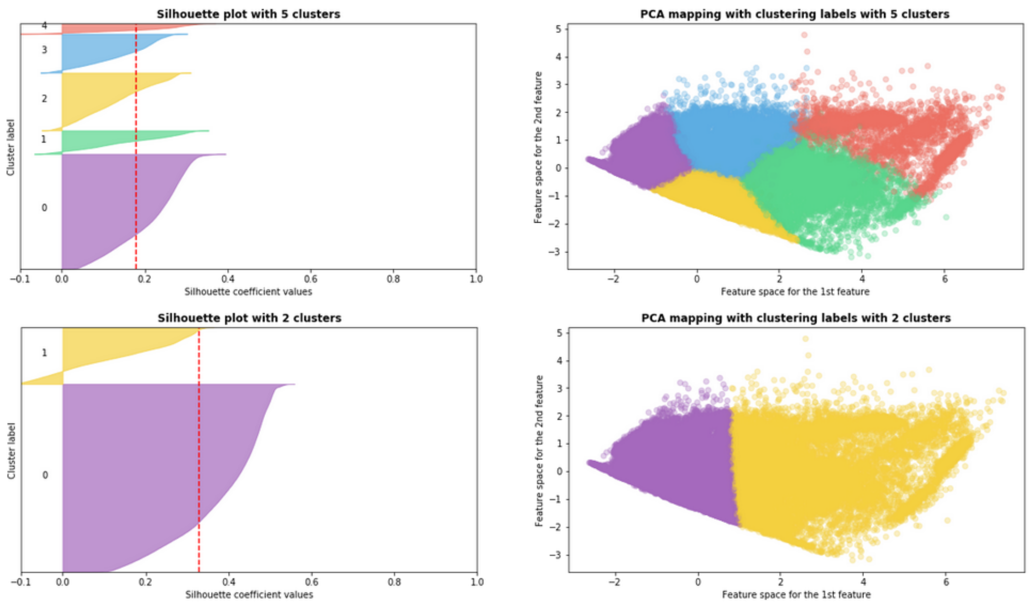


Figure 8: Representation of the results of K-means algorithm with 5 and 2 groups respectively. At left we can see the silhouette plots and at right the labels plotted into a scatterplot in the 2-dimensional PCA space.

3.3.2 Hierarchical Clustering

Hierarchical methods obtain a sequence of nested partitions of the data instead of a single clustering result. This is convenient in this problem, since it starts grouping by the samples that resemble more between them, so we can expect that the minoritarian categories are grouped together from the beginning. To achieve this idea, we can start from one cluster per sample and them keep joining the two closest clusters until one ends up with a unique cluster. This is known as Agglomerating clustering and seems to be the most intuitive idea for facing our problem. However, this method, despite its promising result has been shown to be too much computationally expensive to be developed in a normal computer, so we left open its result as a future possible work.

4 Supervised algorithms

In the previous section we have developed some unsupervised algorithms in order to get insights of the distribution of the data, and we have seen that there is no easy dimensional reduction that allow us to see natural grouping of the data, so it is the time to use the labels provided by the human annotators in order to train supervised algorithms. These algorithms will learn from the labelled samples and will eventually provide us with a model that can classify new samples.

We have developed algorithm in crescent order of complexity:

- Logistic Regression
- Decision Tree
- Random Forest
- Extremely Randomized Trees

then we have also implemented methodologies developed specifically for problems with imbalanced classes:

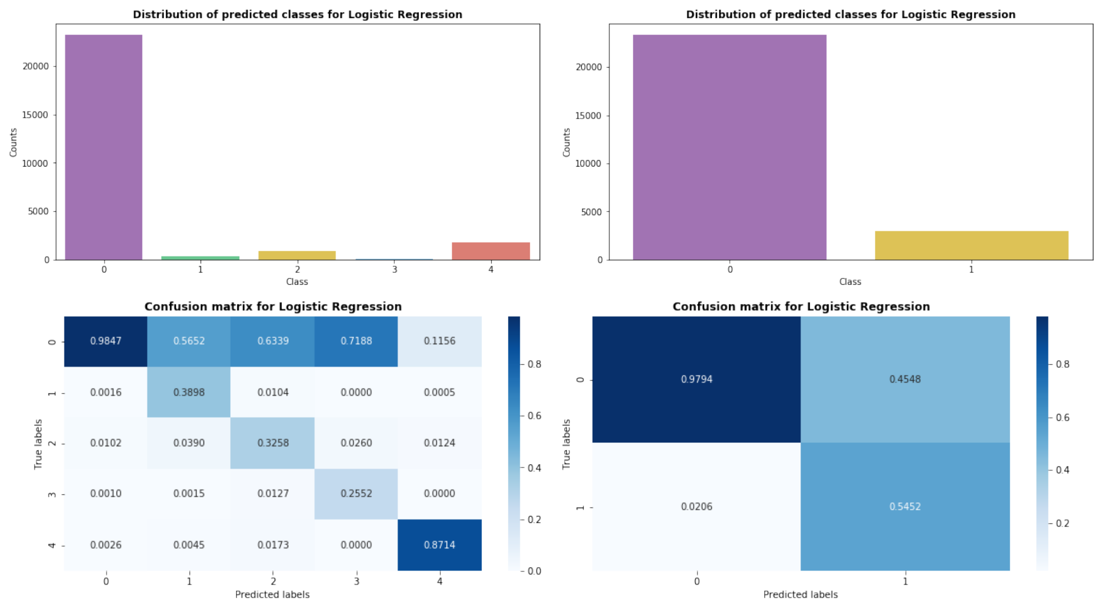
- Undersampling: this technique consists in sampling randomly the majoritarian class in order to reduce its size and make it more comparable with the size of the minoritarian classes.
- Oversampling: this technique consists in duplicate randomly the minoritarian classes in order to increase their size and make it more comparable with the size of the majoritarian class.

For testing the performance of these methods we have divided the dataset, with 87554 samples, into a train set and a validation test set, with a ratio of 70% for the train set and 30% for the test set. The idea is to train the different models with the train set and then to check its performance with the unseen data of the test set. For this validation task we are using three items for assesing the quality of the model:

- Distribution of the predicted labels: an easy way of compare the predictions with the real labels is to see if they follow a similar distribution of classes. This help to detect biasing respect to one or ones determined classes.
- Confusion matrix of the test predictions: this matrix compares the true labels of the samples with the predicted labels by computing a table of counts, or ratios, of which true labels have been predicted as other labels.
- Classification report: we compute some classic measures of performance for classification. We compute the precision, which is the ratio of all predicted positives that are actually positives; the recall, which is the ratio of all actual positives that are predicted as positives; and the f1-score, which is an harmonic mean of the precision and the recall. The intuitive meaning of the precision is that of the confidence of being right and the recall is the confidence of not being wrong. We are going to look at the f1-score since is a combined measure of both ones.

4.1 Logistic Regression

Logistic regression, despite its name, is a linear model for classification also known as logit regression, maximum-entropy classifier or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function. Since we are dealing with a non-binary classification problem, we are going to use the multiclass version of this method with a l_2 regularization scheme for avoiding overfitting. We show the results of this model in Figure 9.



```
              precision    recall  f1-score   support                    precision    recall  f1-score   support

           0       0.98      0.92      0.95     23249               0       0.98      0.91      0.94     23352
           1       0.39      0.83      0.53       313               1       0.55      0.85      0.66      2915
           2       0.33      0.67      0.44       842
           3       0.26      0.53      0.34        93
           4       0.87      0.95      0.91      1770

   micro avg       0.91      0.91      0.91     26267       micro avg       0.90      0.90      0.90     26267
   macro avg       0.57      0.78      0.63     26267       macro avg       0.76      0.88      0.80     26267
weighted avg       0.95      0.91      0.93     26267    weighted avg       0.93      0.90      0.91     26267
```

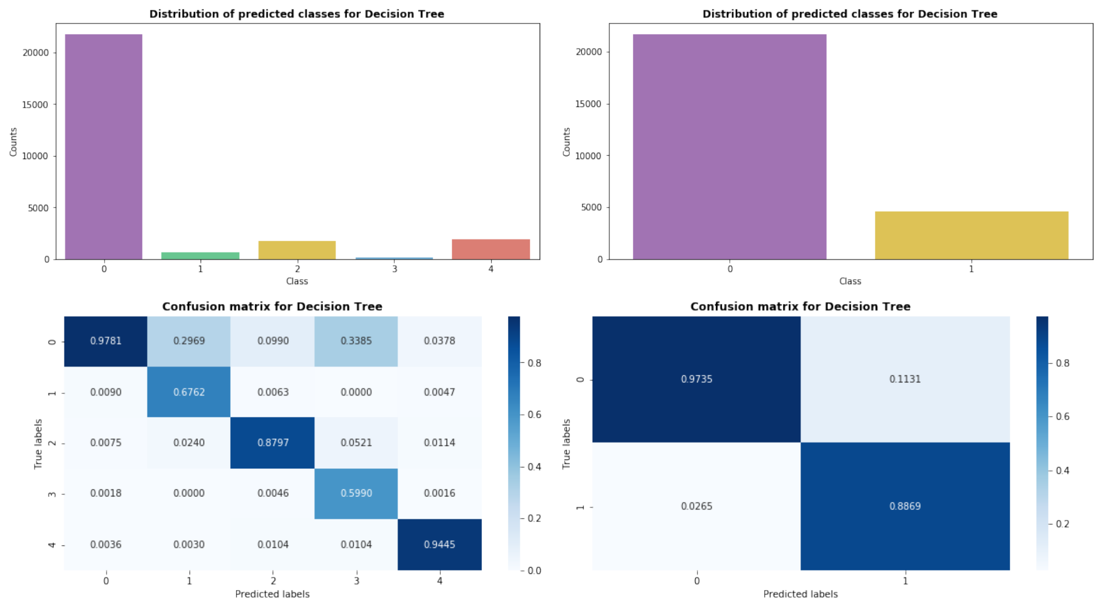
Figure 9: Representation of the results of Logistic Regression with 5 and 2 classes at left and right respectively. At top we can we can seethe distribution of the predicted labels for the test set, at medium we can see he confusion matrix of the classification over the test set and at bottom we can see a classification report including the precision, the recall and the f1-score of each class plus an average of them.

Where we can see various things:

- Looking at the distribution of the labels, we can see that it resembles the one of the original data, so we can say that, in principle, the model has no bias with any class.
- Looking at the confusion matrix, we can see that it shows that the model do not confuse classes 1,2,3 and 4 between them but only with the 0 class. This occurs due to the high variability of the class 0 individuals, the healthy ones as we have seen before in the PCA representations.
- Looking at the classification report, we can confirm what we have said before. The class 0 is well recognized generally, but the other classes are recognized poorly in favour of the class 0.

4.2 Decision Tree

We can see that the Logistic Regression might be too simple for our problem due to its intrinsic difficulty and the imbalancing between classes. Its well known that tree-based models work very well in this kind of problems. So we are starting this approach with a Decision Tree. The goal of this method is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. We are using a tree which uses the entropy as impurity criterium and without limits in the depth. We show the results of this model in Figure 10.



```
              precision    recall  f1-score   support                   precision    recall  f1-score   support

           0       0.98      0.98      0.98     21773              0       0.97      0.98      0.97     21678
           1       0.68      0.68      0.68       667              1       0.89      0.87      0.88      4589
           2       0.88      0.88      0.88      1739
           3       0.60      0.69      0.64       166
           4       0.94      0.95      0.95      1922

   micro avg       0.96      0.96      0.96     26267      micro avg       0.96      0.96      0.96     26267
   macro avg       0.82      0.83      0.82     26267      macro avg       0.93      0.93      0.93     26267
weighted avg       0.96      0.96      0.96     26267   weighted avg       0.96      0.96      0.96     26267
```

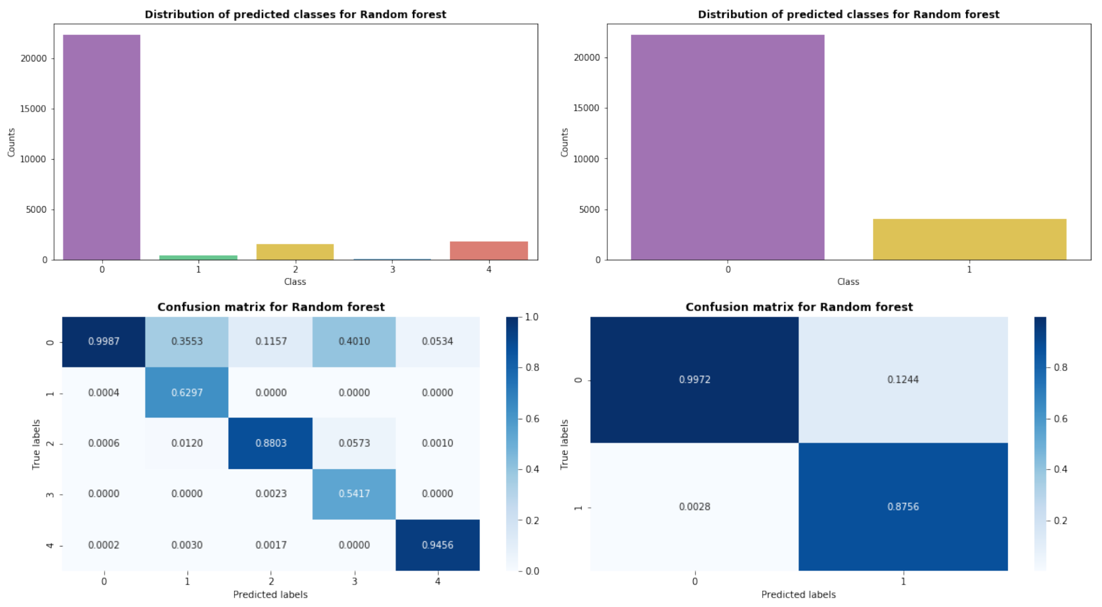
Figure 10: Representation of the results of Decision Tree with 5 and 2 classes at left and right respectively. At top we can we can seethe distribution of the predicted labels for the test set, at medium we can see he confusion matrix of the classification over the test set and at bottom we can see a classification report including the precision, the recall and the f1-score of each class plus an average of them.

Where we can see various things:

- Looking at the distribution of the labels, as in the previous case, we can see that it resembles the one of the original data.
- Looking at the confusion matrix, we can see the same features that in the previous case but with some improvements in the first row, meaning that the confusion between the class 0 and the others have improved.
- Looking at the classification report, we can confirm what we have said about the confusion matrix. Here we can see that the improvement in the recall of all the classes has suffered a huge improvement.

4.3 Random Forest

The Random Forests is a model constituted by a ensemble of trees, in our case 50, where each tree in is built from a sample drawn with replacement from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.



```
              precision    recall  f1-score   support

           0       1.00      0.97      0.99     22332
           1       0.63      0.98      0.77       429
           2       0.88      0.98      0.93      1564
           3       0.54      0.95      0.69       109
           4       0.95      1.00      0.97      1833

   micro avg       0.97      0.97      0.97     26267
   macro avg       0.80      0.98      0.87     26267
weighted avg       0.98      0.97      0.98     26267
```

```
              precision    recall  f1-score   support

           0       1.00      0.97      0.99     22244
           1       0.88      0.98      0.93      4023

   micro avg       0.98      0.98      0.98     26267
   macro avg       0.94      0.98      0.96     26267
weighted avg       0.98      0.98      0.98     26267
```

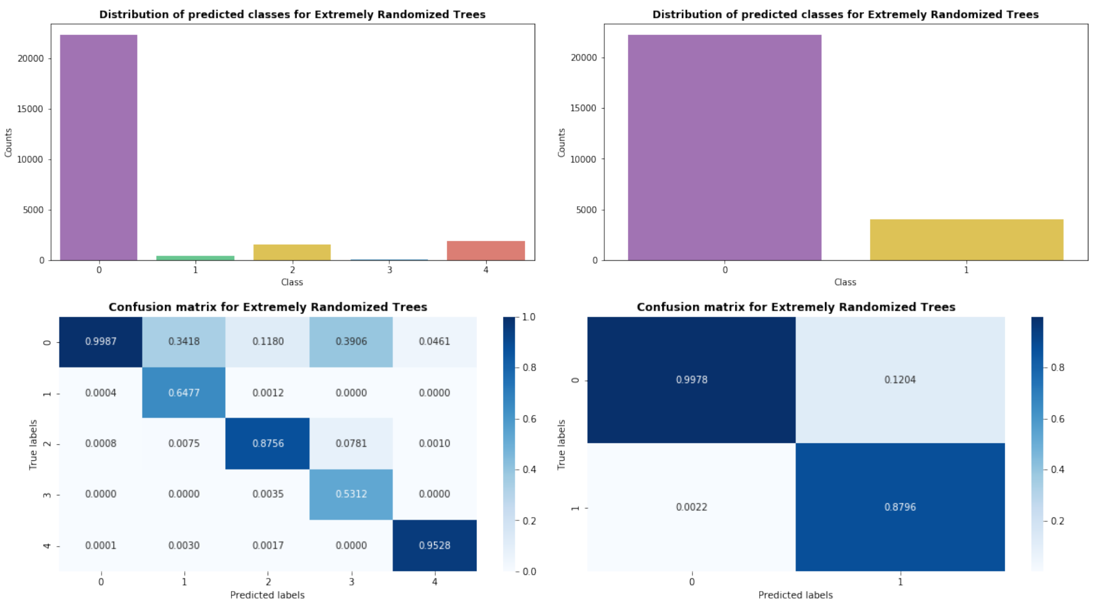
Figure 11: Representation of the results of Random Forest with 5 and 2 classes at left and right respectively. At top we can we can seethe distribution of the predicted labels for the test set, at medium we can see he confusion matrix of the classification over the test set and at bottom we can see a classification report including the precision, the recall and the f1-score of each class plus an average of them.

In figure 11 we can see various things:

- Looking at the distribution of the labels, as in the previous cases, we can see that it resembles the one of the original data.
- Looking at the confusion matrix, we can see even more improvement in the classification.
- Looking at the classification report, we can confirm what we have said about these models. We can see that the improvement in the recall of all the classes has improved even more, and consequently the f1-score is reaching very high values.

4.4 Extremely Randomized Trees

As a final step in this section we are considering Extremely Randomized Trees, here randomness goes one step further in the way splits are computed. As in random forests, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule. This usually allows to reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias. However this also uses to increase the effectivity of the models. We are using here 50 trees in order to being able to compare it with the Random Forest.



```
              precision    recall  f1-score   support

           0       1.00      0.97      0.99     22310
           1       0.65      0.98      0.78       443
           2       0.88      0.97      0.92      1560
           3       0.53      0.94      0.68       109
           4       0.95      1.00      0.97      1845

   micro avg       0.97      0.97      0.97     26267
   macro avg       0.80      0.97      0.87     26267
weighted avg       0.98      0.97      0.98     26267
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     22239
           1       0.88      0.99      0.93      4028

   micro avg       0.98      0.98      0.98     26267
   macro avg       0.94      0.98      0.96     26267
weighted avg       0.98      0.98      0.98     26267
```

Figure 12: Representation of the results of Extremely Randomized Trees with 5 and 2 classes at left and right respectively. At top we can we can seethe distribution of the predicted labels for the test set, at medium we can see he confusion matrix of the classification over the test set and at bottom we can see a classification report including the precision, the recall and the f1-score of each class plus an average of them.

In figure 12 we can see various things:

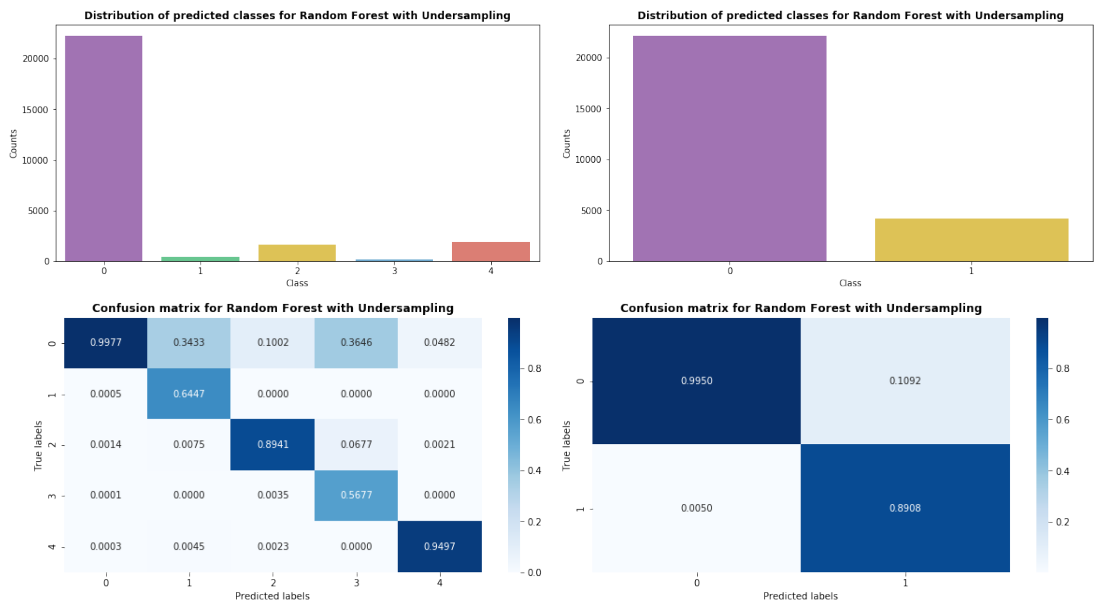
- Looking at the distribution of the labels, as in the previous cases, we can see that it resembles the one of the original data.
- Looking at the confusion matrix, we can see that the confusions have reduced a lot since the Logistic Regression.
- Looking at the classification report, we can confirm what we have said about these models. We can see that the recall of all the classes is almost perfect now.

4.5 Imbalanced methodologies

As we have said before, we are going to develop some models using the classic techniques for imbalanced classes. We are going to develop two methodologies:

- Undersampling: we are going to undersample the class zero to 20000 individuals, reducing by a factor 3 its size approximately. We expect this to improve the results since the gap between the population will be less pronunciated.
- Oversampling: we are going to oversample the minoritarian classes to 5000 inidivuals in each one, with a total of 20000 between them. We expect this also to improve the results since the gap between the population will be less pronunciated.

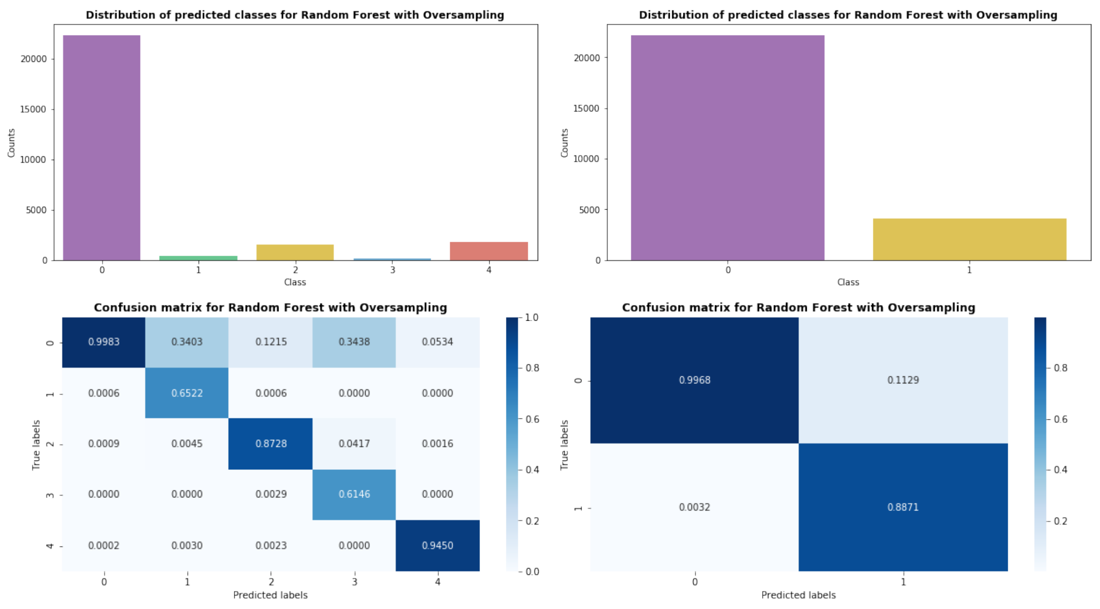
We are applying these methodologies to our best models until this point, the Random Forest and the Extremely Randomized Trees. Proceeding first with the Random Models:



```
             precision    recall  f1-score   support

           0       1.00      0.97      0.99     22259
           1       0.64      0.98      0.78       441
           2       0.89      0.97      0.93      1605
           3       0.57      0.93      0.71       117
           4       0.95      0.99      0.97      1845

   micro avg       0.98      0.98      0.98     26267
   macro avg       0.81      0.97      0.87     26267
weighted avg       0.98      0.98      0.98     26267
```

```
             precision    recall  f1-score   support

           0       0.99      0.98      0.99     22127
           1       0.89      0.97      0.93      4140

   micro avg       0.98      0.98      0.98     26267
   macro avg       0.94      0.98      0.96     26267
weighted avg       0.98      0.98      0.98     26267
```

Figure 13: Representation of the results of Random Forest with Undersampling with 5 and 2 classes at left and right respectively. At top we can we can seethe distribution of the predicted labels for the test set, at medium we can see he confusion matrix of the classification over the test set and at bottom we can see a classification report including the precision, the recall and the f1-score of each class plus an average of them.



```
              precision    recall  f1-score   support

           0       1.00      0.97      0.99     22311
           1       0.65      0.97      0.78       449
           2       0.87      0.98      0.92      1550
           3       0.61      0.95      0.75       124
           4       0.95      0.99      0.97      1833

   micro avg       0.97      0.97      0.97     26267
   macro avg       0.82      0.97      0.88     26267
weighted avg       0.98      0.97      0.98     26267
```

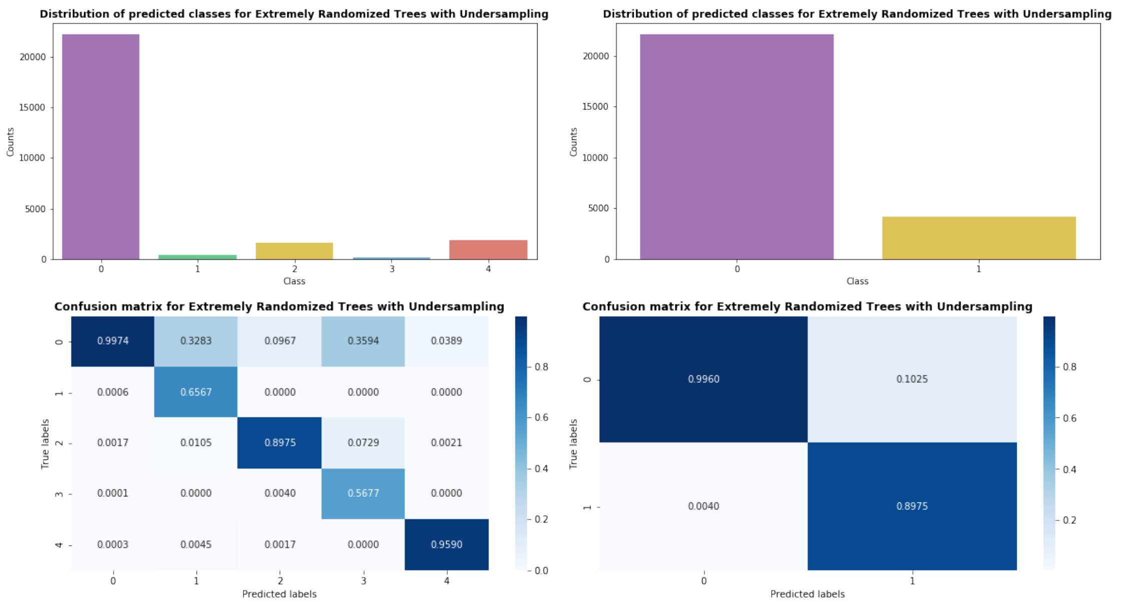
```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     22184
           1       0.89      0.98      0.93      4083

   micro avg       0.98      0.98      0.98     26267
   macro avg       0.94      0.98      0.96     26267
weighted avg       0.98      0.98      0.98     26267
```

Figure 14: Representation of the results of Random Forest with Oversampling with 5 and 2 classes at left and right respectively. At top we can we can seethe distribution of the predicted labels for the test set, at medium we can see he confusion matrix of the classification over the test set and at bottom we can see a classification report including the precision, the recall and the f1-score of each class plus an average of them.

We can see in Figures 13 and 14 that the models might have improved a little, but it is complicated to say something now, we better wait until the final comparison. But some insights

- Undersampling: seems to improve the result over the class 3, since it is the most minoritarian it is the most benefited by this method, since its proportion in the training set grows much faster than the other classes.
- Oversampling: also seems to improve the result over the class 3, since it is the most minoritarian the proportional growth is much notorious than the other classes.

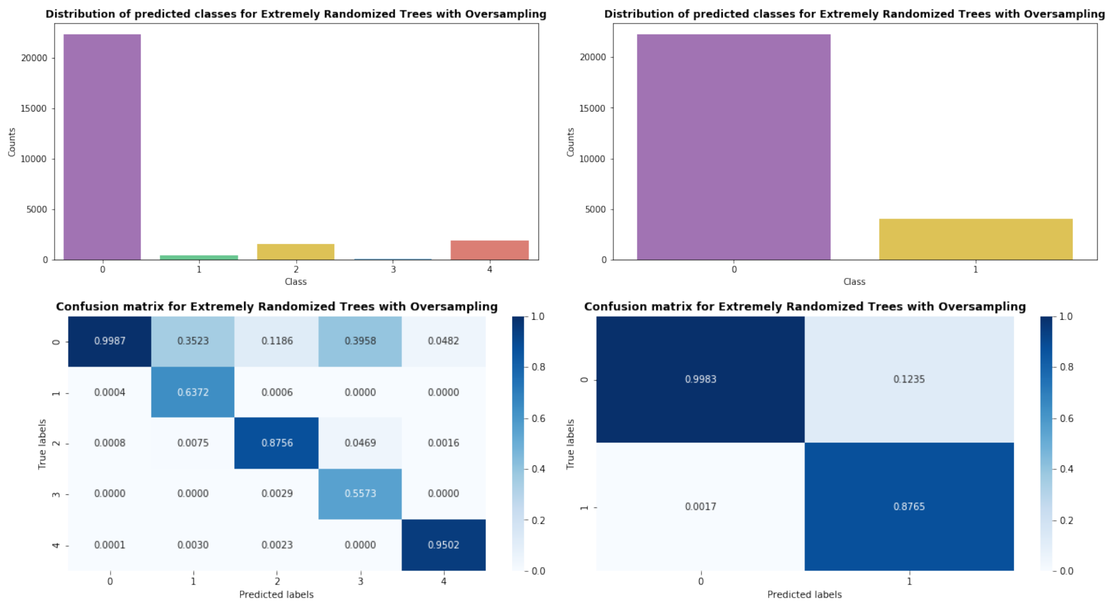
Now, we repeat this process with Extremely Randomized Trees:



```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     22216
           1       0.66      0.97      0.78       450
           2       0.90      0.96      0.93      1621
           3       0.57      0.92      0.70       118
           4       0.96      0.99      0.98      1862

   micro avg       0.98      0.98      0.98     26267
   macro avg       0.82      0.97      0.88     26267
weighted avg       0.98      0.98      0.98     26267
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     22118
           1       0.90      0.98      0.94      4149

   micro avg       0.98      0.98      0.98     26267
   macro avg       0.95      0.98      0.96     26267
weighted avg       0.98      0.98      0.98     26267
```

Figure 15: Representation of the results of Extremely Randomized Trees with Undersampling with 5 and 2 classes at left and right respectively. At top we can we can seethe distribution of the predicted labels for the test set, at medium we can see he confusion matrix of the classification over the test set and at bottom we can see a classification report including the precision, the recall and the f1-score of each class plus an average of them.



```
              precision    recall  f1-score   support

           0       1.00      0.97      0.99     22324
           1       0.64      0.98      0.77       435
           2       0.88      0.98      0.92      1555
           3       0.56      0.96      0.70       112
           4       0.95      1.00      0.97      1841

   micro avg       0.97      0.97      0.97     26267
   macro avg       0.80      0.98      0.87     26267
weighted avg       0.98      0.97      0.98     26267
```

```
              precision    recall  f1-score   support

           0       1.00      0.97      0.99     22265
           1       0.88      0.99      0.93      4002

   micro avg       0.98      0.98      0.98     26267
   macro avg       0.94      0.98      0.96     26267
weighted avg       0.98      0.98      0.98     26267
```

Figure 16: Representation of the results of Extremely Randomized Trees with Oversampling with 5 and 2 classes at left and right respectively. At top we can we can seethe distribution of the predicted labels for the test set, at medium we can see he confusion matrix of the classification over the test set and at bottom we can see a classification report including the precision, the recall and the f1-score of each class plus an average of them.

Where we can see similar insights than in the case of the Random Forest algorithm.

5 Deep Learning

We can see that the level of precision achieved with the Extremely Randomized Trees algorithm is high. But we are going to try to get a step further and develop some Deep Learning models in order to improve this even more. Deep Learning is based on the use of Neural Networks, models with complicated architectures which involve thousands of parameters. These models usually need huge quantities of data in order to properly train this set of parameters and avoid overfitting. Many different approaches of this methods have been used in the task of classifying ECG signals [6, 9, 11, 12].

5.1 Dense Neural Network

The first Deep Learing model considered is a dense model. This is the simplest architecture possible and involved connections between all the neurons in each layer with all the neurons in the next layer. We are developing a model with 187 inputs, the number of features of the signals, two hidden layers of 50 neurons each one, and a output layer of 5 or 2 neurons corresponding with the 5 or 2 categories of our problem. Here we show the resume provided by the Keras interface for the case of 5 classes:

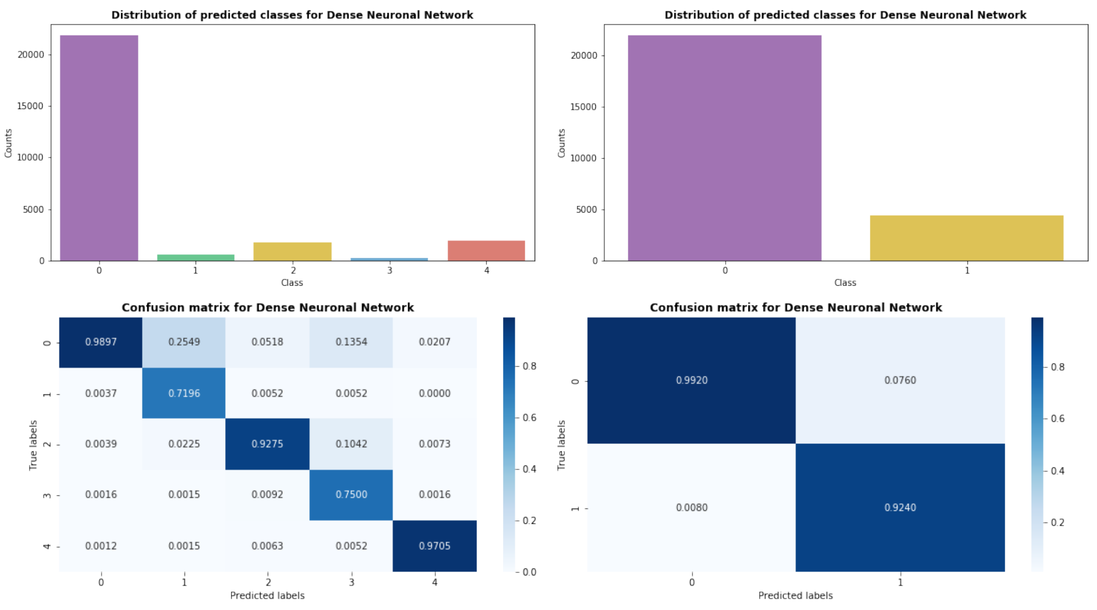
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 50)                9400
_________________________________________________________________
dense_2 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 255
=================================================================
Total params: 12,205
Trainable params: 12,205
Non-trainable params: 0
_________________________________________________________________
```

and the case of 2 categories:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 50)                9400
_________________________________________________________________
dense_2 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 102
=================================================================
Total params: 12,052
Trainable params: 12,052
Non-trainable params: 0
_________________________________________________________________
```

where we can see that this kind of models have a set of 12,205 and 12,052 trainable parameters respectively, so we have to be careful with the overfitting.

Proceeding in the same way that in the previous section, we show the results of this model in Figure 17



```
                 precision    recall  f1-score   support                   precision    recall  f1-score   support

           0          0.99      0.99      0.99     21843             0          0.99      0.98      0.99     21913
           1          0.72      0.84      0.78       570             1          0.92      0.96      0.94      4354
           2          0.93      0.92      0.93      1744
           3          0.75      0.72      0.74       199
           4          0.97      0.98      0.98      1911

   micro avg          0.98      0.98      0.98     26267     micro avg          0.98      0.98      0.98     26267
   macro avg          0.87      0.89      0.88     26267     macro avg          0.96      0.97      0.96     26267
weighted avg          0.98      0.98      0.98     26267  weighted avg          0.98      0.98      0.98     26267
```

Figure 17: Representation of the results of Dense Neuronal Network with 5 and 2 classes at left and right respectively. At top we can we can see the distribution of the predicted labels for the test set, at medium we can see he confusion matrix of the classification over the test set and at bottom we can see a classification report including the precision, the recall and the f1-score of each class plus an average of them.

Where we can see that, despite being one of the simplest possible models, its performances equals the one of the best model of the previous sections, the Extremely Randomized Trees model.

5.2 Convolutional Neural Network

Now, since a ECG signal can be interpreted as a 1-dimensional image, we can apply the architecture known as Convolutional Neural Network. This architecture incorporate new kinds of layers:

- Convolution: the use of local connections, that is, connection with only the nearest neurons, and weight sharing resemble the mathematical operation of convolution.
- Batch normalization: perform a normalization of the data entering into a hidden layer.
- Max-pooling: The max-pooling layer computes the max value of a selected set of output neurons from the convolutional layer and uses these as inputs to higher layers.

We propose the following architecture, inspired by [6, 9]:

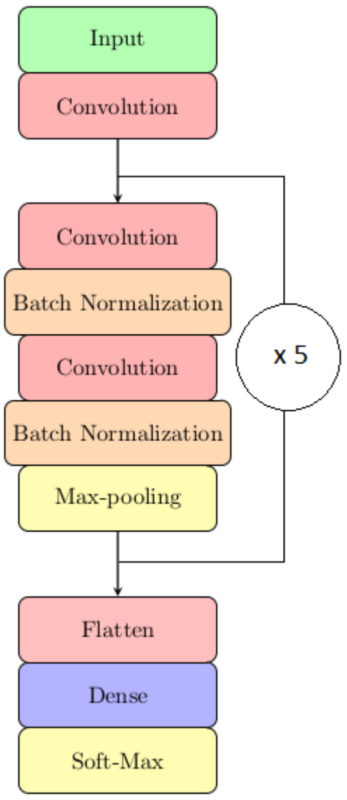
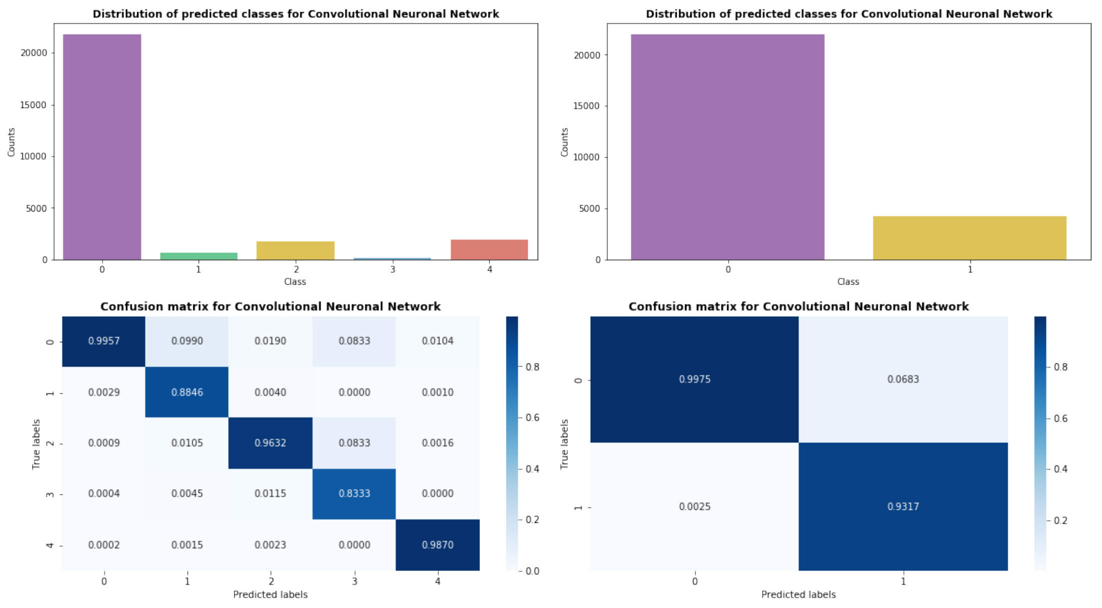


Figure 18: Schematic representation of the architecture of the Convolutional Neural Network developed.

here the convolutional layer is implemented with a filter value of 32 and a RELU activation function. For every convolution, a batch normalization is implemented in order not to let that the values of the convolution grow too much. After that, a max-pooling layer is implemented for two reasons: first to reduce the size of the data (a factor two in each step) and second to achieve the most relevant features of the data. The central part is repeated five times in order to improve the results of the model. At the final part we use a Flatten and a Dense layers in order to achieve the output classification. Here are not showing the keras resume because of its big size, but we can say that the complete model has a set of 51,365 trainable parameters, so we can guess that it will require much more time to be trained and is more prone to overfitting that the simpler model.

Then, proceeding in the same way that in the rest of the work, we show the results of this model in Figure 19



```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99     21783
           1       0.88      0.89      0.89       662
           2       0.96      0.97      0.97      1718
           3       0.83      0.84      0.84       191
           4       0.99      1.00      0.99      1913

   micro avg       0.99      0.99      0.99     26267
   macro avg       0.93      0.94      0.94     26267
weighted avg       0.99      0.99      0.99     26267
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99     21997
           1       0.93      0.99      0.96      4270

   micro avg       0.99      0.99      0.99     26267
   macro avg       0.96      0.99      0.98     26267
weighted avg       0.99      0.99      0.99     26267
```

Figure 19: Representation of the results of Convolutional Neuronal Network with 5 and 2 classes at left and right respectively. At top we can we can see the distribution of the predicted labels for the test set, at medium we can see he confusion matrix of the classification over the test set and at bottom we can see a classification report including the precision, the recall and the f1-score of each class plus an average of them.

Where we can see that, the new model outperform all the previous one by a huge margin in all categories, even in the most minoritarian one.

6 Comparison and final comments

In order to fully compare the supervised method that we have developed in this work we are going to show a table summing up all f1-scores, since this metric summarizes better the problem of the imbalancing than the precision and recall metrics separately. We show this comparison for the problem with 5 classes in Table 3 and for the problem with 2 classes in Table 4

Method	Class 0	Class 1	Class 2	Class 3	Class 4	Weighted Average
LR	0.95	0.53	0.44	0.34	0.91	0.91
DT	0.98	0.68	0.88	0.64	0.95	0.96
RF	0.99	0.77	0.93	0.69	0.97	0.98
RFU	0.99	0.78	0.93	0.71	0.97	0.98
RFO	0.99	0.78	0.92	0.75	0.97	0.98
ERT	0.99	0.78	0.92	0.68	0.97	0.98
ERTU	0.99	0.78	0.93	0.70	0.98	0.98
ERTO	0.99	0.77	0.92	0.70	0.97	0.98
DNN	0.99	0.78	0.93	0.74	0.98	0.98
CNN	0.99	0.89	0.97	0.84	0.99	0.99

Table 3: Resume of the f1-score for each class in each model of this work for the problem with 5 classes.

Method	Class 0	Class 1	Weighted Average
LR	0.94	0.96	0.91
DT	0.97	0.88	0.96
RF	0.99	0.93	0.98
RFU	0.99	0.93	0.98
RFO	0.99	0.93	0.98
ERT	0.99	0.93	0.98
ERTU	0.99	0.94	0.98
ERTO	0.99	0.93	0.98
DNN	0.99	0.94	0.98
CNN	0.99	0.96	0.99

Table 4: Resume of the f1-score for each class in each model of this work for the problem with 2 classes.

In these table we can see that increasing the complexity of the model increases its performance in this dataset. Tree-based methods have been shown to be very useful for this problem due to the imbalancing between the classes, but at the end of the day, the are outperformed by little by the Deep Learning algorithms. However is important to note that the application of specific methodologies for imbalanced classes has not improved so much the performance of these methods, since they are already capable of managing this problem. The performance of this classifier is higher than the achievable by a human annotator as can be seen in [6], so we can say that we have accomplish the goal that we have stated at the beginning of this project.

However, there are still thing that can be done to improve our work. The first one is to asses the parametrization of our models, due to our restrictions of computational capacity we cannot perform an adequate hyperparameter tuning of our models by the use of a Grid Search technique. A correct initialization of the parameters could improve the prediction of the models in the task of classifying the signals. The second one is to develop our own preprocessing of the raw ECG signals into beats. An end to end process is more likely to perform well since all the steps are correctly linked between them. Here we can introduce different techniques in order to improve the representation of the data, such as Fourier transforms or Wavelet transforms. Finally, we can also try to introduce more Neural Network-specific methodologies such that data augmentation. We left these ideas opened for future works.

References

[1] Daniel E. Becker. Fundamentals of electrocardiography interpretation. *Anesthesia Progress*, 53(2):53–64, 2006.

[2] Mincholé A. Martínez J. P. Laguna P. Lyon, A. and B. Rodriguez. Computational techniques for ecg analysis and interpretation in light of their contribution to medical advances. *Journal of the Royal Society, Interface*, 15(2):20170821, 2018.

[3] Serkan Kiranyaz, Turker Ince, and Moncef Gabbouj. Real-time patient-specific ecg classification by 1-d convolutional neural networks. *IEEE Transactions on Biomedical Engineering*, 63(3):664–675, 2016.

[4] Linpeng Jin and Jun Dong. Classification of normal and abnormal ecg records using lead convolutional neural network and rule inference. *Science China Information Sciences*, 60(7), 2017.

[5] U. Rajendra Acharya, Shu Lih Oh, Yuki Hagiwara, Jen Hong Tan, Muhammad Adam, Arkadiusz Gertych, and Ru San Tan. A deep convolutional neural network model to classify heartbeats. *Computers in Biology and Medicine*, 89:389–396, 2017.

[6] Pranav Rajpurkar, Awni Y. Hannun, Masoumeh Haghpanahi, Codie Bourn, and Andrew Y. Ng. Cardiologist-level arrhythmia detection with convolutional neural networks. *CoRR*, abs/1707.01836, 2017.

[7] Leon Glass Jeffrey M. Hausdorff Plamen Ch. Ivanov Roger G. Mark Joseph E. Mietus George B. Moody Chung-Kang Peng Ary L. Goldberger, Luis A.N. Amaral and H. Eugene Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[8] Mark R. G. Moody G. B. The impact of the mit-bih arrhythmia database. *IEEE Eng in Med and Biol*, 20(3):45–50, 2001.

[9] Mohammad Kachuee, Shayan Fazeli, and Majid Sarrafzadeh. Ecg heartbeat classification: A deep transferable representation. *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, 2018.

[10] A. for the Advancement of Medical Instrumentation et al. Testing and reporting performance results of cardiac rhythm and st segment measurement algorithms. *ANSI/AAMI EC38*, 1998, 1998.

[11] Özal Yıldırım, Paweł Pławiak, Ru-San Tan, and U. Rajendra Acharya. Arrhythmia detection using deep convolutional neural network with long duration ecg signals. *Computers in Biology and Medicine*, 102:411–420, 2018.

[12] Linpeng Jin and Jun Dong. Deep learning research on clinical electrocardiogram analysis. *Scientia Sinica Informationis*, 45:398, 2015.