IMA205 Challenge:
Cardiac Pathology Prediction

Benjamin TERNOT

May 7, 2023

Contents

I Introduction to our data set

First of all, let's summarize what data set was given for this challenge, and what was the objective.
I decided to work on a *Jupyter notebook* for this project.
The data, that we imported to our notebook, consisted on and will be designed by:

- A data set that can be used for training
  The data set consisted of 100 elements that are described through:
  - an unique id (from 1 to 100 included)
  - the height of the patient (in cm)
  - the weight of the patient (in kg)
  - 2 3D cardiac MRI, one during diastole (ED) and one during systole (ES)
  - the 2 segmentation images of this 2 3D MRI, that segment the Right Ventricle Cavity (RVC), the Left Ventricle Myocardium (LVM) and the Left Ventricle Cavity (LVC).
  - the label from 0 to 4 corresponding to the true diagnostic class of the patient
- A data test set, for which we have to find the correct label corresponding to the diagnostic class of the patient.
  It contains 50 elements, each consisting of:
  - an unique id (from 101 to 150 included)
  - the height of the patient (in cm)
  - the weight of the patient (in kg)
  - 2 3D cardiac MRI, one during diastole (ED) and one during systole (ES)
  - the 2 partial segmentation images of this 2 3D MRI, that segment the Right Ventricle Cavity (RVC) and the Left Ventricle Myocardium (LVM) but not the Left Ventricle Cavity (LVC).

Before testing any machine learning method, I decided to display some images (ex Fig 1), the distribution over the diagnostics of my data set used for training (Fig 2) and some dimensions of the images. Here I saw that:

- I had an even distribution of classes in my train data, that can avoid imbalances in my models over a majority class, so no oversampling would be needed.
- My MRI images were not of same shape for all my patients. A lot of pre processing would be needed.

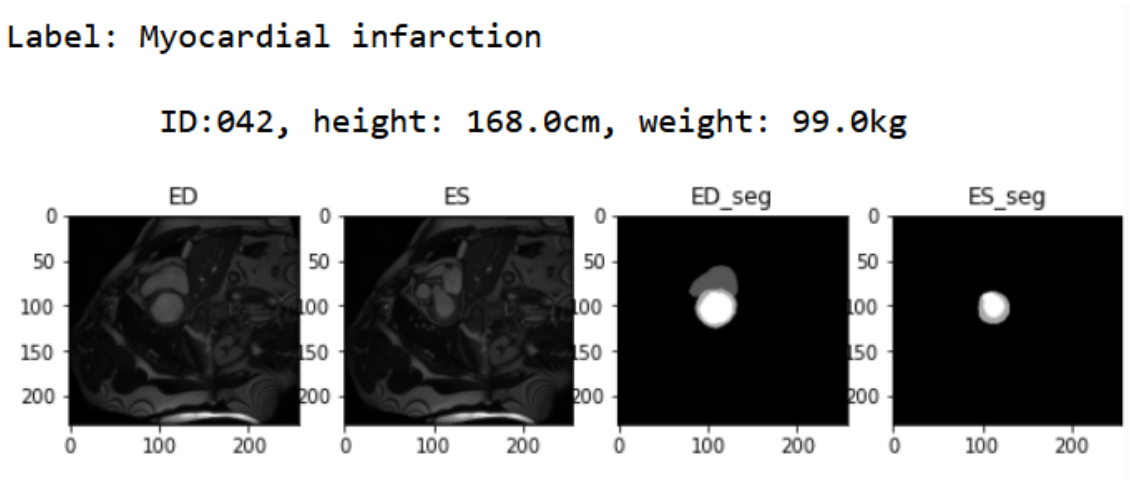


Figure 1: Example of the data from patient 42

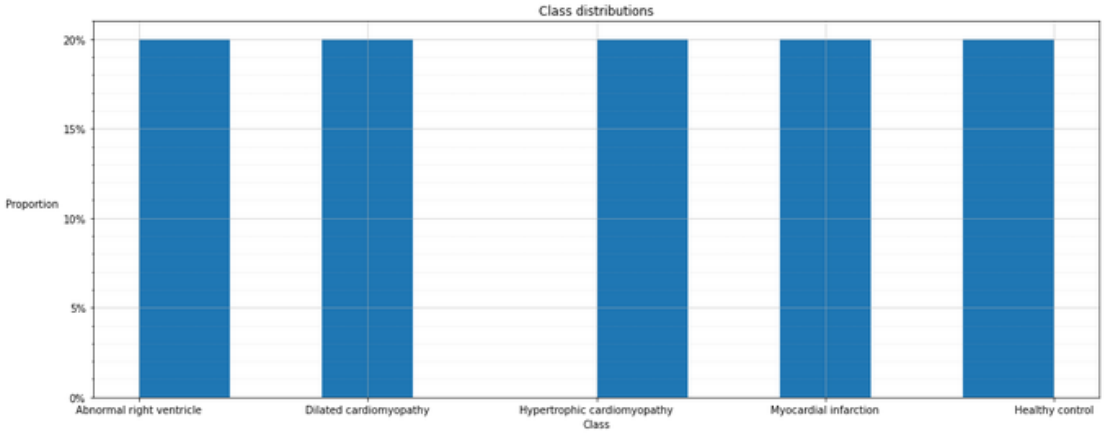


Figure 2: Distribution of classes in my train data

II Initial Preprocessing

I decided to begin with all the given data, without any smart selection of features, just to see if we can achieve a result (even if I was a bit hopeless).

II.1 Reshaping images

In order to achieve that, I decided to first reshape my images in order to have the same amount of pixels for each patient an images (simply by selecting the minimal shape given over all images), then flatten all my data for each patient and standardize it.
Noting that it gave me more than 760 000 "features" for each patient, I realized that I would never be able to train any model using all of that data.

II.2 Dimension reduction

So I decided to reduce this dimension madness, by selecting a number of features that kept about 95% of the variability of my data (using PCA). By doing that, I knew that I would have a big issue of overfitting my data, but I wanted to still give it a try.

II.3 Train and Validation set

In oder to verify the accuracy of models over unseen data, I choose to split my train data over 2 sets:

- the train set (X_train, y_train), that would be used to train the ML models
- the validation set (X_val, y_val), that would be used to evaluate the performances of my models over unseen data.

I just used a split method to achieve that, keeping 20% of my train data for evaluation, with the same even class distribution (Fig 3).
The data was then ready for first model training and predictions.

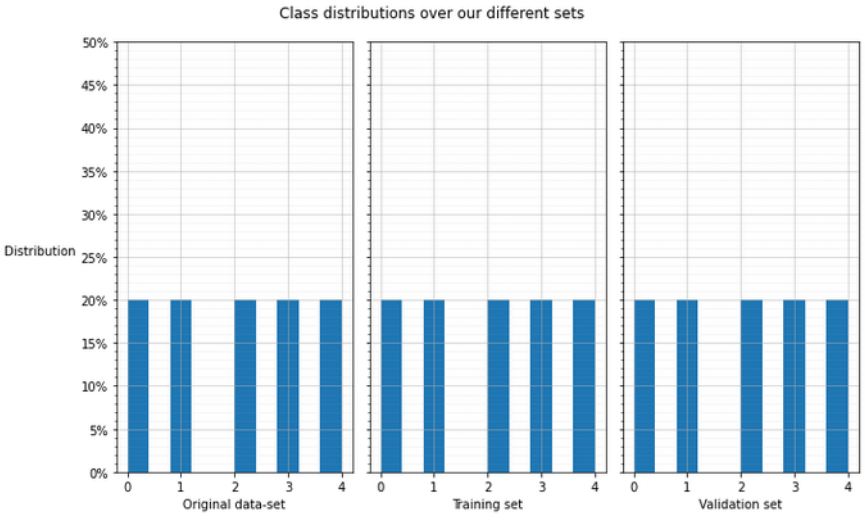


Figure 3: Distributions of classes in my train and validation sets

III Methods using trivial data and PCA

III.1 ML methods

I chose to test some classic methods seen in practicals sessions, such as *Linear Discriminant Analysis (LDA)*, *Quadratic Discriminant Analysis (QDA)*, *Perceptron*, *Linear SVM (LSVM)* and *Non Linear SVM (SVM)*.
Results obtained with these methods using the previous defined train and validation sets shown that I clearly overfitted, with accuracy varying between 25% and 50%, but that was a bit expected due to PCA reduction.
But, when I tested these method using Cross-Validation, I could clearly see that I had a very high standard deviation for every method and that then the methods tested could not be trusted at all (Fig 4).

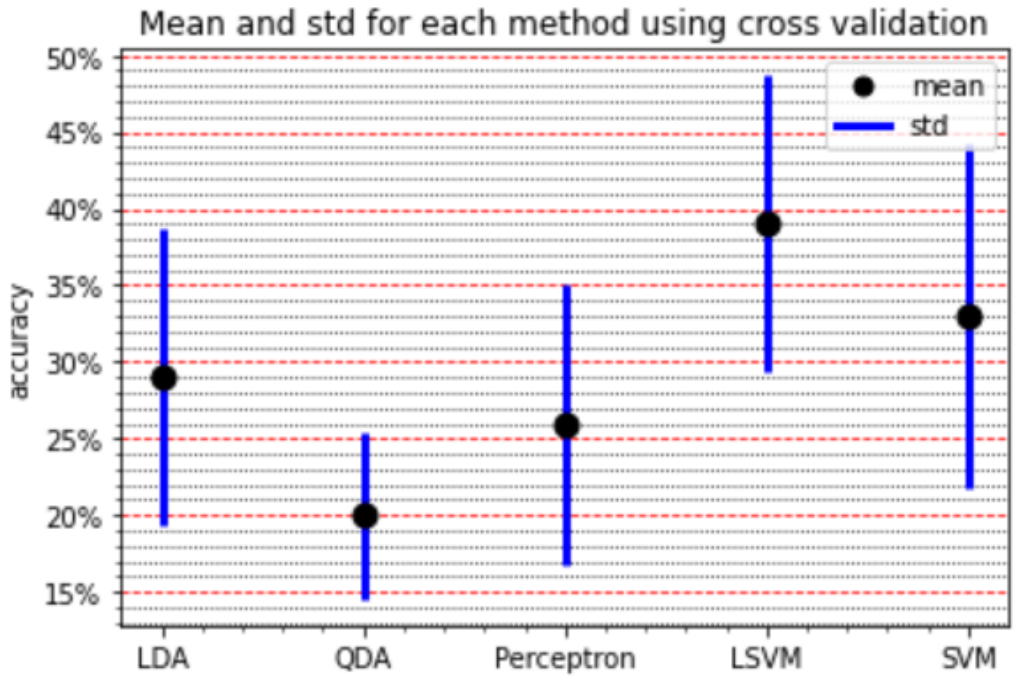


Figure 4: Cross-validation scores for ML methods based on PCA reduction

III.2 Deep learning method

Even if I knew that I probably should change my approach and defining smart features as adviced in the Kaggle challenge description, I wanted to go all the way through my idea and test it using a basic Artificial Neural Network.
I used for that a multi-layer perceptron, with 3 hidden layers of 256, 256 and 128 neurons (maybe it was high to begin with and I should have predicted it). I used Adam's optimizer.
The result was not that bad, but clearly shown an overfit, because my ANN converged at 100% accuracy but "only" gave 60% over my validation set (Fig 5). I then tried to find the best number of neurons using grid search cross validation over the parameters of my ANN, but could not find any amelioration even witgh lower number of neurons (Fig 6).
Clearly, the train set was not well used.

```
Categorization Accuracy : 60.00%
```

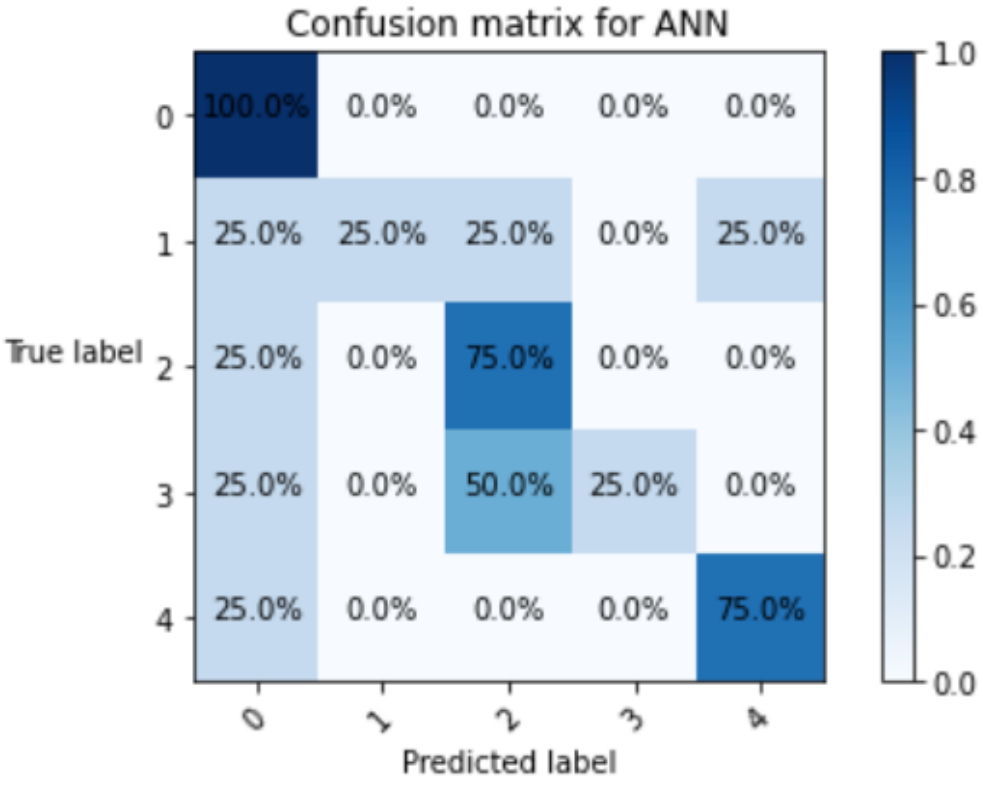


Figure 5: Ann results with 3 big hidden layers

```
Categorization Accuracy : 40.00%
```

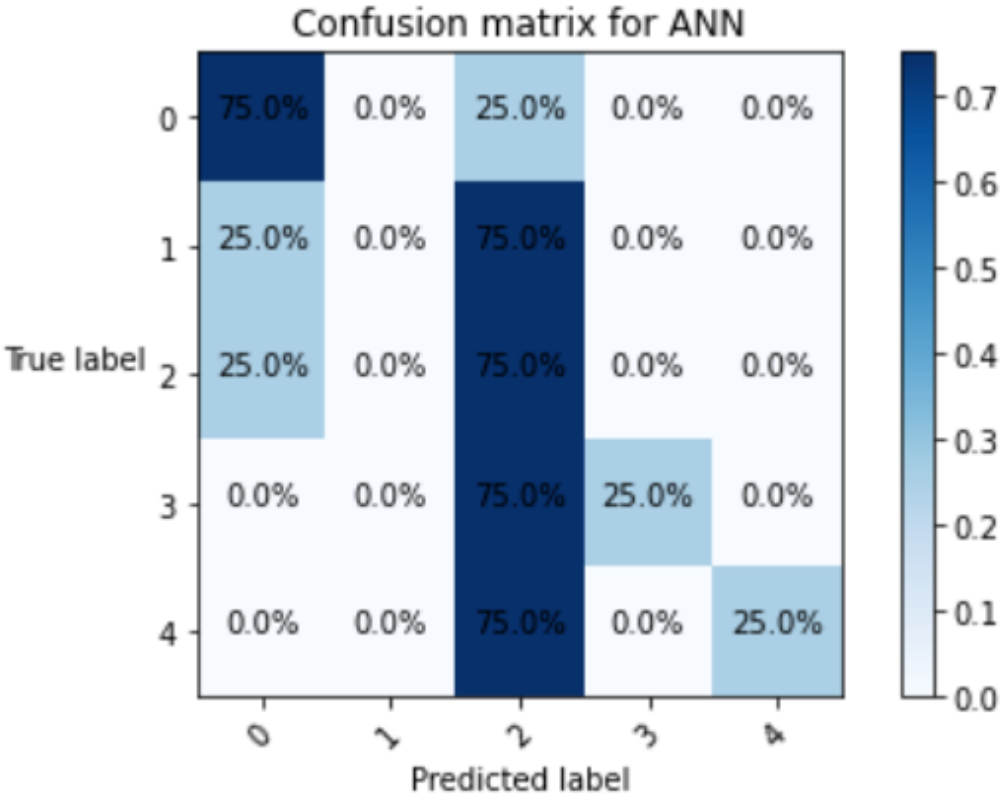


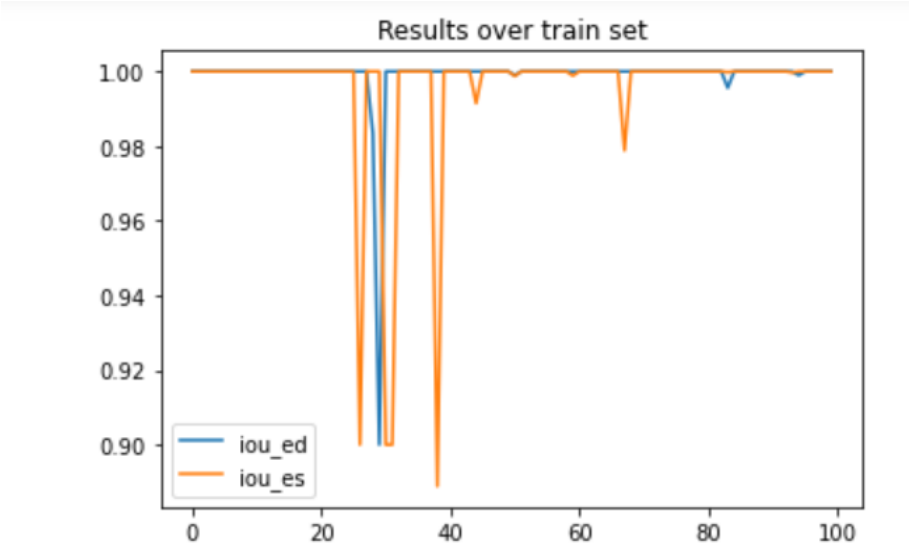
Figure 6: Ann results with 3 smaller hidden layers

IV Defining new features

As I saw that I kept having overfitting, I decided to go to the "smart" approach, that is defining features that were advised in the article, such as volumes of the differents cardiac elements.
The first issue with that was that I did not have access to any full segmented MRI image for my test set, so I needed to begin with finding a method to segment the LVC.

IV.1 Segmenting LVC

While many approaches could be used to detect the LVC in the images, I deeply thought and realized that the LVC was just the cavity surrounded my the LVM !
Knowing that, I only needed a method to correctly fill the hole inside the LVM of my partial segmented MRI.
Concretely, I found a method in *skimage.segmentation* named *flood_fill* that did that, and combined it with some geometric limits and initialization in order to be able to correctly find the LVC.
I tested this geometric method with the train data set in which I "hid" the segmentation of LVC and was able to re-segment it almost all the time (Fig 7). I also took advantage of the fact that in some cases the segmentation was harder to detect to store this "status" of segmentation into a new feature (maybe a MRI where the segmentation of LVC could not be found easily could give me an info on the diagnosis of the patient).




```
Results over train dataset:
ED:     mean:99.9%
        std:1.0%

ES:     mean:99.6%
        std:2.0%
```

Figure 7: Results of LVC segmentation in my data used for training using IOU

IV.2 Calculating features

Now that I had the missing segmentation (or a status that told me that the segmentation could not be reconstructed), I managed to compute some features such as :

- volumes of LVC, LVM and RVC during ES and ED phases.
- the ratio of these volumes between the 2 phases
- the thickness of LVM between LVC and RVC during both phases

That gave me a total of 24 features:

- height
- weight
- bmi = $\frac{weight}{height^2}$
- statuses of segmentation (8 dimensions because distribution over 4 statuses at each phase)
- volumes of LVC, LVM and RVC at each phase (6 dimensions)
- volume ratios (3 dimensions)
- mean and std of thickness of LVM between LVC and RVC at each phase (4 dimensions)

The training was then possible !

V Methods using new features

Using the new features, I chose to only focus myself over the ML methods, trying to find the bests parameters. Note that in order to have a coherent model that use the same type of data for train and test, i did not used the given full segmented RMI images of train set but rather the reconstructed ones.
After using train and validation sets to have an idea, I quickly moved to cross-validation and Grid Search cross-validation for the hyper parameters of the LSVM and SVM (Fig 8).
After founding them, it appeared that non linear SVM gave me the best results so I decided

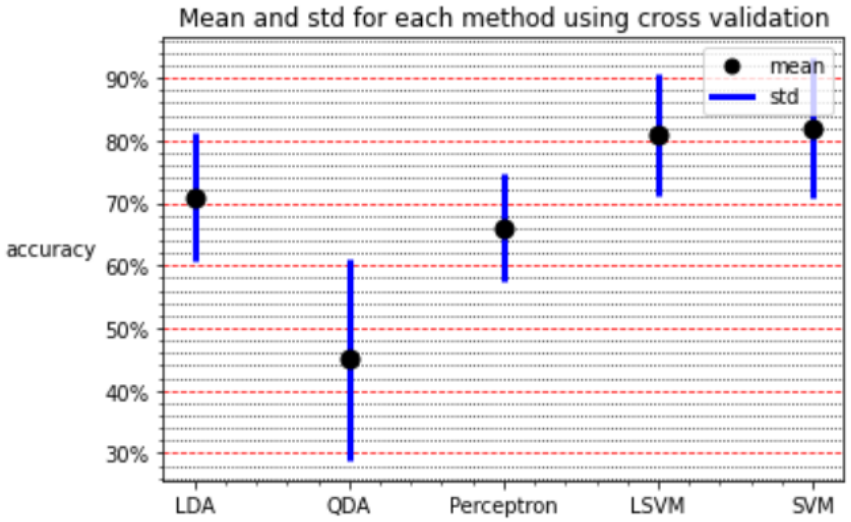


Figure 8: Cross-validation scores for ML methods based on new features

to predict the classification of my test set. Then I just display the distribution (Fig 9) and simply exported it in a csv file !

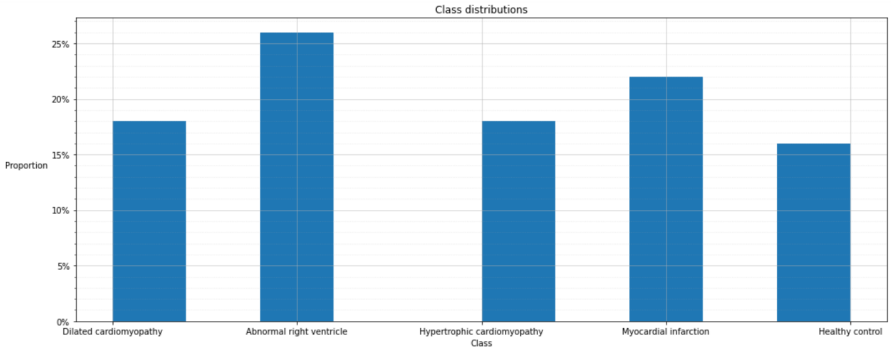


Figure 9: Class distribution for predicted results