1. Feature selection

As we can see on our datasets, especially the Chronic Kidney Disease’s (CKD) one, there are a lot of features. Indeed, there are 24 columns for the features. This may be a problem for our neural network (or any classification algorithm) : indeed, having a high search dimension increases the complexity of the algorithm and its run-time. The neural network will also require a huge amount of data to converge without under-fitting. Therefore, it is essential to define a strategy to reduce the features’ space. We have already dropped one column during the preprocessing phase, as more than 30% of its lines where empty. We will use a Principal Component Analysis (PCA) algorithm to drop redundant columns. We already have a standardize dataset thanks to the pre-processing step, which is essential to apply a PCA algorithm without suffering from high variance.

PCA consists in calculation the eigen values of our feature matrix, and to take the columns which as the highest eigen values (highest variance ratio). In our case, we define a threshold of 90%, and we take the first n columns with highest variance ratio which sums is higher than 90%. Performing a PCA on the CKD dataset yields to the selection of 10 columns (which ones?). On the Bank Note Dasaset, it gives …

1. Neural network model chosen

As we deal with numerical data (nor images or temporal data), one of the most appropriate layer to use here is the dense layer (also called fully-connected layer). In order to get a better precision in the predictions, we are going to use 3 dense layer. The first 2 layers will use “ReLU” as an activation function, which is the most commonly used activation function in hidden layers. The last dense Layer, composed of only one neuron (binary classification), will use the common sigmoid function.

In our model (see annex X), you will notice 2 particular layers. The batch normalization layer…. The dropout allows us to ignore the prediction of a part of the neurons (only during training), 15% of them here. This help our algorithm to avoid overfitting and to better generalize the results on another dataset.

Some parameters will be chosen later through a gridsearch (cf next part). We decided to perform this strategy on the number of epoch, the batch size, and the optimizer. We won’t perform gridsearch on the activation function for example as they may not have a very significant impact compared with the number of epochs or the batch size.

1. Training and testing strategy

A common practice is to split the dataset into a training, a validation and a test set. Each dataset has a precise function :

* Training set : train the weights of the neural network model chosen.
* Validation set : used as a test set to fine-tune the model (ie choose the best parameters or training strategy in order to get the best results).
* Test set : only used to test the final model, once we had optimize the training parameters with the validation test. It allows use to confirm the actual performance of our network.

Our strategy is here to divide the dataset in two : 0.75% for training/validation set, and 25% for the test set. We decided to take a bigger test set in order to challenge our model and to see if it can easily generalize its results on an unseen dataset.

On the training/validation set, we are going to train and fine-tune a model. We will do that in 2 main steps :

* Step1 : K-Fold cross-validation and gridsearch in order to find which parameters lead to the highest accuracy and the lowest loss. We will split the set in 10 folds, as it is a common value used for this kind of cross-validation. The gridsearch will be performed on 3 relevant parameters : the number of epochs, the batch size and the kind of optimizer. Playing on the number of epochs and on the batch size will have an impact on the quality of the convergence and also on its speed. The choice of the optimizer depends on the dataset’s characteristics.
* Step 2 : K-Fold cross-validation using the best parameters found in first step. We will keep the weights of the model which achieved the best accuracy and the lowest lost.

Once we have found our best model, we are going to train it on the whole dataset (it was trained on only 90% of the training/validation set) and then we are going to test it on the unseen test set to evaluate it.

1. Metrics to evaluate our model

Common => accuracy, loss. It’s easy to access it. But to test our model and see its power, its essential to look at other metrics. Explain which one is the most important to look at

Here, our classification strategy may have been better if we had looked at the Recall or other when we chose the best model. Indeed, as we’ll see later, many models achieved a good accuracy. Looking at other metrics could have helped us to choose the best model.

1. Results
2. CDK

Best parameters, validation, accuracy, final test. Attention : many 100% accuracy when we chose after K-fold cross-validation. We should maybe do less folds (5) in order to train on a smaller dataset. This will lead to worse results but will probably help us to better chose the algorithm. Do a test to confirm? What can we conclude ?

1. Banknote

Idem