|  |
| --- |
| Master 1 - ISII |
| Artificial Intelligence |
| Machine Learning Project |

|  |
| --- |
| Team N° 10   * Taleb Mehdi * Sennoun Merouane * Zouai Serine Maria * Aouanouk lmane * Zemmouri Arslane |

Table of Contents

[1. Introduction 3](#_Toc96539584)

[2. Project Descriptions 3](#_Toc96539585)

[2.1. Regression Problem 3](#_Toc96539586)

[2.2. Classification Problem 3](#_Toc96539587)

[3. Real Estate Valuation 4](#_Toc96539588)

[3.1. Dataset Description 4](#_Toc96539589)

[3.2. Preprocessing 4](#_Toc96539590)

[3.3. Linear Regression 6](#_Toc96539591)

[4. Immunotherapy Treatment Result 7](#_Toc96539592)

[4.1. Dataset Description 7](#_Toc96539593)

[4.2. Preprocessing 7](#_Toc96539594)

[4.2.1 Graphic Representation 8](#_Toc96539595)

[4.2.2 Pairwise Correlation of the features 8](#_Toc96539596)

[4.2.3 Missing values 9](#_Toc96539597)

[4.2.4 Removing Categorical values 9](#_Toc96539598)

[4.2.5 Detecting Outliers 10](#_Toc96539599)

[4.2.6 Feature scaling 11](#_Toc96539600)

[4.3. Logistic Regression 12](#_Toc96539601)

[4.3.1 Splitting the dataset 12](#_Toc96539602)

[4.3.2 definition of the functions 12](#_Toc96539603)

[4.3.3 Implementation of the Gradient Descent 12](#_Toc96539604)

[4.3.4 Testing different values of alpha 13](#_Toc96539605)

[4.4 Neural Network 14](#_Toc96539606)

[4.4.1 Splitting the dataset 14](#_Toc96539607)

[4.4.2 Tuning parameters for the model 14](#_Toc96539608)

[4.4.3 Final Results 15](#_Toc96539609)

[4.4.4 Confusion Matrix - Plotting the results 16](#_Toc96539610)

[4.5 SVM 17](#_Toc96539611)

[4.6 Decision Trees 18](#_Toc96539612)

[4.6.1 Train and Test Datasets 18](#_Toc96539613)

[4.6.2 Building the Decision Tree 18](#_Toc96539614)

[4.6.3 Text representation of the decision tree 19](#_Toc96539615)

[4.6.4 Feature importance in the Decision Tree: 19](#_Toc96539616)

[4.6.5 Graphic representation: 20](#_Toc96539617)

[Annex 21](#_Toc96539618)

[A.1 The GridSearchCV implementation 21](#_Toc96539619)

[A.1.1 Initializing the algorithm 21](#_Toc96539620)

[A.1.2 The final results 21](#_Toc96539621)

# Introduction

The project that we had to undertake is based on a subcategory of machine learning and artificial intelligence, which is Supervised Learning. Essentially, supervised learning is when the computer is taught by example. It learns from past data and applies the learning to present data to predict future events. In this case, both input and desired output data provide help to the prediction of future events.

Supervised learning can also be divided into two subcategories, classification and regression. Which are the two sorts of problems that we had to operate.

We had to apply the specific analysis methods to each category, to solve the two problems and train different models capable of predicting the best possible results.

On the other hand, we also had to comply with specific working methods to solve a machine learning problem. The use of a notebook is therefore essential for this, because it is used to present the analysis process step by step by arranging the code, images, text, output etc. in a step-by-step manner. Jupyter Notebook provides a full set of features that makes it one the best components of Python Machine Learning. Another tool that we used, this time for the arrangement and organization of work in general, is Git and GitHub, such as each member of the group made changes on the part of the notebook that corresponded with a specific part of the project (preprocessing, graphic visualization etc.)

In the following parts, we are going to discuss the achieved tasks in each of the two problems, explaining the dataset content, the different steps of preprocessing realized to improve its quality, and testing and visualizing the obtained results produced from the built models.

# Project Descriptions

## Regression Problem

One of the two problems that we had to process is a regression problem. A regression analysis must be considered when we deal with a problem that requires a prediction of a continuous value. Our regression problem is about valuating real estate properties given certain characteristics about them and their surroundings. Basically, the main task is to predict a house price, which is continuous value, considering a group of numerical/categorical features describing this house.

## Classification Problem

The second problem is a classification problem. This is the kind of problems that require the result of the prediction to be a categorical value. In our case it is a binary classification, for the reason that the model can only predict a 0 or 1 value for a given set of feature values concerning an instance. Our classification problem is about predicting if Immunotherapy, when used for wart treatment, produces a positive or negative result on a patient, given different informations about this one. A prediction of “1” being a result of a successful treatment and “0” of an unsuccessful one.

# Real Estate Valuation

## Dataset Description

The real estate valuation is a regression problem. The market historical data set of real estate valuation are collected from Sindian Dist., New Taipei City, Taiwan.

The data set was randomly split into the training data set (2/3 samples) and the testing data set (1/3 samples).

**Attribute Information:**

The inputs are as follows:  
X1=the transaction date (for example, 2013.250=2013 March, 2013.500=2013 June, etc.)  
X2=the house age (unit: year)  
X3=the distance to the nearest MRT station (unit: meter)  
X4=the number of convenience stores in the living circle on foot (integer)  
X5=the geographic coordinate, latitude. (Unit: degree)  
X6=the geographic coordinate, longitude. (Unit: degree)  
  
The output is as follows:  
Y= house price of unit area (10000 New Taiwan Dollar/Ping, where Ping is a local unit, 1 Ping = 3.3 meter squared)

## Preprocessing

## Linear Regression

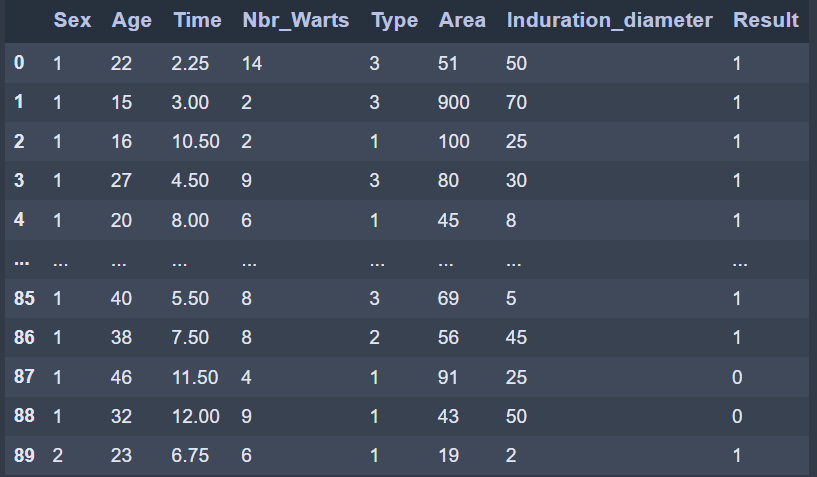
# Immunotherapy Treatment Result

## Dataset Description

This dataset contains information about wart treatment results of 90 patients using immunotherapy.

## Preprocessing

The imported dataset df:



### 4.2.1 Graphic Representation

We represent all the 7 features in comparison to ‘Result’ in 7 subplots:



### 4.2.2 Pairwise Correlation of the features

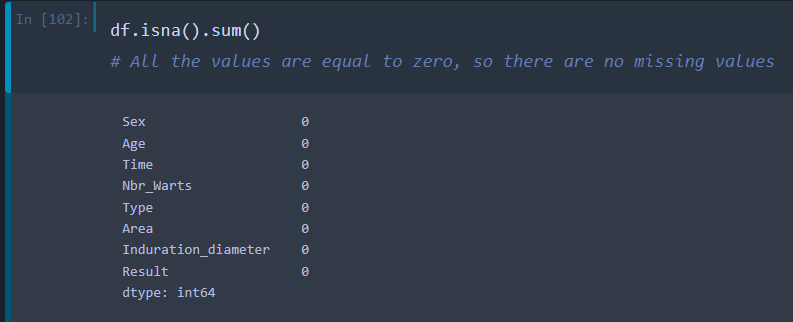
We use df.corr() to show pairwise correlation of all the features:



We notice that the features ‘Time’ and ‘Age’ have the strongest correlation to ‘Result’

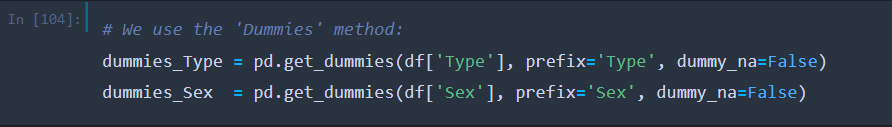
### 4.2.3 Missing values

We use the command **isna().sum()** to check if there are any missing values

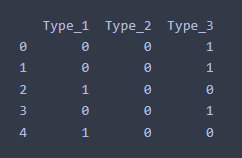


### 4.2.4 Removing Categorical values

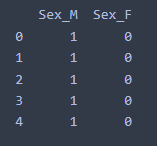
The features 'Sex' and 'Type' are of Categorical type, we have to transform them to a numerical type (int) using the 'Dummies' method:



‘Type’ dummies:

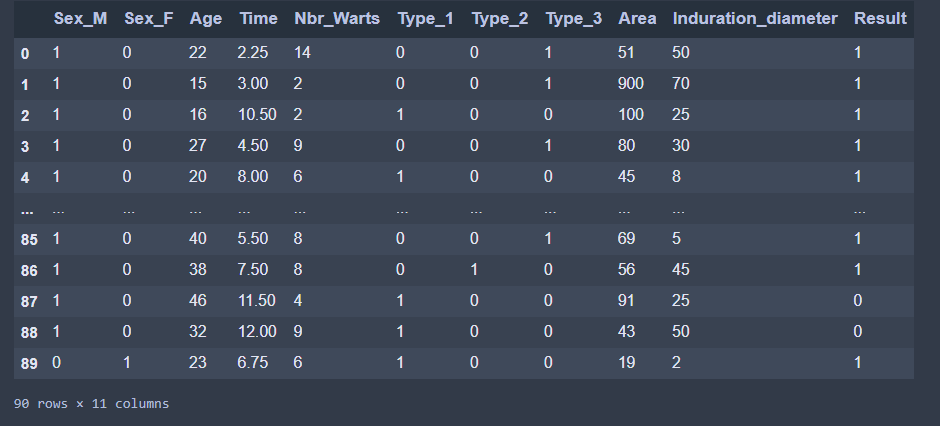


‘Sex’ dummies:

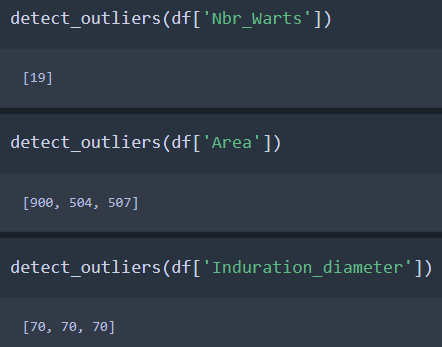
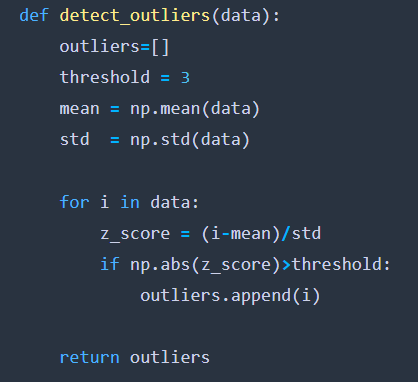


-We remove the features ‘Type’ and ‘Sex’ from the initial dataset df.

-We merge all the features to get our new and processed dataset:

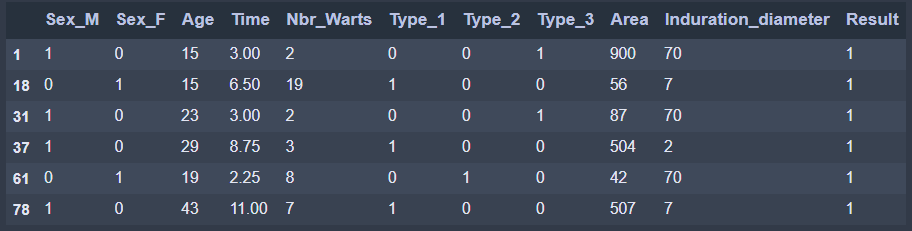


### 4.2.5 Detecting Outliers

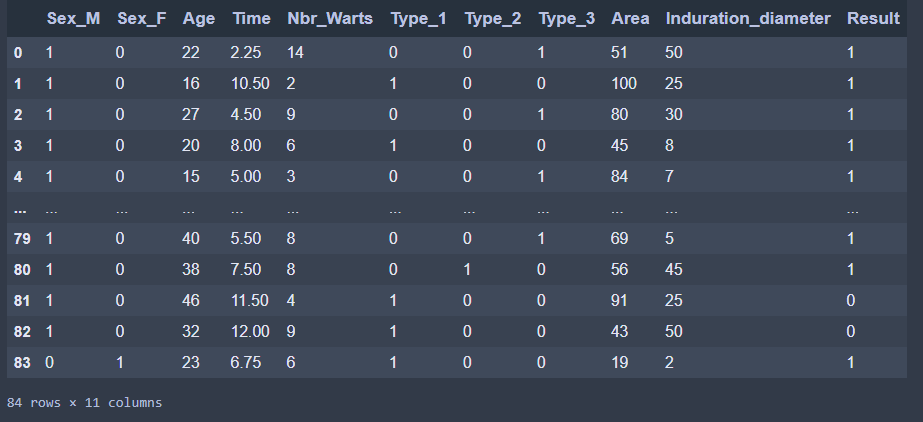
We use the **‘z\_score’** method to detect the outliers:

**Remark:** we checked all the other features and they don’t contain any outliers.

Table of all the outliers of ‘Nbr\_Warts’, ‘Area’ and ‘Induration\_diameter’:

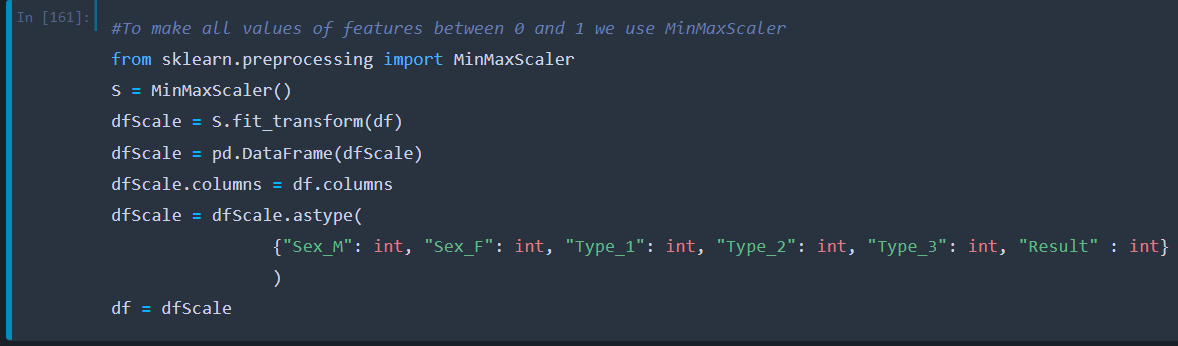


Df after removing the outliers:

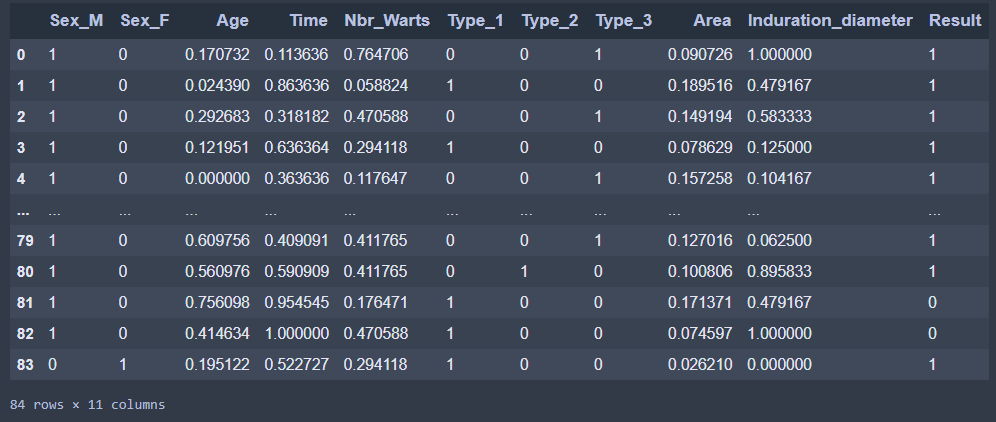


### 4.2.6 Feature scaling

To make all values of features between 0 and 1 we use MinMaxScaler from sklearn.preprocessing



Df after the feature scaling:



And this is also the final result of all of the pre-processing methods.

## Logistic Regression

### 4.3.1 Splitting the dataset

After splitting the dataset, we get the following shapes:

Une image contenant texte

Description générée automatiquement

### 4.3.2 definition of the functions

Une image contenant texte

Description générée automatiquement

### 4.3.3 Implementation of the Gradient Descent

Une image contenant texte

Description générée automatiquement

### 4.3.4 Testing different values of alpha

In this section we tested different values of alpha in order to find the best value to minimize the cost and found that 0.6 is the best value for our dataset. If we go further than alpha = 0.6, we will no longer see a big decrease in the final value of the cost.

Une image contenant texte

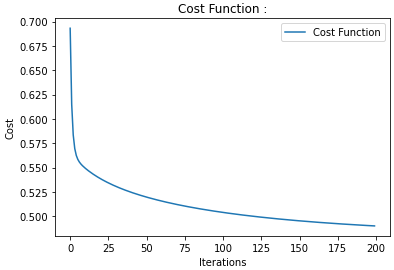
Description générée automatiquement

We can see below the decrease of the cost value while training the model.

Une image contenant texte

Description générée automatiquement

Ans here is the plot of the cost function.

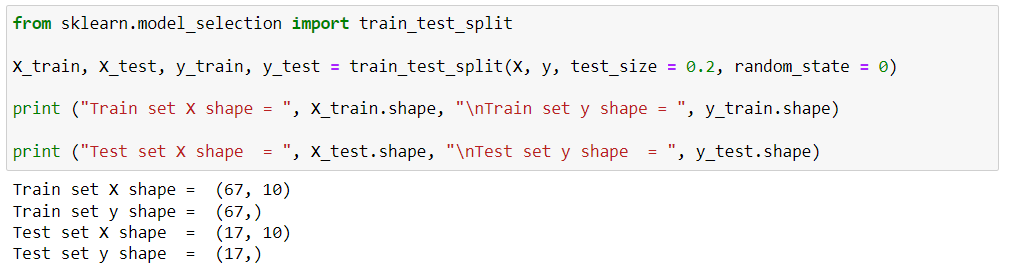


At the end, we get an accuracy of 77.77% when predicting the test set, which is a good value. (Picture of the accuracy function in the annex)

At the end of these steps, we use the predefined function in “sklearn” to compare our results, and we can see that we got an identical accuracy of 77.77%. (Picture of the code function in the annex)

## 4.4 Neural Network

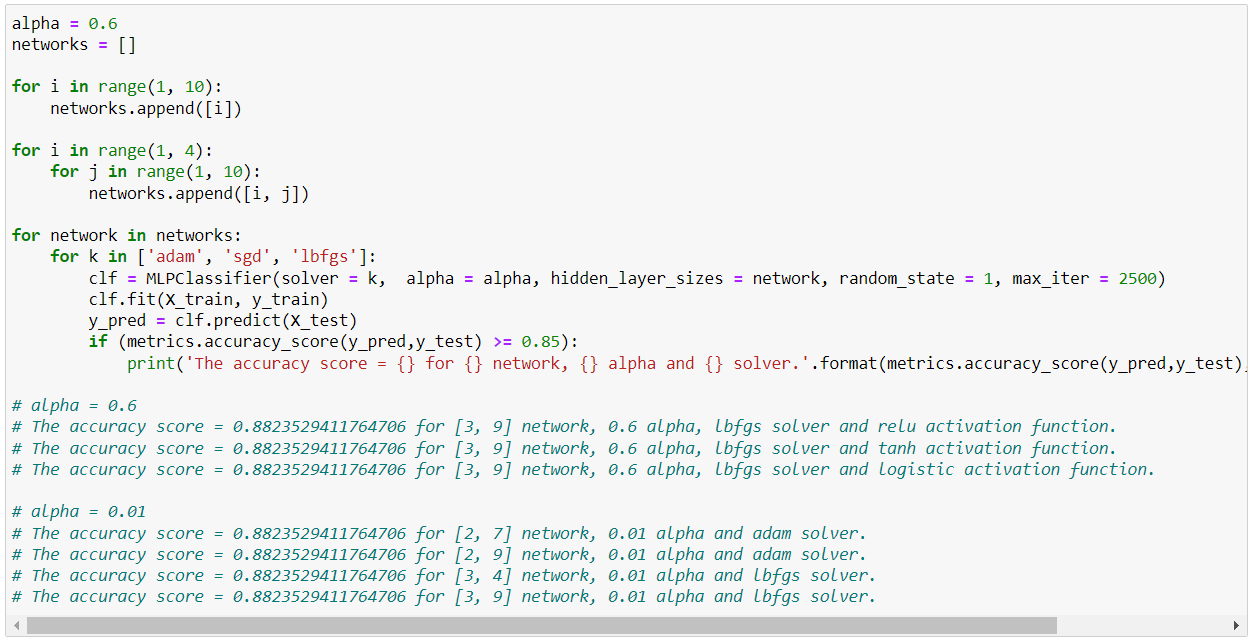
### 4.4.1 Splitting the dataset



### 4.4.2 Tuning parameters for the model

The goal of this part of the process is to find the best parameters for the multi-layer perceptron classifier so it can predict the best possible output data based on input data (number of layers and neurons, alpha value etc.)

To do that, we used for a couple first tests a set of loops that go through some values of different parameters that the model can use.

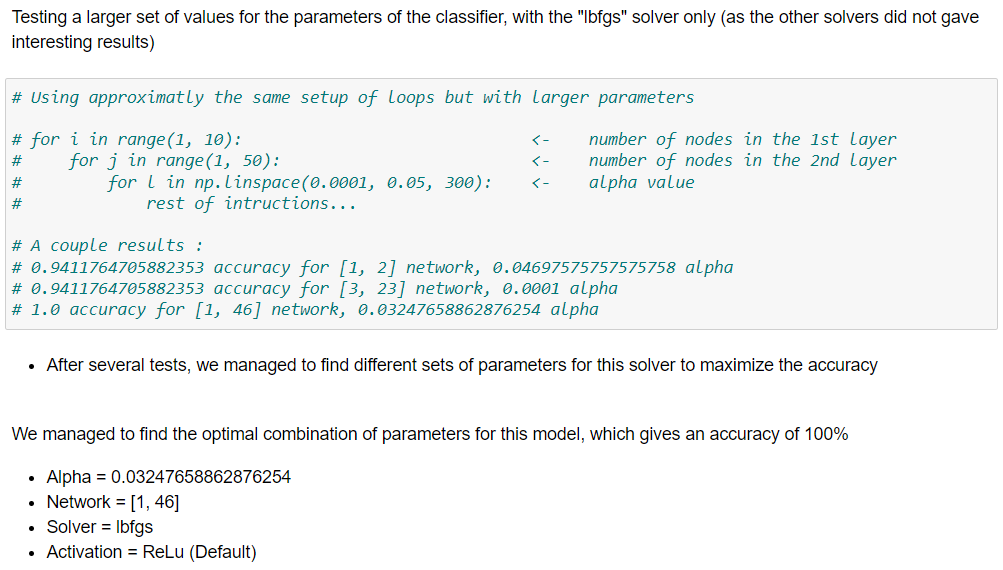
****

After several tests with this method, we noticed after that :

* We can use “relu” activation function as a parameter for the classifier for the remaining tests.
* No matter the alpha value chosen manually, we can reach a maximum accuracy value of 94.11%, we must add a loop to test a larger set of alphas to try to increase the accuracy.

We must keep testing all solvers with more precise parameters (alpha, network, but the same “relu” activation function).

After several tests, we noticed that the “adam” and “sgd” solvers could not give good results no matter the range of parameters we fitted into the “for” loops.



**GridSearchCV:**

We tried to optimize the model parameters using the GridSerachCV algorithm.

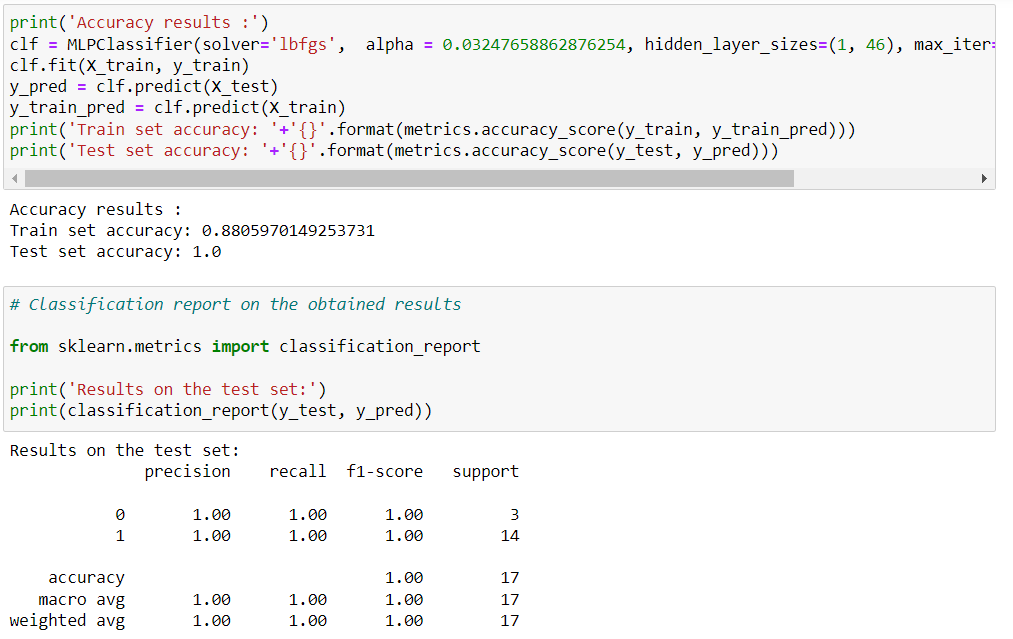
In brief, it does the process of performing parameter tuning in order to determine the optimal values for a given model, as we pass predefined values for parameters to the GridSearchCV function by defining a dictionary in which we mention a particular parameter along with the values it can take

We initialized the searching algorithm by giving it the model we’re trying to optimize, a parameter space that contains different values for the different parameters that the model can have, and other parameters. After the search, the algorithm returned the best parameters it could find.

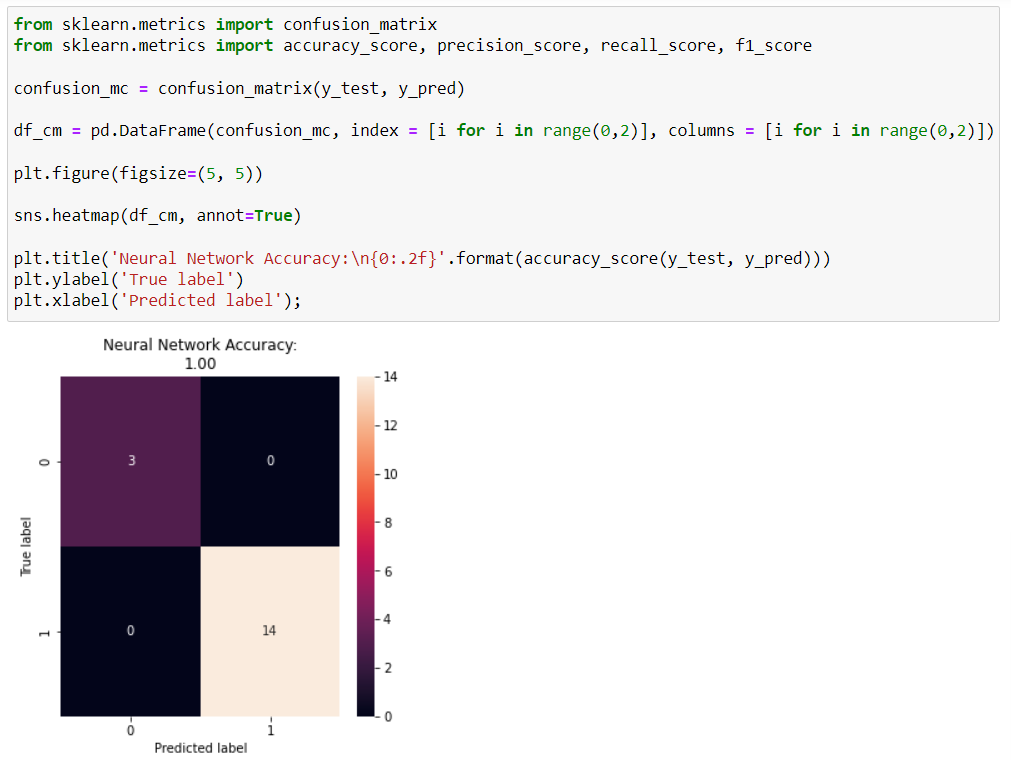
We concluded that using this algorithm was not an interesting way to find the optimal parameters for our model, as the accuracy produced with the best parameters the algorithm could find fitted in the mlp classifier is lower as the accuracy we found with the method above.

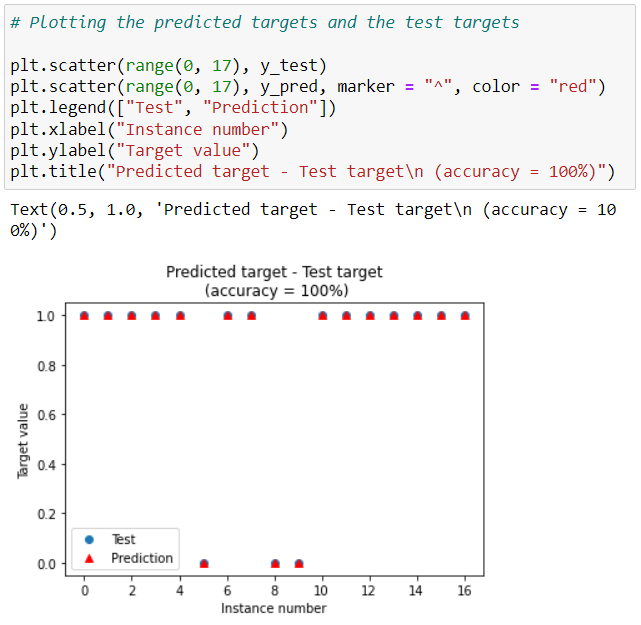
* All the details of this part are to be found in the annex A.1.

### 4.4.3 Final Results



### 4.4.4 Confusion Matrix - Plotting the results

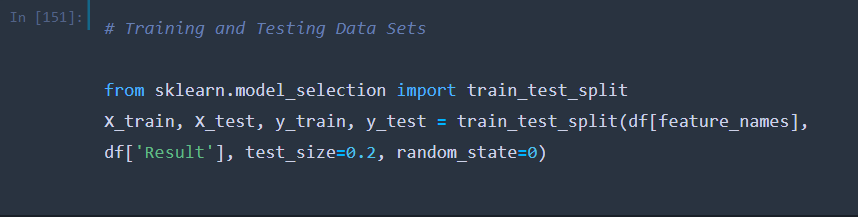




## 4.5 SVM

## 4.6 Decision Trees

### 4.6.1 Train and Test Datasets

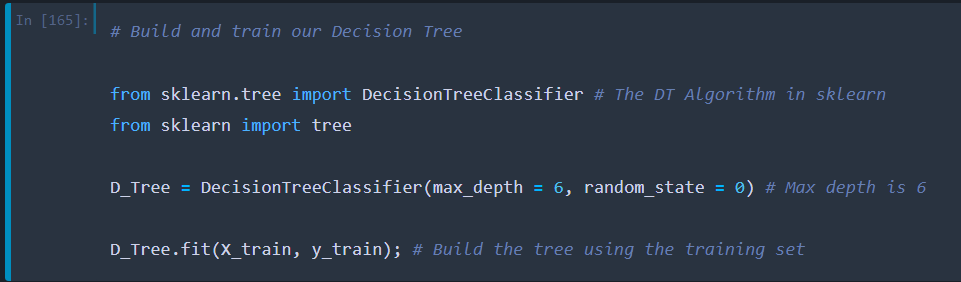


After trying many test-sizes, 0.2 gave the best test accuracy:

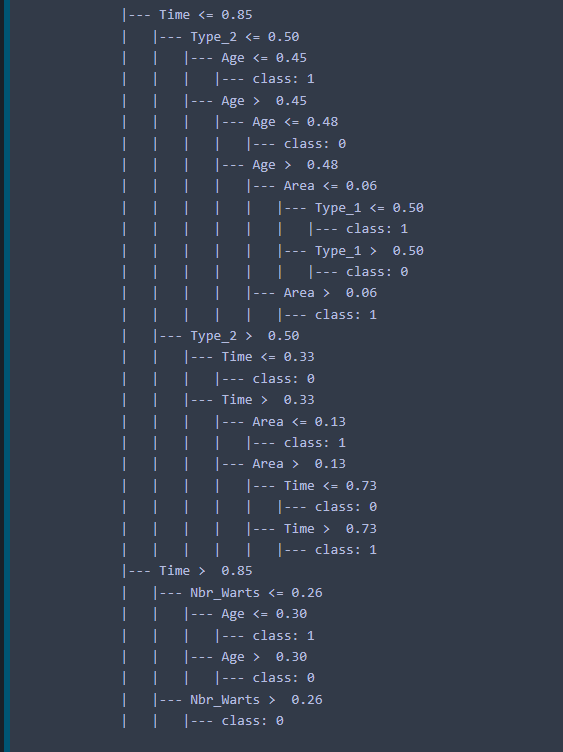
|  |  |  |
| --- | --- | --- |
| **Test Size** | **Train Accuracy** | **Test Accuracy** |
| 0.1 | 98.67 % | 88.89 % |
| 0.15 | 98.59 % | 92.31 % |
| 0.2 | 100 % | 94.12 % |
| 0.25 | 100 % | 80.95 % |
| 0.3 | 98.28 % | 92.31 % |
| 0.35 | 98.15 % | 93.33 % |
| 0.4 | 100 % | 91.18 % |

### 4.6.2 Building the Decision Tree

We found that the best max\_depth is 6

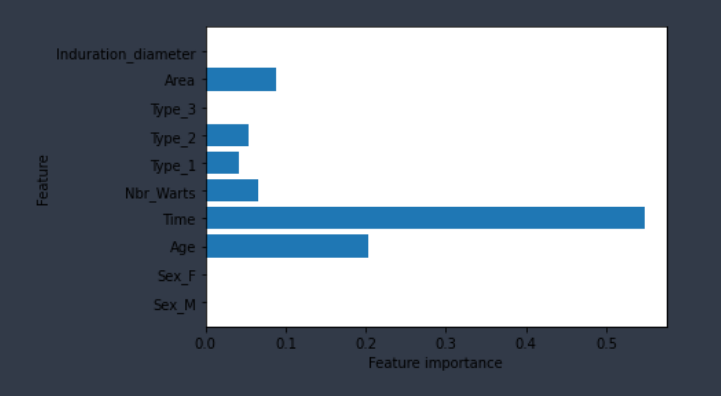


### 4.6.3 Text representation of the decision tree



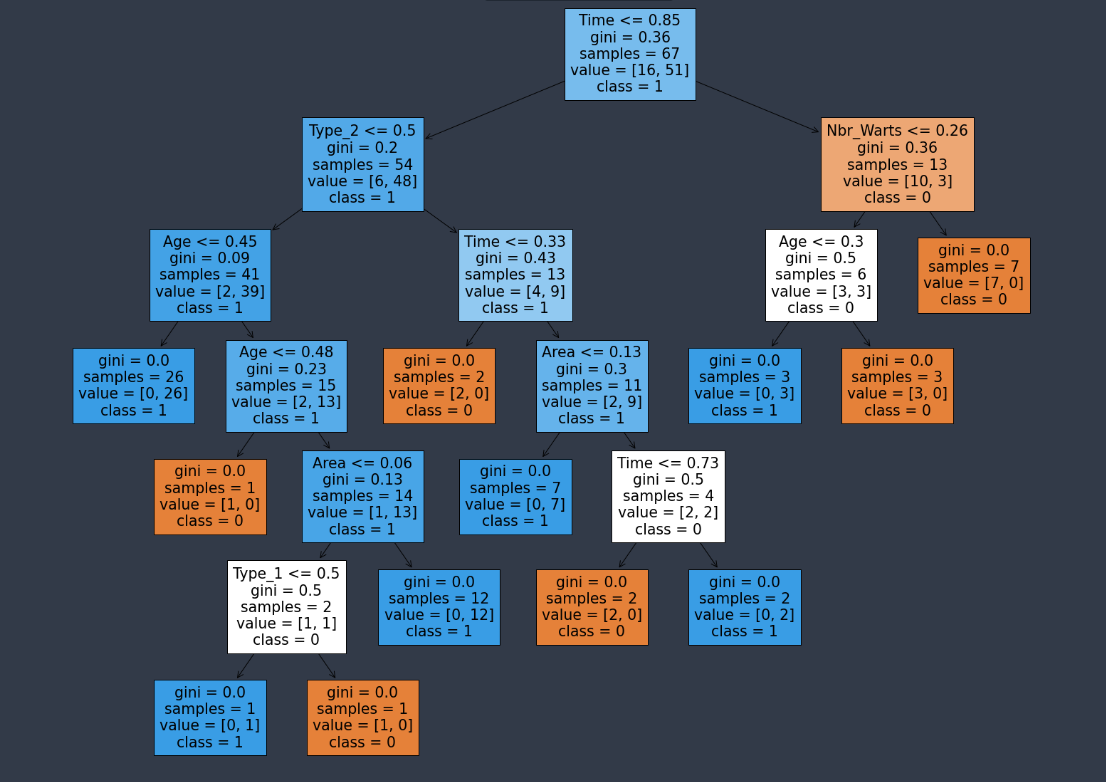
### 4.6.4 Feature importance in the Decision Tree:

We used D\_Tree.feature\_importances and plt.barh() to create this graph :



We notice that ‘Time’,’Age’ and ‘Area’ are the most important features, while ‘Sex\_F’, ‘Sex\_M’ and ‘Type\_3’ are not used.

### 4.6.5 Graphic representation:

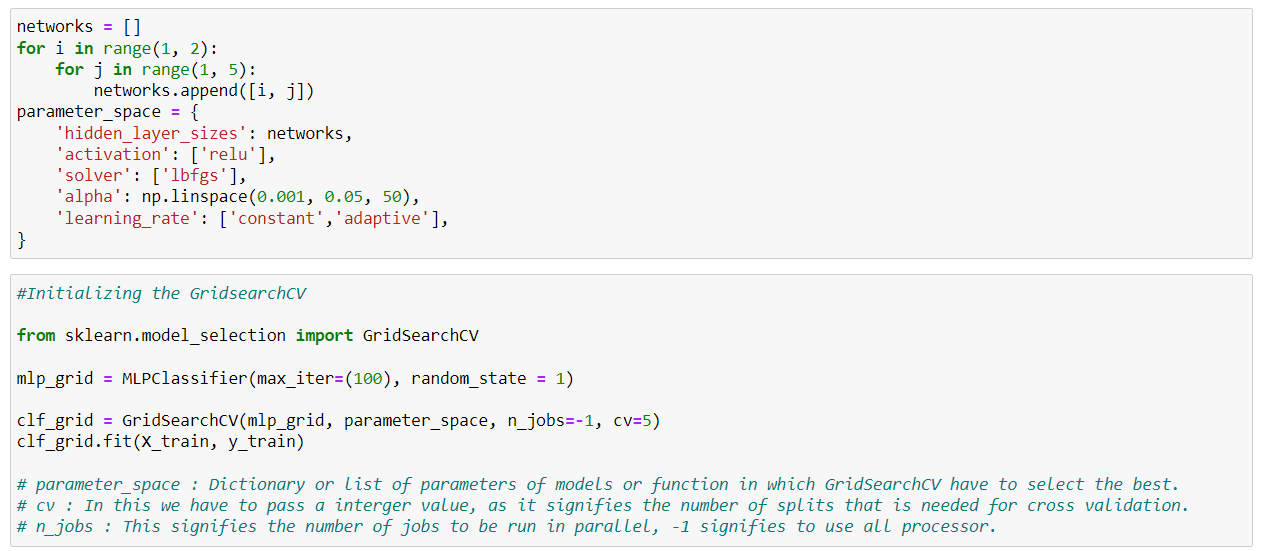


# Annex

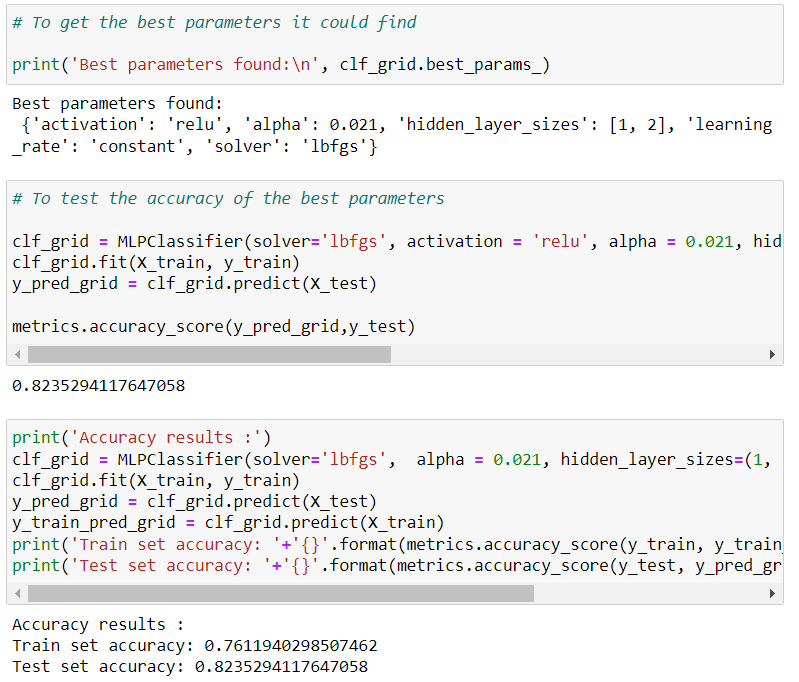
## A.1 The GridSearchCV implementation

### A.1.1 Initializing the algorithm

As mentioned above, we have to pass predefined values for parameters to the GridSearchCV function by defining a dictionary in which we mention a particular parameter along with the values it can take.



### A.1.2 The final results



Finally, we notice that using GridSearchCV to find the best parameters for the classifier is not the optimal way to do it, doing it with for loops seems to be better, in our case at least.

This limitation is due to the fact that the search can only test the parameters that we fed into, and giving a large set of values in each parameter can produce a really high time-consuming search.