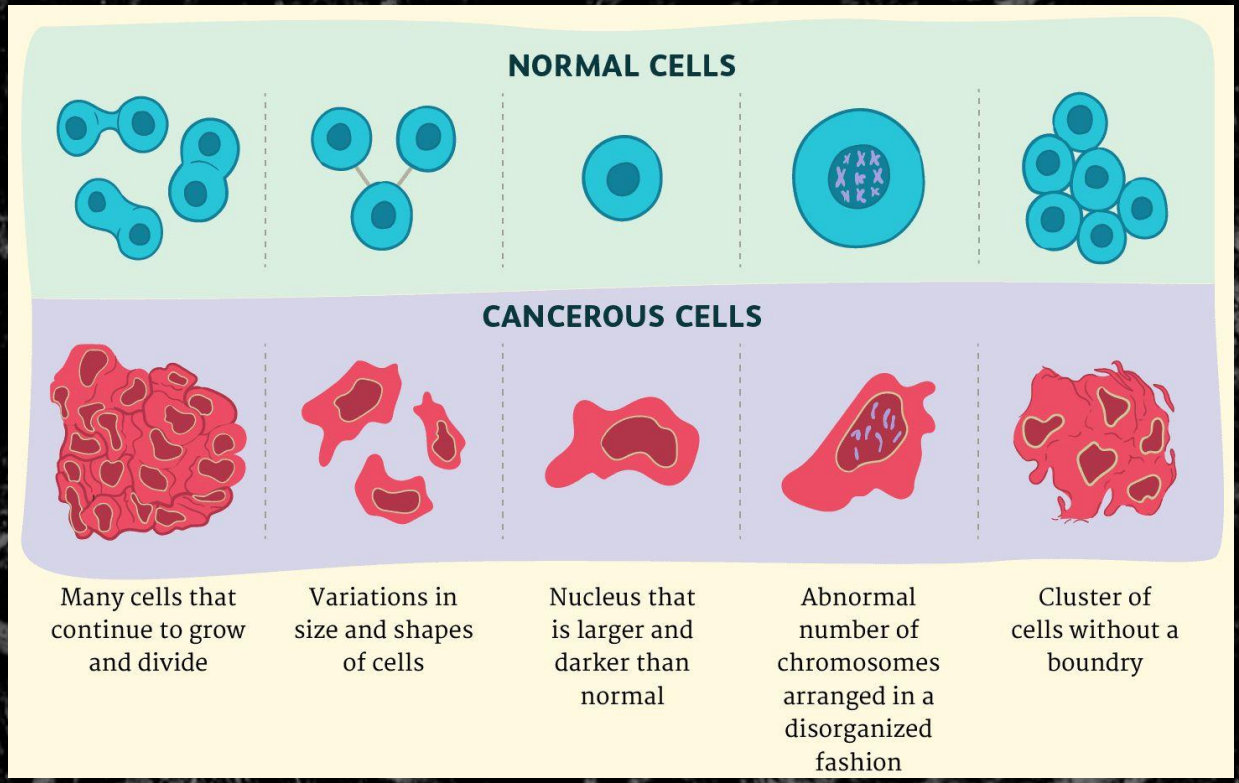
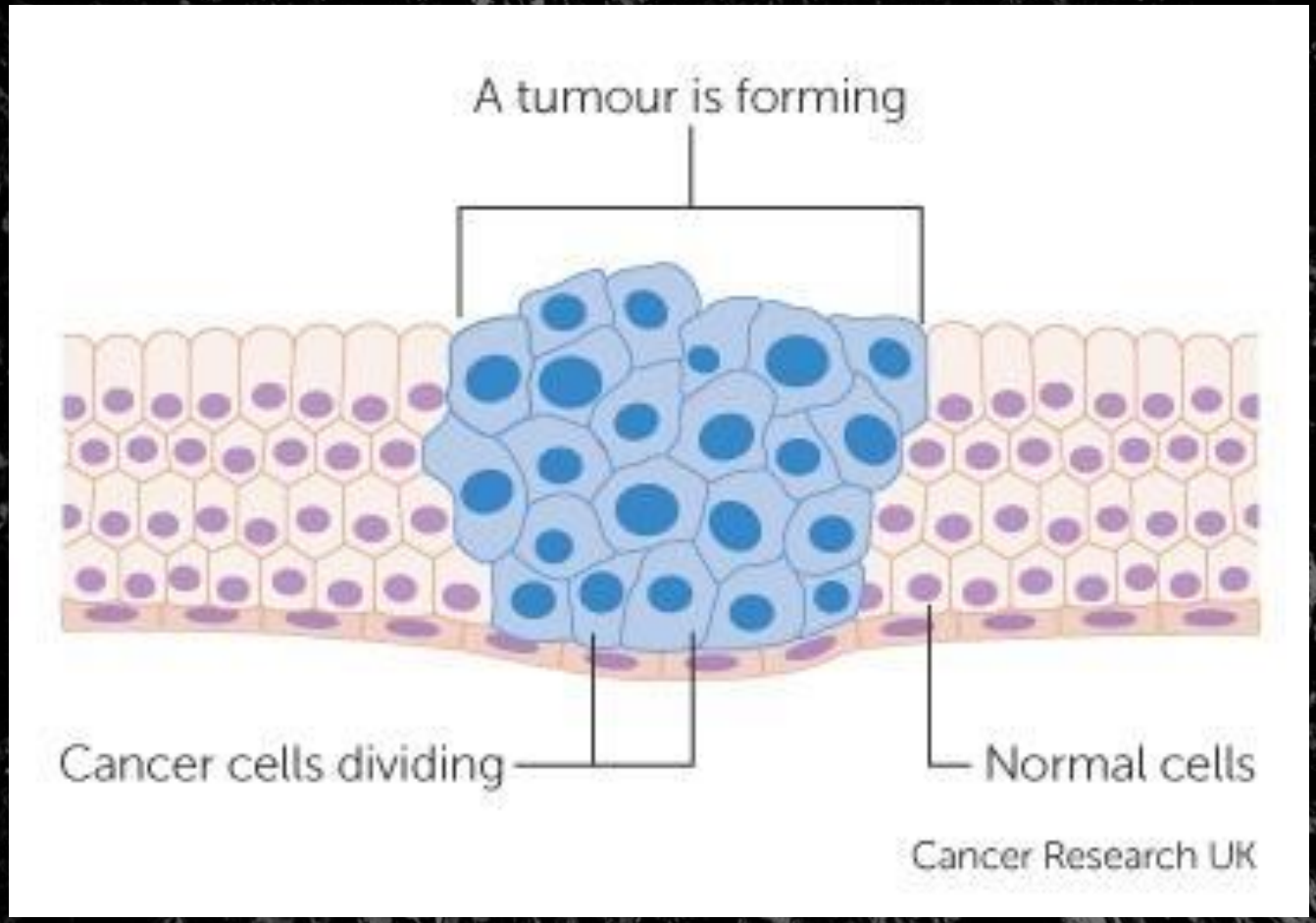

HADACA project
Group FACTOR

What is this project about?

- About 90.5 million people have cancer.
- Only in 2017, 400.000 new types of cancer were discovered.
- Unfortunately, even in age of technologies, there is still no absolute cure for it.



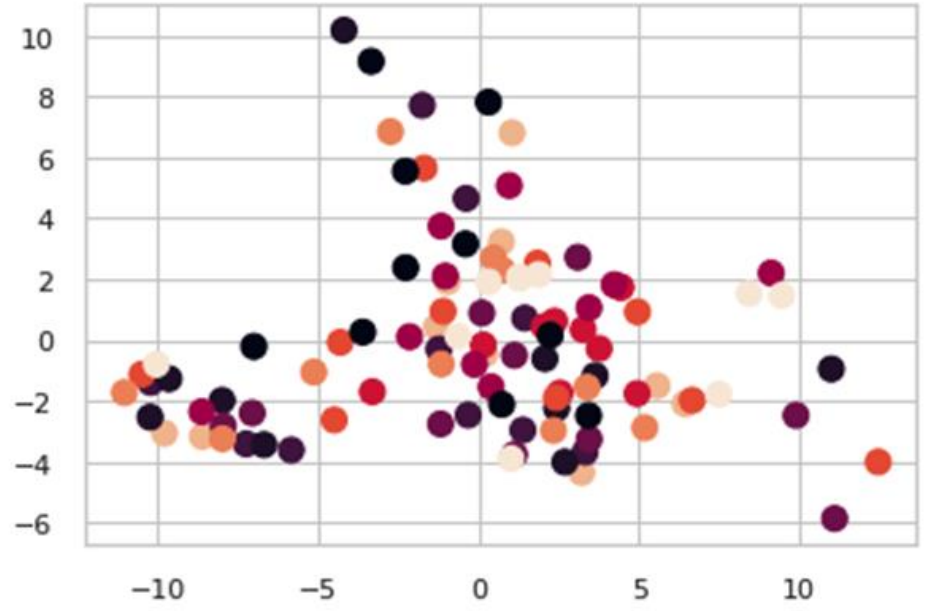
This project is a way to help the research advance and fight the scourge of 21st century.



- We will use Machine Learning to automate the process of diagnosing the tumors on different stages and make it easier to define the exact stage of cancer.

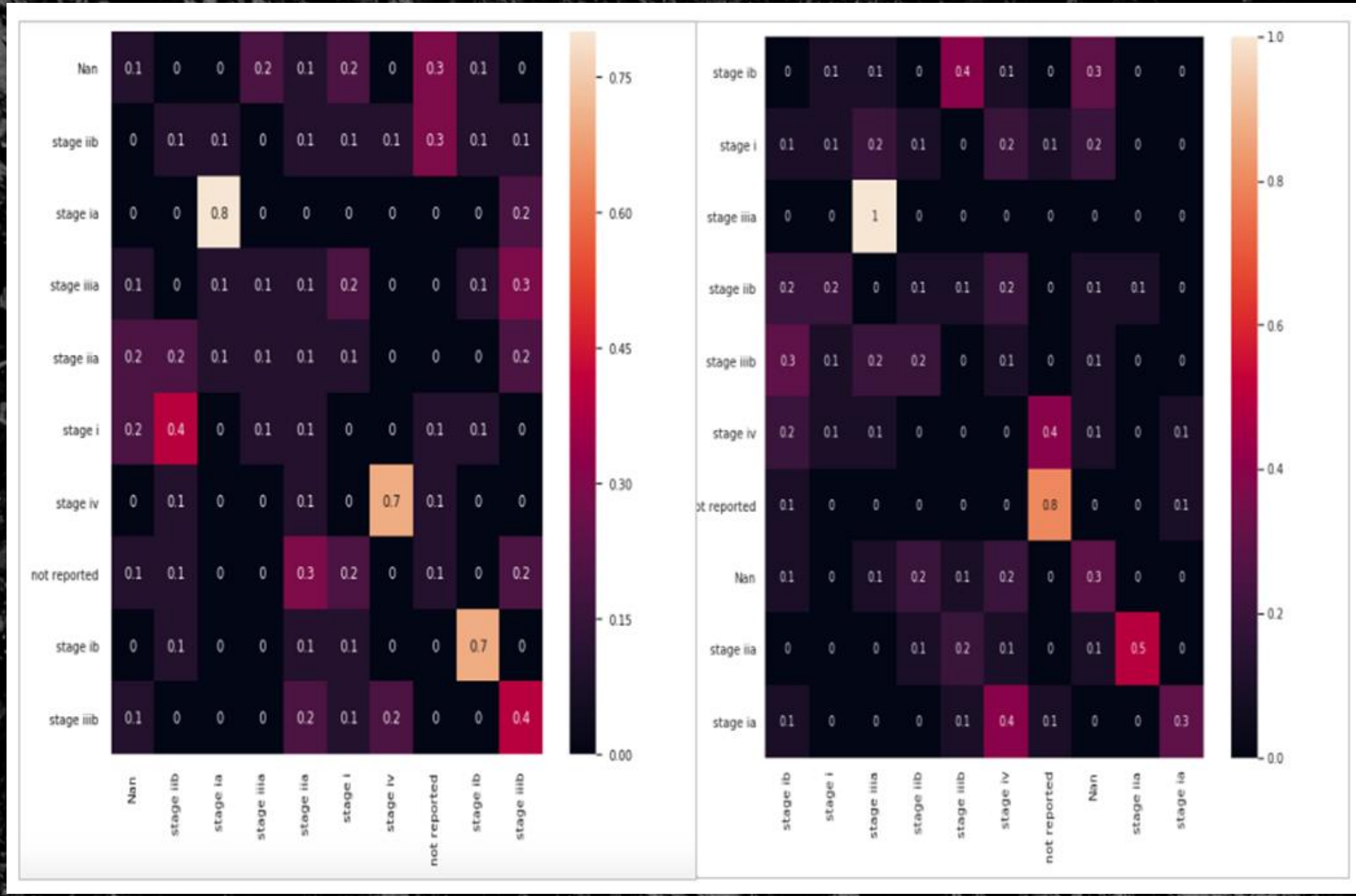
- We have data based on dataset of 5000 real patients with 10 different types of cancer.

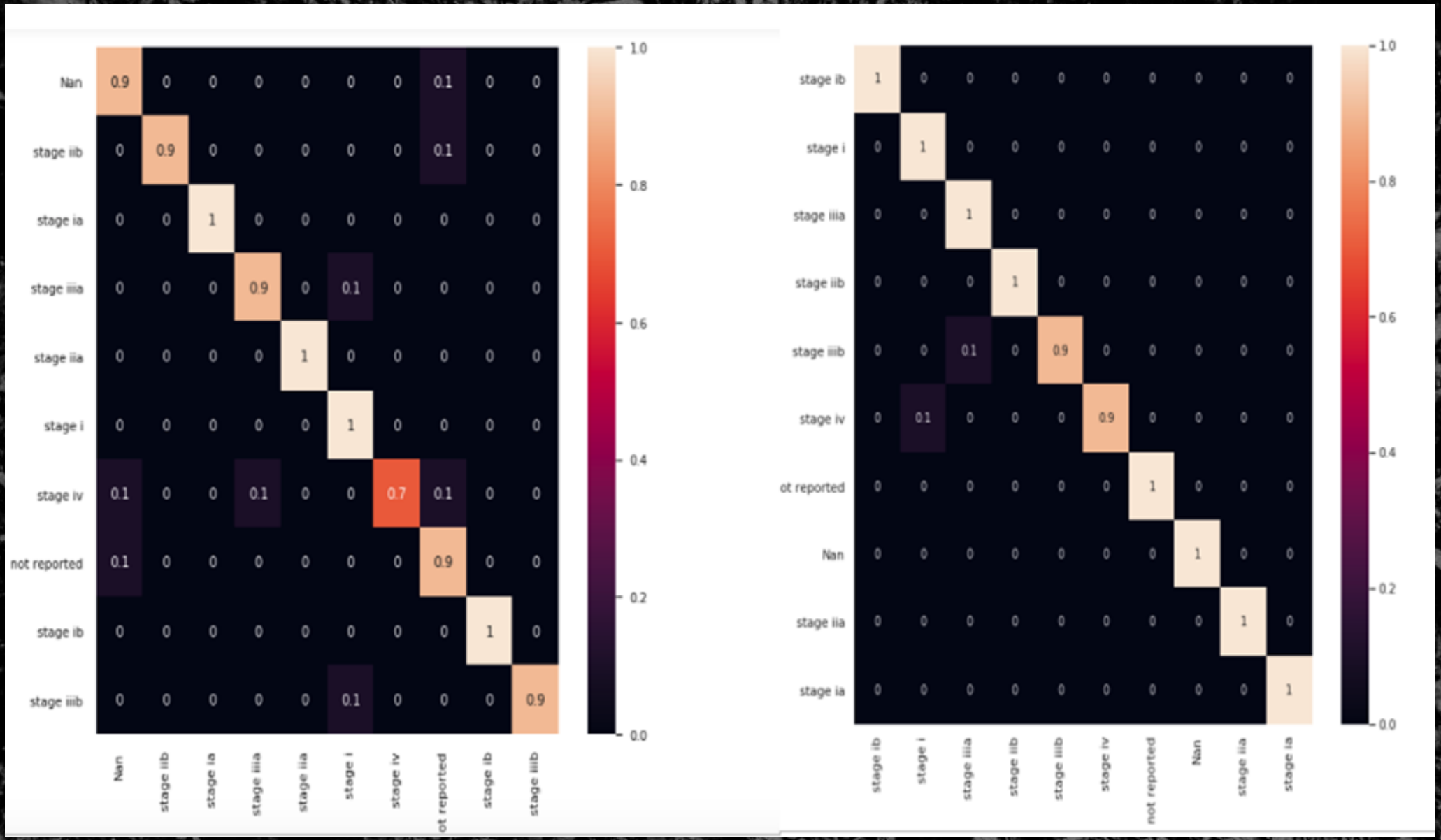
```
Number of examples = 100
Number of features = 5000
           Class
0       stage ib
1       stage ia
2        stage i
3      stage iib
4       stage iv
5     stage iiia
6   not reported
7      stage iia
8             Nan
9     stage iiib
Number of classes = 10
```



Confusion matrix

- The confusion matrix is, in the terminology of supervised learning, a tool for measuring the quality of a classification system. One of the interests of the confusion matrix is that it quickly indicates whether a classification system is able to classify correctly.


Nan 0.1 0 0 0.2 0.1 0.2 0 0.3 0.1 0
stage iib 0 0.1 0.1 0 0.1 0.1 0.1 0.3 0.1 0.1
stage ia 0 0 0.8 0 0 0 0 0 0 0.2
stage iiia 0.1 0 0.1 0.1 0.1 0.2 0 0 0.1 0.3
stage iia 0.2 0.2 0.1 0.1 0.1 0.1 0 0 0 0.2
stage i 0.2 0.4 0 0.1 0.1 0 0 0.1 0.1 0
stage iv 0 0.1 0 0 0.1 0 0.7 0.1 0 0
not reported 0.1 0.1 0 0 0.3 0.2 0 0.1 0 0.2
stage ib 0 0.1 0 0 0.1 0.1 0 0 0.7 0
stage iiib 0.1 0 0 0 0.2 0.1 0.2 0 0 0.4
0.75
0.60
0.45
0.30
0.15
0.00
Nan
stage iib
stage ia
stage iiia
stage iia
stage i
stage iv
not reported
stage ib
stage iiib
stage ib 0 0.1 0.1 0 0.4 0.1 0 0.3 0 0
stage i 0.1 0.1 0.2 0.1 0 0.2 0.1 0.2 0 0
stage iiia 0 0 1 0 0 0 0 0 0 0
stage iib 0.2 0.2 0 0.1 0.1 0.2 0 0.1 0.1 0
stage iiib 0.3 0.1 0.2 0.2 0 0.1 0 0.1 0 0
stage iv 0.2 0.1 0.1 0 0 0 0.4 0.1 0 0.1
ot reported 0.1 0 0 0 0 0 0.8 0 0 0.1
Nan 0.1 0 0.1 0.2 0.1 0.2 0 0.3 0 0
stage iia 0 0 0 0.1 0.2 0.1 0 0.1 0.5 0
stage ia 0.1 0 0 0 0.1 0.4 0.1 0 0 0.3
1.0
0.8
0.6
0.4
0.2
0.0
stage ib
stage i
stage iiia
stage iib
stage iiib
stage iv
not reported
Nan
stage iia
stage ia


Nan 0.9 0 0 0 0 0 0 0.1 0 0
stage iib 0 0.9 0 0 0 0 0 0.1 0 0
stage ia 0 0 1 0 0 0 0 0 0 0
stage iiia 0 0 0 0.9 0 0.1 0 0 0 0
stage iia 0 0 0 0 1 0 0 0 0 0
stage i 0 0 0 0 0 1 0 0 0 0
stage iv 0.1 0 0 0.1 0 0 0.7 0.1 0 0
not reported 0.1 0 0 0 0 0 0 0.9 0 0
stage ib 0 0 0 0 0 0 0 0 1 0
stage iiib 0 0 0 0 0 0.1 0 0 0 0.9
Nan
stage iib
stage ia
stage iiia
stage iia
stage i
stage iv
ot reported
stage ib
stage iiib
1.0
0.8
0.6
0.4
0.2
0.0
stage ib 1 0 0 0 0 0 0 0 0 0
stage i 0 1 0 0 0 0 0 0 0 0
stage iiia 0 0 1 0 0 0 0 0 0 0
stage iib 0 0 0 1 0 0 0 0 0 0
stage iiib 0 0 0.1 0 0.9 0 0 0 0 0
stage iv 0 0.1 0 0 0 0.9 0 0 0 0
ot reported 0 0 0 0 0 0 1 0 0 0
Nan 0 0 0 0 0 0 0 1 0 0
stage iia 0 0 0 0 0 0 0 0 1 0
stage ia 0 0 0 0 0 0 0 0 0 1
stage ib
stage i
stage iiia
stage iib
stage iiib
stage iv
not reported
Nan
stage iia
stage ia
1.0
0.8
0.6
0.4
0.2
0.0

Preprocessing

- Preprocessing is all about sorting and selecting accurate data to optimize the process of classification and to make it as easy and fast as possible.

```
import pandas as pd                                          // Importing the librairies we need
From data.io import read as df

Data = read_as_df('data_dir + '/' + 'data_name')             // Importing the dataset

X = dataset.iloc[:, :-1].values                              // Create a matrix of features in our dataset

From sklearn.preprocessing import Imputer                    // Taking care of missing data
Imputer = Imputer(missing_values = "NaN",                    // With the class called imputer in sklearn.preprocessing
          strategy = "mean", axis = 0)                           we will search for missing data
Imputer = imputer.fit(X[:,1:3])
X[:, 1:3] = imputer.transform(X[:, 1:3])                     // Now we will just replace the missing values with
                                                                the mean of the column by the method transform.


from sklearn.model_selection import train_test_split        // We separe the data into three datasets, for training,
X_train, X_test, Y_train, Y_test =                              testing and validation.
           train_test_split(X,Y, test_size=0.2)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()

X_train = sc_X.fit_transform(X_train)                        // We applied a feature scaling to the data :
X_test = sc_X.transform(X_test)                                 It is a method used to standardize the range of
                                                                independent variables or features of data.
```

Classification

- There are many ways to carry on the classification of data. One of them is to use the perceptron. We used the multiclass perceptron.

Algorithm 5 PERCEPTRONTRAIN($\mathbf{D}$, *MaxIter*)

```
w_d ← 0, for all d = 1 ... D                          // initialize weights
b ← 0                                                 // initialize bias
for iter = 1 ... MaxIter do
    for all (x,y) ∈ D do
        a ← Σ_{d=1}^{D} w_d x_d + b                   // compute activation for this example
        if ya ≤ 0 then
            w_d ← w_d + y x_d, for all d = 1 ... D    // update weights
            b ← b + y                                 // update bias
        end if
    end for
end for
return w_0, w_1, ..., w_D, b
```

Algorithm 6 PERCEPTRONTEST($w_0, w_1, \ldots, w_D, b, \hat{\mathbf{x}}$)

```
a ← Σ_{d=1}^{D} w_d x̂_d + b                           // compute activation for the test example
return SIGN(a)
```

Conclusion: (Results)

Table 2 : Preliminary Results

Dataset	Base Estimator (starting kit)	Naive Bayes Classifier	SVM	Decision Tree	Random Forest	Perceptron
Training	0.8991	0.3860	0.8994	0.8996	0.9980	0.2100
Cross-Validation	0.6872	0.8233	0.834	0.8677	0.8666	0.6581
Validation	0.41	0.49	0.42	0.43	0.39	0.67

RESULTS

#	User	Entries	Date of Last Entry	Prediction score ▲	Duration ▲	Detailed Results
1	luc.gibaud	4	01/24/19	0.9843 (1)	0.00 (1)	View
2	Malikkazi	3	01/24/19	0.9804 (2)	0.00 (1)	View
3	takfarinas.nait-larbi	5	03/22/19	0.9790 (3)	0.00 (1)	View
4	doctor	25	03/22/19	0.9790 (3)	0.00 (1)	View
5	martin.bauw	32	01/24/19	0.9721 (4)	0.00 (1)	View
6	Cancer	23	03/24/19	0.9506 (5)	0.00 (1)	View
7	HEALTH	11	03/24/19	0.8776 (6)	0.00 (1)	View
8	Cure	19	03/24/19	0.5932 (7)	0.00 (1)	View
9	Zhengying	4	02/08/19	0.3589 (8)	0.00 (1)	View
10	FACTOR	8	03/26/19	0.2399 (9)	0.00 (1)	View
11	OmarAbdoulayeBADIANE	2	03/26/19	0.2399 (9)	0.00 (1)	View