# REPORT INFO232

**Team :** SAHARA
**Team members :** Guillaume ABADIE < guillaume.abadie@u-psud.fr > (Group 1), Louise ALLAIN < louise.allain@u-psud.fr > (Group 1), Paul CATILLON < paul.catillon@u-psud.fr > (Group 1), Alexandre CIORASCU < alexandre.ciorascu@u-psud.fr > (Group 1), Mohamed-Fouad FELLAH < mohamed-fouad.fellah@u-psud.fr > (Group 2), Nelly LAM < nelly.lam@u-psud.fr > (Group 4)
**Codalab challenge :** ¡https://codalab.lri.fr/competitions/625#learn_the_details¿
**Last submission :** 21
**GitHub :** ¡https://github.com/LouiseALLAIN/SAHARA/tree/master/starting_kit¿

## Introduction

Malaria is one of the most considerable and impactful parasitic disease that mostly affect children and pregnant women in sub-Saharan Africa and India. We chose this project because of the actual need to prevent the spread of the disease. The goal of the classifier is to distinguish the patients harmed by malaria of the other who are not by analysing images of stings.

Medichal is a project involving a classification problem. We have to determine if the cells are infected or not, based on their pictures provided by data science students of Paris-Saclay University. In order to classify our cells, we already have several features.

In our whole project, we used the AUC scoring method. In fact, this method allows us to estimate the rate of false negatives that we are trying to minimize and our aim is to prevent sick people from being diagnosed as such.

## Description

### Preprocessing

At the beginning, the preprocessing duo wanted to "simplify" the dataset by eliminating useless features, reducing the data dimension and detecting/removing potential outliers. These tasks would have helped our model go faster and get better results.

First of all, we deleted some features because we did not want the other steps of the preprocessing to take too much time due to the useless data. We analyzed boxplots of features independently to decide whether they are useful or not. A feature is considered as useless if there isn't any correlation between the value of the feature are the infection of cell. In this case, the feature doesn't help detecting which cells are infected, and thus we can remove it.

After that, we wanted to reduce the number of dimension for the same reasons as the feature selection. We used PCA algorithm from the scikit-learn library.

Finally, we had to remove outliers, which are observations that are far from the others. This step is important because we want our model to fit the region where the training data is the most concentrated, in order not to be disturb by extreme values and thus have better results with true data. For that, we used the method "Local outlier factor" from scikit-learn, which calculates the anomaly score of each sample, measuring how isolated it is with respect to the surrounding neighborhood. We compare the local density of each sample to the local densities of its neighbors, and the ones which had a lower comparative density (beyond a certain threshold) were considered outliers, so we can remove them.

But at this point, we realized that the kind of preprocessing we did, consisted in reducing our dataset didn't fit our challenge well. In fact, the goal of our challenge is to detect infected people, and we don't really care about the time it takes. However, we really need a good score to avoid diagnostic errors, and for this, we have to train our model on a more important dataset. Thus, we gave up all the methods we tried until this moment and started working on algorithms that enabled us to expand our dataset.

First of all, we tried the method PolynomialFeatures from scikit.learn. The PolynomialFeatures function adds complexity to the model by considering nonlinear features. It generates new features that are the polynomial combinations of the features. We specify the degree of the polynomial features and then all the new combinations we get are of a degree less than or equal to it. However, our results weren't very concluding so we moved to another method, more useful this time : the K-means method. This method is often used in data classification but we use it here in the preprocessing part. We re-organize our dataset which results in a simplification of our methods. We do not simplify the number of features but how they are organized. It is easier for the model group to get better results with a dataset made with "easy, well organized" features.

## Model

We decided to use different scikit-learn classification models and GridSearchCV to adapt the hyper-parameters. We started to look for the best hyper-parameters for each model. Then, we compared the results using cross-validation to avoid the phenomenon of over-learning. The main difficulties we encountered, were : the hyperparameters understanding and the ≪ not too long data training ≫ models search. At the end, we compared the scores obtained thank to the AUC scoring method on our differents models.

## Visualization

To represent the dataset, we decided to use models which derive from Manifold Learning, (we took some of our sources of scikit-learn in particular to study the learning scores).We tried to choose the most relevant method, in relation to our subject, here we chose to mainly carry out a reduction of dimension to see the raw results.

We started to study populations, using research methods (Neighbors Search), with the LLE method (Locally Linear Embedding) which comes directly from Manifold Learning. This method searches for a lower dimension projection of the data which preserves the distances in the local "neighborhoods". We can consider this as a series of analyze of local principal components which are globally compared to find the best nonlinear incorporation.This method is very practical mainly for its relevance , in fact it makes it possible to analyze the data on small samples and thus obtain a kind of average which makes it possible to visualize the results.
Then we used an estimator of Random Forest which adjusts a number of decision tree classifiers on various subsamples among all of our data (cf. RFC - Figure 5). This method combines the result of multiple predictions which aggregates many decision trees (RandomForestClassifier). We chose this method because it allowed us to obtain more precise prediction results according to our estimates.

Next, we used a representation of training score and validation scores of an SVM in order to study the learning scores and validation scores of an SVM for different values of the $\gamma$ parameter of the kernel. This method was chosen because support vector machines (SVM) are a set of supervised learning methods which have several advantages, in particular the fact that it is a very effective method in large spaces. . In addition, this method uses a subset of learning points in the decision function (called support vectors), so it is also efficient in memory. Then we made the observations that for very low $\gamma$ values, we see that the training score and the validation score are low. This is called under-adjustment. Average $\gamma$ values will result in high values for both scores, this shows that the classifier is working quite well. On the other hand, if the $\gamma$ is too high, the classifier will be overloaded, so the training score will be good except that the validation score will be bad.(cf.Figure 4).

We have reused the graph representing the plot of the ROC curve, this plot is a good indicator for us to find a good threshold for our model, indeed if we really want something close to 100% of true positives, we could consider a model giving us 40% false positive rate, so we could consider this threshold as a medical diagnosis.

Finally, we decided to put a 2d plot of our data (Figure 2). Indeed this indicator allows us, in the visualization part to better visualize the data that we have to process, it serves like the ROC curve, to best estimate the number of false positives and the number of true negatives. This allows us to measure both the model error on the training data and on the test data, the parameter of our model is then adjusted to minimize the test error. In addition, it's an agnostic indicator of data, indeed it doesn't depend on the threshold, that is to say that this indicator doesn't care about the manner in which the data which it receives were sent to it.

## Results

We observed that the most efficient model was the random forest, which obtained a score on cross-validation around 0.9553. GridSearchCV provided us the best value for some hyperparameters as the max depth of trees, the number of trees and the random_state. We fixed these different parameters at 100, 100 and 30 respectively. We also tried other intervals for our values by making vary the number of trees between 5 and 50 and the number of trees between 500 and 2000 but the results weren't better. A comparison of the results obtained without having previously set the hyper-parameters, with the results obtained by having adjusted them, showed us the importance of the hyper-parameters on the score of the algorithm in cross-validation : in particular for the decision tree, the naive Bayes method and the gradient descent. However, our tests on the RAW version of the project turned out to be much less successful, with scores of 0.7, at best.

Then, we applied our preprocessed method (K-means cluster) on our data, we made several submissions on Codalab in order to find the best number of clusters and finally fixed it at 10. It enabled us to improve our score by 0.5 points, reaching to 0.9558.

## Conclusion and Discussions

Finally, we could explore how to solve data science problems with Python. We thus discovered that the most effective classification algorithm on our problem, was the random forest. Thanks to the project MEDICHAL, combining medicine and IT, and requiring group work, we understand, now, why data science is a high potential field.

# FIGURES PAGE

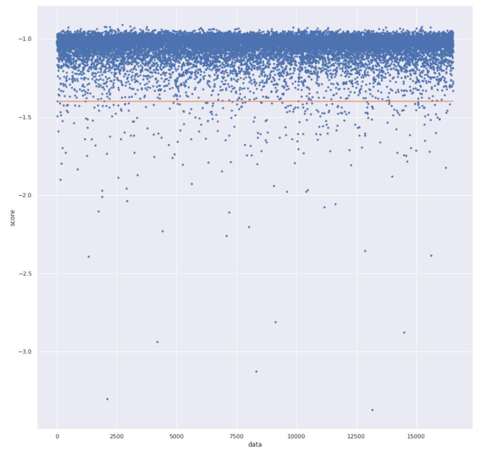


*Figure 1 : Outlier detection with anomaly score method*

In Figure 1, each sample is represented by a point, whose ordinate is his anomaly score. The lower a point, the lower its score is. We decided to fix the threshold at 1.4, it is represented with the red line. Samples that are below the red line have a score too low, they are too far from the others so they are considered outliers : we have to remove them.

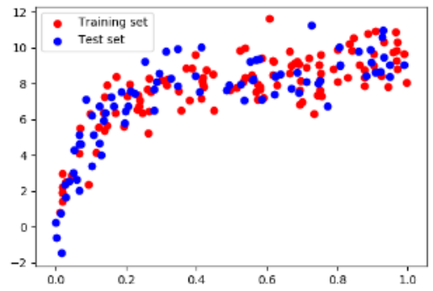


*Figure 2 : 2D plot of datas*

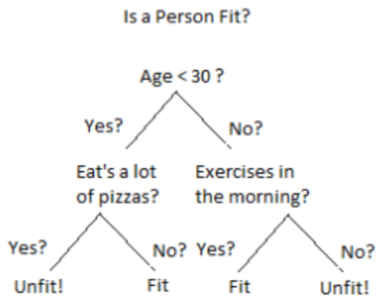


*Figure 3 : Example of Decision Tree*

A decision tree model is a non-parametric supervised learning method used for classification and regression. The series of questions and their possible answers can be organized in the form of a decision tree, which is a hierarchical structure consisting of nodes and directed edges.[...]The tree has three types of nodes :

— A root node that has no incoming edges and zero or more outgoing edges.

— Internal nodes, each of which has exactly one incoming edge and two or more outgoing edges.

— Leaf or terminal nodes, each of which has exactly one incoming edge and no outgoing edges.

In a decision tree, each leaf node is assigned a class label. The non- terminal nodes, which include the root and other internal nodes, contain attribute test conditions to separate records that have different characteristics.

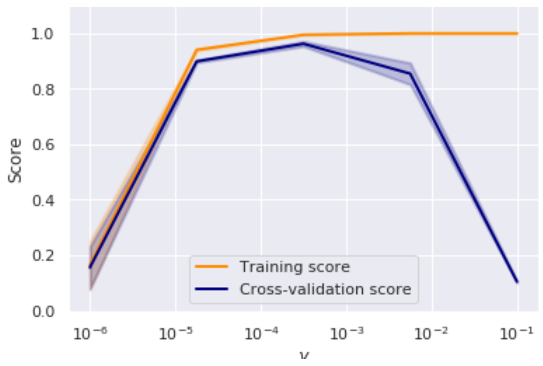


*Figure 4 : Validation Curve with SVM [2]*

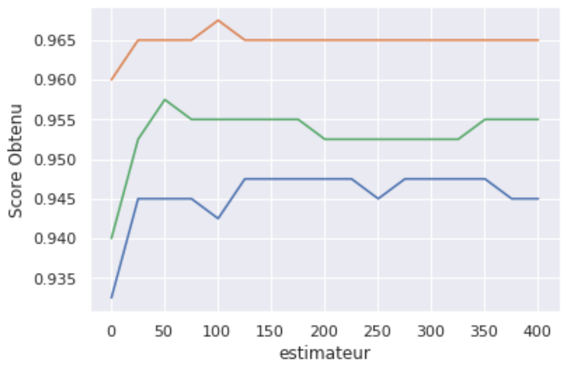
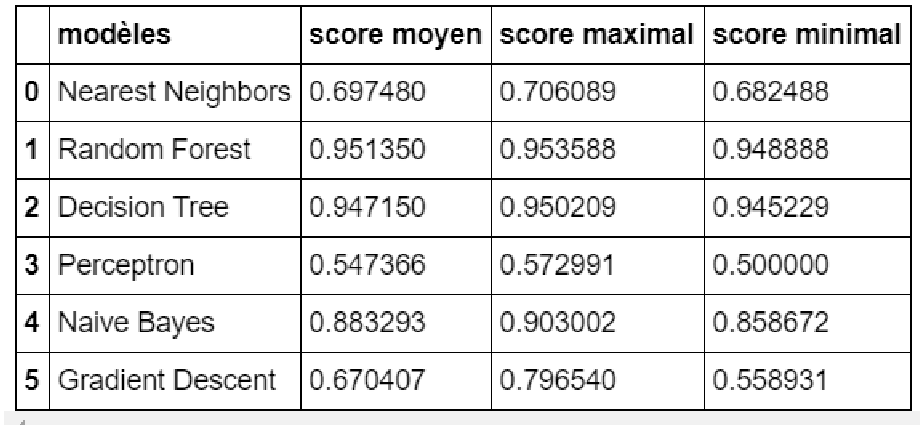


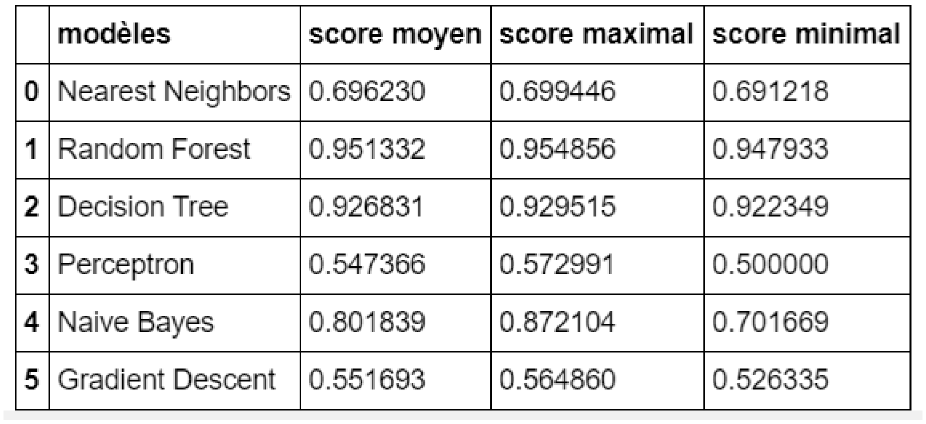
*Figure 5 : Meta-estimator (Method of RandomForrestClassifier) [2]*

## BONUS PAGE



| | modèles | score moyen | score maximal | score minimal |
|---|---|---|---|---|
| 0 | Nearest Neighbors | 0.697480 | 0.706089 | 0.682488 |
| 1 | Random Forest | 0.951350 | 0.953588 | 0.948888 |
| 2 | Decision Tree | 0.947150 | 0.950209 | 0.945229 |
| 3 | Perceptron | 0.547366 | 0.572991 | 0.500000 |
| 4 | Naive Bayes | 0.883293 | 0.903002 | 0.858672 |
| 5 | Gradient Descent | 0.670407 | 0.796540 | 0.558931 |

*Table 1 : Cross Validation score (with AUC metric) without fixing hyperparameters*



| | modèles | score moyen | score maximal | score minimal |
|---|---|---|---|---|
| 0 | Nearest Neighbors | 0.696230 | 0.699446 | 0.691218 |
| 1 | Random Forest | 0.951332 | 0.954856 | 0.947933 |
| 2 | Decision Tree | 0.926831 | 0.929515 | 0.922349 |
| 3 | Perceptron | 0.547366 | 0.572991 | 0.500000 |
| 4 | Naive Bayes | 0.801839 | 0.872104 | 0.701669 |
| 5 | Gradient Descent | 0.551693 | 0.564860 | 0.526335 |

*Table 2 : Cross Validation score (with AUC metric) with better hyperparameters*

In these two tables, we display the average, minimum and maximum scores for each model : first on a model whose hyper-parameters have been fixed and then, on the same models with non-worked hyper-parameters. We observe the incidence of hyper-parameters on the model score. The impact seems particularly strong on the decision tree, the naive Bayes method and the gradient descent, whose scores increase a lot with good hyper-parameters. In addition, we note that certain models, like the perceptron or the descent of gradient, obtain very varied scores with a minimum score around 0.5 and a maximum score around 0.82.

## ALGORITHMS AND REFERENCES PAGE

**Algorithm 1** Random Forest

```
Precondition: A training set $S := (x_1, y_1), \dots, (x_n, y_n)$, features $F$, and number
    of trees in forest $B$.
function RandomForest($S$, $F$)
    $H \leftarrow \emptyset$
    for $i \in 1, \dots, B$ do
        $S^{(i)} \leftarrow$ A bootstrap sample from $S$
        $h_i \leftarrow$ RandomizedTreeLearn($S^{(i)}, F$)
        $H \leftarrow H \cup \{h_i\}$
    end for
    return $H$
end function
function RandomizedTreeLearn($S$, $F$)
    At each node:
        $f \leftarrow$ very small subset of $F$
        Split on best feature in $f$
    return The learned tree
end function
```

The random forests model is an ensemble learning method. It operates by constructing a multitude of decision trees at training time and outputting the average of all the classes. This model is way better than decision trees because it corrects the overfitting problem of this method.

**References :**

[1] Mini Project  L2 Data Science Course : slide by Adrien Pavao
[2] Scikit-Learn : Python library for machine learning
[3] Wikipedia : Ensemble Learning
[4] Matthew N. Bernstein Postdoctoral Fellow at the Morgridge Institute of Research : Random Forests
[5] Vipin Kumar : Classification: Basic Concepts, Decision Trees, and Model Evaluation