Project name : MediChal

Team name : MOSQUITO

Team members :

- Alexandre PHAM
- Clément ELLIKER
- Yacine MOKHTARI
- Lynda ATTOUCHE
- Lilia IZRI
- Anis SMATI

Challenge URL : https://codalab.lri.fr/competitions/654
Github repository : https://github.com/PhamAlexT/MOSQUITO
WebSite : https://sites.google.com/view/medichal-mosquito/

Table of contents

1) Challenge description

This last decade, medical imaging has become one of the most leading and popular fields for AI applications. It consists in using machine learning on analysing and diagnosing the environment of many diseases, in our case "Malaria". We will therefore study one of the most basic cases of this kind of application : the classification of medical images.

Malaria is a serious and sometimes fatal disease spread by mosquitoes. According to WHO (World Health Organization) each year, more than 200 million new cases of the disease are reported killing about 655,000 people, especially in third world countries. This disease affects red blood cells, Our challenge consists in distinguishing between infected and uninfected cells. This is then a simple binary classification problem.

Illness and death from malaria can usually be prevented by analysing the red cells. This is why our project is built around this topic ; make a step forward in modern medicine and ultimately, help save lives in most underprivileged areas of the world.

2) Data description

The dataset used for this challenge is formed of black and white images (of blood cells) and their labels : "uninfected" and "parasitized". Two versions of the challenge have been treated (two types/kinds of data) : "the processed data" where the training consists of 16534 images reduced to 19 numerical features and "RAW data" which contains the same number of images but here the images aren't reduced to features. Both of the datasets have almost an equal number of instances of "uninfected" and "parasitized" cells. The area under the curve ROC (AUC) has been our metric to evaluate our model.

3) Procedure and Algorithms used

In order to solve this problem, a combination of different algorithms has been tested and used. Here is a brief explanation of every algorithm that has been used:

a) <u>In pre-processing</u>

- *Preprocessed data*

To make a feature reduction, the results of PCA and SelectKBest algorithms on our data have been concatenated (So 14 features are used instead of 19). Then, StandardScaler algorithm have been applied on what we obtained from the previous methods.

- ➢ **StandardScaler**[1] : This algorithm will center our features with a standard deviation of 1. We used it because some models (classifiers) need it to work better.
- ➢ **PCA**[2] : which is a method which consists to reduce the number of features by reducing the number of correlated features And thus, that duplicated information can be avoided
- ➢ **SelectKBest**[2] : This scores the features by using a function (in our case f_classif) then selects the K highest scored features.

- *RAW data*

Extract_features has been used to extract the features. As for the preprocessed data, StandardScaler has been applied on these latter.

- ➢ **f_min_max_mean_std_gray :** As its name suggests, it extracts the features which consists in the minimum, maximum, mean and standard deviation of gray in our images.
- ➢ **Nb_pixels :** Which returns an histogram from values of pixels of each image in our data. (it takes time)
- ➢ **Extract_features :** Which uses the two previous functions and returns a concatenation of their results.

Scikit-learn algorithms have been used in processed data contrary to RAW data where functions of our own have been used for extracting the features.

b) In Model:

After applying preprocessing methods to our data, we fed both the Preprocessed and Raw data to RandomForest which seems to be the most promising classifier among the ones tested. Indeed, In order to find the best classifier model for our features, we tested 5 classifiers. Some have been chosen because they were the topic of our practical work like RandomForest, DecisionTree and Adaboost. Others like GausianNB and KNeighbors, to bring a certain variety.

As we can see in figures 1-4, the classifiers with the highest scores are AdaBoost classifier and Random forest. To select the best of these two and to find the best hyperparameters for our next model so we can improve its performances, a Randomized Search Cross-Validation has been used with different hyperparameters combinations. The results of each combination have been put in a data frame (Table 1-2). The RandomForest seems to give slightly better results than the Adaboost Classifier. That's why the RandomForest classifier has been chosen for our model.

Note:

The maximum depth for RandomForest was fixed to a small number to prevent overfitting.

Cross-Validation has been used to select the best model. This method splits our dataset into two subsets, one used in the training process and the other used to compute the score of a model using the metric given (here AUC). The best classifier is the one with the score closest to 1 which is the perfect score.

Random Forest[4] : Is a supervised learning algorithm which is based on a number limited of decision trees.

c) In Visualization

Visualization is one of the most important parts in machine learning. In order to work more efficiently functions have been implemented. These functions save the plots of the results made while running the main program to compare the results between different models.

For example :

Plot_test_estimator plots the function of score depending on n_estimators to better analyze our model (figure 2, 5). Here it seems that the score increases with the number of estimators at the beginning and then stabilizes approximately .

- **Plot_mat_conf :** which plots the confusion matrix.
- **Plot_test_model :** which plots the bar chart of training and test scores of different models.
- **Plot_test_estimator :** which plots the score depending on the number of estimators used in a random forest.
- **Plot_decision_surface_tree_classif [5]:** which plots the decision surface of 3 classifiers : Adaboost, RandomForest and Decision tree.
- **Plot_ROC (Given) :** Which plots the area under the curve ROC.

4) Results

a) PRE-PROCESSED Challenge

With the best model, the score reached a value of 0.9869 on codalab and 0.9873 on our machines (figure 2).

Note :

Given our willingness and enthusiasm to work on more interesting things, more focus has not been put on this version of the challenge.

RESULTS

#	User	Entries	Date of Last Entry	Prediction score ▲	Duration ▲	Detailed Results
1	Africa	65	04/18/20	0.9870 (1)	0.00 (1)	View
2	MOSQUITO	25	04/12/20	0.9869 (2)	0.00 (1)	View
3	SAHARA	37	04/18/20	0.9868 (3)	0.00 (1)	View

b) RAW Challenge

With the best model, the score obtained on our machines is equal to 0.9709 (figure 5) and 0.9707 on codalab.

RESULTS

#	User	Entries	Date of Last Entry	Prediction score ▲	Duration ▲	Detailed Results
1	Africa	21	04/20/20	0.9824 (1)	0.00 (1)	View
2	MOSQUITO	15	04/18/20	0.9707 (2)	0.00 (1)	View
3	medichal	3	10/19/19	0.7566 (3)	0.00 (1)	View

5) Unsuccessful attempts:

On the other hand, there are things that have been tried but have not been successful and have not brought any results. For instance :

- Expanding the data rather than reducing it, was useless and did not raise the score. Indeed, after testing that, no results were obtained.

- Since the score was not improving with our model, another direction was taken. Indeed, a CNN was tested on the preprocessed data. However, the score was always the same, neither increasing nor decreasing. Due to that and to a lack of knowledge of the CNN algorithm it was decided to stop this process and just use our model.

6) Conclusion

Not only This project taught us how to collaborate successfully with our team and how to manage 'big' projects. It also improved our knowledge in machine learning and data science and all the necessary tools for these fields such as Python language and its wonderful libraries (scikit learn, pandas, ...etc).

For next year, we hope that this class will have a real lesson about the basics of machine learning (with better explanations and less focus on 'Homework/Deadline' thing.

We wish future students a good time and a fulfilling work on their part. The most important thing it's to have fun.

7) Figures

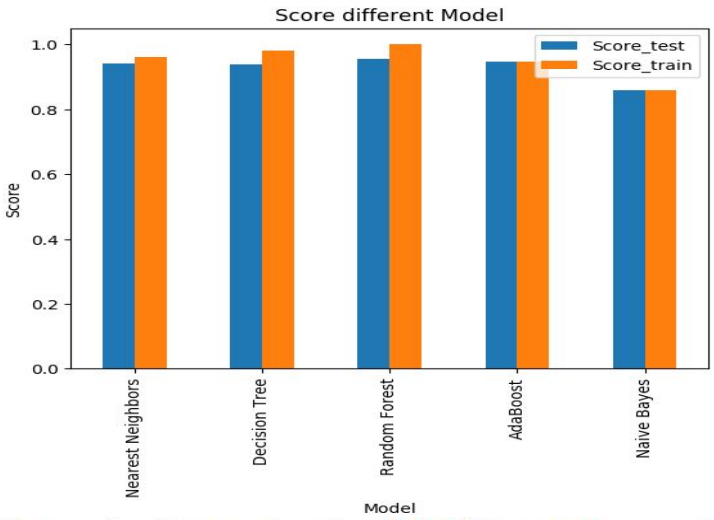

Figure 1 : Score of each model for preprocessed data.

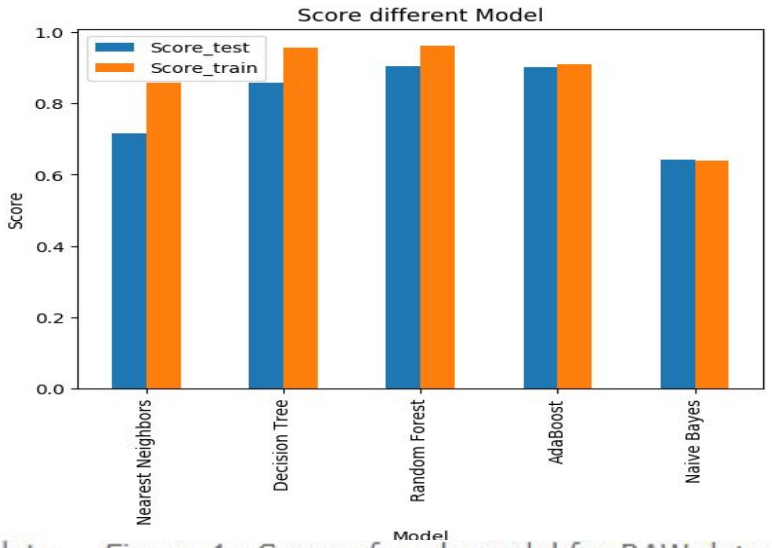

Figure 4 : Score of each model for RAW data.

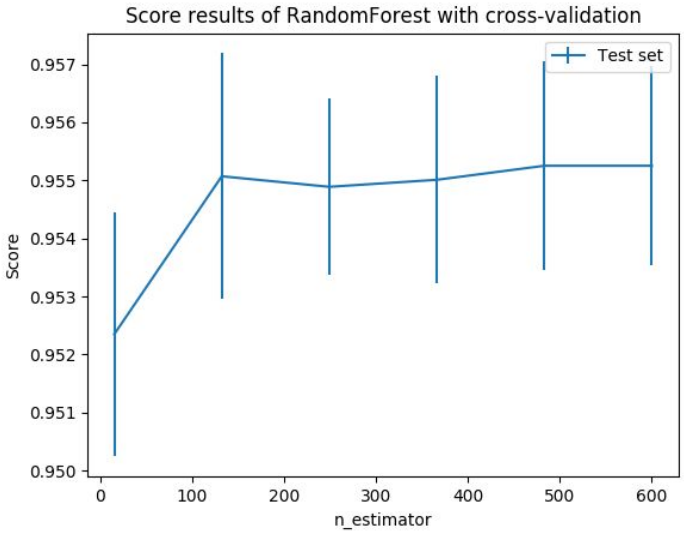

Figure 2 : Performance with hyperparameter number of estimators for preprocessed data.

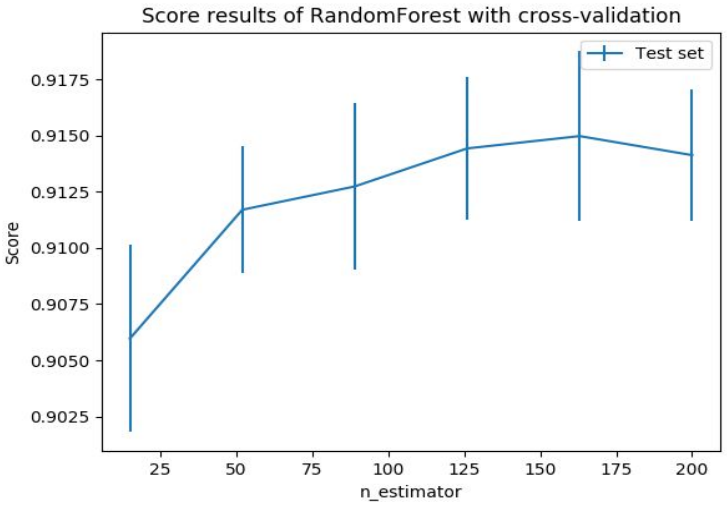

Figure 5 : Performance with hyperparameter number of estimators for RAW data.

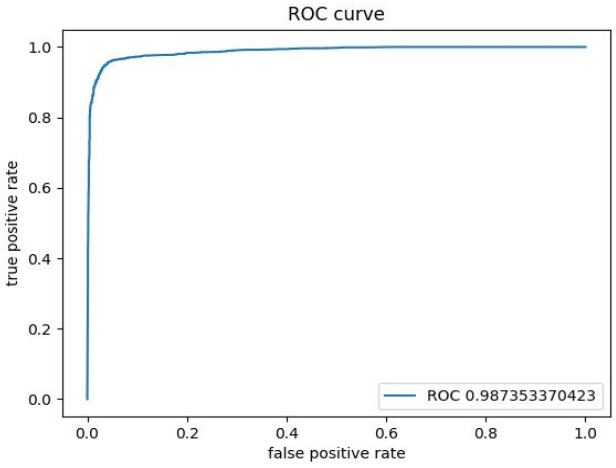

Figure 3 : Representation of ROC curve for preprocessed data.

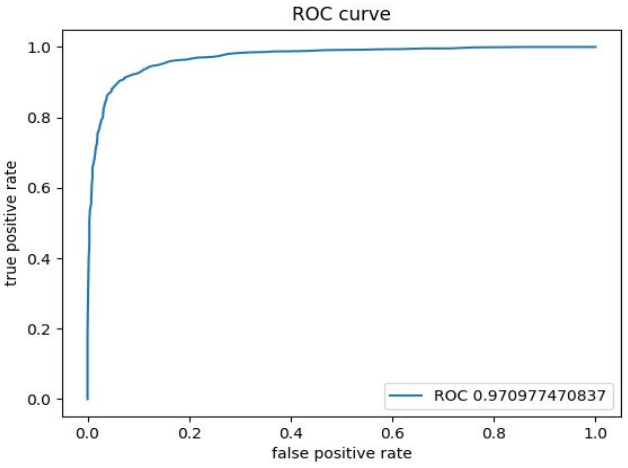

Figure 6 : Representation of ROC curve for RAW data.

Table 1 : Score for each hyperparameter combination for RandomForest.

	n_estimator	max_depth	Precision/Recall	mean	std
0	600	15	precision	0.954155	(+/-0.005080440304826461)
1	650	15	precision	0.954155	(+/-0.005080440304826461)
2	700	15	precision	0.954397	(+/-0.0038708116608201504)
3	750	15	precision	0.953974	(+/-0.0042337002540220325)
4	600	20	precision	0.954337	(+/-0.0044756259828232 5)
5	650	20	precision	0.953792	(+/-0.005806217491230225)
6	700	20	precision	0.953853	(+/-0.005685254626829561)
7	750	20	precision	0.953974	(+/-0.004717551711624579)
8	600	25	precision	0.954276	(+/-0.005564291762428897)
9	650	25	precision	0.954155	(+/-0.005322366033627679)
10	700	25	precision	0.954639	(+/-0.0048385145760251325)

Table 2 : Score for each hyperparameter combination for Adaboost.

	n_estimator	learning_rate	Precision/Recall	mean	std
0	300	0.01	precision	0.940002	(+/-0.006048143220031443)
1	400	0.01	precision	0.940607	(+/-0.007257771864037754)
2	500	0.01	precision	0.940728	(+/-0.007499697592838972)
3	600	0.01	precision	0.942422	(+/-0.005806217491230225)
4	300	0.05	precision	0.943631	(+/-0.007499697592839083)
5	400	0.05	precision	0.946111	(+/-0.003507923067618157)
6	500	0.05	precision	0.946413	(+/-0.004112737389621368)
7	600	0.05	precision	0.946958	(+/-0.005201403169227015)
8	300	0.10	precision	0.946897	(+/-0.005322366033627679)
9	400	0.10	precision	0.947684	(+/-0.0044756259828232 5)
10	500	0.10	precision	0.947623	(+/-0.0050804403048263 5)

8) References

[1]: scikit-learn standard scaler documentation ;
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[2]: scikit-learn PCA documentation ;
https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[3]: scikit-learn select k best documentation ;

https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

[4]: scikit-learn RandomForest documentation ;

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[5]: scikit-learn Decision surface documentation ;

https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_iris.html