Project name : MediChal
Team name : MOSQUITO
Team members :

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

Challenge URL : https://codalab.lri.fr/competitions/654
Github repository: https://github.com/PhamAlexT/MOSQUITO
Slides:

Challenge description

This last decade, medical imaging has become the leading and most popular field 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 affects red blood cells, Our challenge consists in distinguishing between infected and uninfected cells. This is then a simple binary classification problem.

Data description

The dataset used for this challenge is formed of black and white images (of blood cells) and their labels : “uninfected” and “parasitized”. We worked on two versions of the challenge (two kinds of data) : “the processed data” where we have for the training 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 datas have almost an equal number of instances of “uninfected” and “parasitized” cells. We used the area under the curve ROC (AUC) as our metric to evaluate our model

Procedure and Algorithms used

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

In pre-processing:

Preprocessed data :

We chose to make a features reduction, for that we concatenated the results of PCA and SelectKBest algorithms on our data. Then we applied StandardScaler algo on what we obtained by 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. So we can avoid duplicated information.
- **SelectKBest**[2] : This scores the features by using a function (in our case f_classif) then selects the K highest scored features.

RAW data :

We used Extract_features to extract the features then we applied on them like we did for preprocessed data StandardScaler.

- **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.

We notice that in preprocessed data, we used algorithms of scikit-learn libraries. While in RAW data, we used our own functions in order to extract the features.

In Model:

After applying preprocessing methods to our data, we use to both Preprocessed and Raw data a random forest classifiers which seems to be the most promoting classifier among the ones tested (figure 1)
To measure the performances of our model we used cross validation on one hand and We split our training data set into training and test sets and computed AUC on the other.

We used a Grid Search Cross Validation process (GridSearchCV) to find the best hyperparameters so we can improve our model performances. We also use a plot of function of score depending on hyperparameters such as n_estimators to better analyze our model (figure 2).

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

In Visualization :

Here we created our own functions to save our plots in order to visualize the results of our models for boths datas more efficiently.

- **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, RandForest and Decision tree.
- **Plot_ROC (Given) :** Which plots the area under the curve ROC.

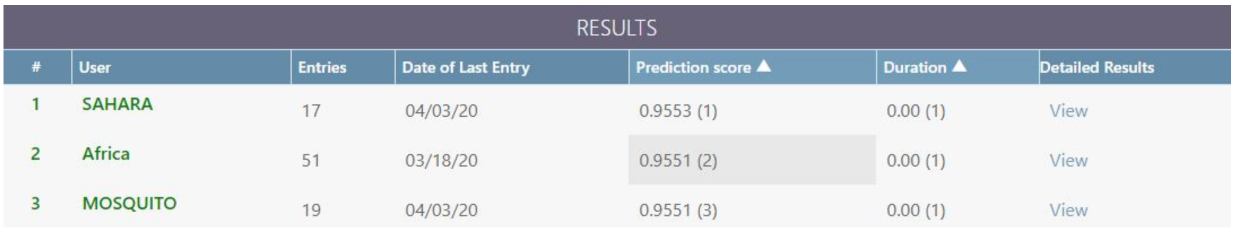
Results

PRE-PROCESSED Challenge

With our model for preprocessed data, we obtained a value 0.9864 for the AUC (figure 2) and a score on codalab equal to 0.9551.

Improvements :

We didn't focus a lot on this version of the challenge because we wanted something more interesting (e.g more challenging) to work on. But we would like to test to expand our data (instead of reducing it) and try to get a higher score.



RESULTS

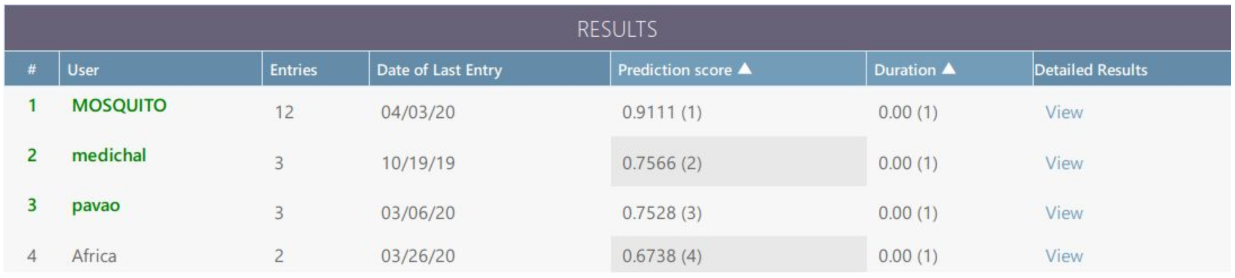
#	User	Entries	Date of Last Entry	Prediction score ▲	Duration ▲	Detailed Results
1	SAHARA	17	04/03/20	0.9553 (1)	0.00 (1)	View
2	Africa	51	03/18/20	0.9551 (2)	0.00 (1)	View
3	MOSQUITO	19	04/03/20	0.9551 (3)	0.00 (1)	View

RAW Challenge

With our model for RAW data, we obtained a value 0.96 for the AUC (figure 4) and a score on codalab equal to 0.9111.

Improvements :

We would like to try other classifiers and as for the preprocessed challenge : test different preprocessing in addition to the one we use currently.



RESULTS

#	User	Entries	Date of Last Entry	Prediction score ▲	Duration ▲	Detailed Results
1	MOSQUITO	12	04/03/20	0.9111 (1)	0.00 (1)	View
2	medichal	3	10/19/19	0.7566 (2)	0.00 (1)	View
3	pavao	3	03/06/20	0.7528 (3)	0.00 (1)	View
4	Africa	2	03/26/20	0.6738 (4)	0.00 (1)	View

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).

We hope that next year, this UE will have more sense (e.g maybe an introduction to machine learning?), no video to do (Hello Coronavirus) and the more important: That you have fun.

Enjoy your first year.

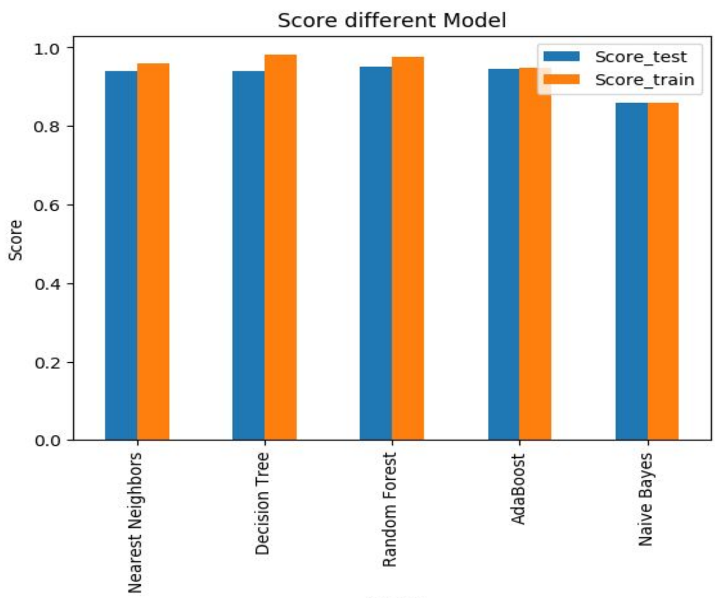


Figure 1 : Score of each model for preprocessed data.

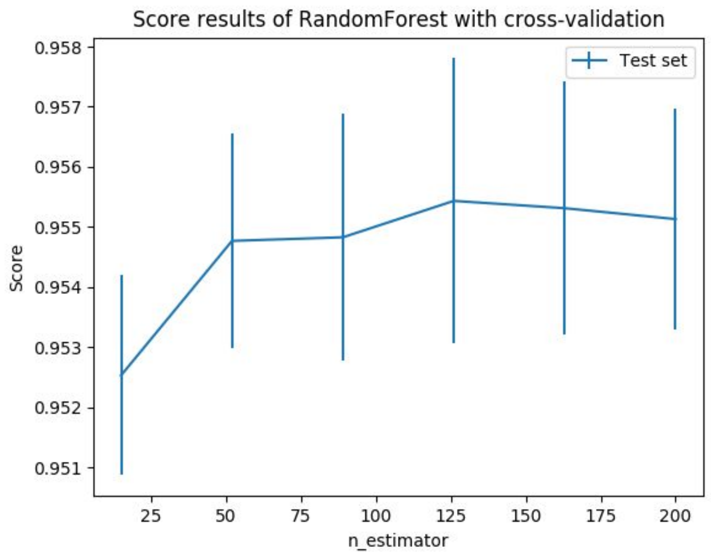


Figure 2 : Performance with hyperparameter number of estimators for preprocessed data.

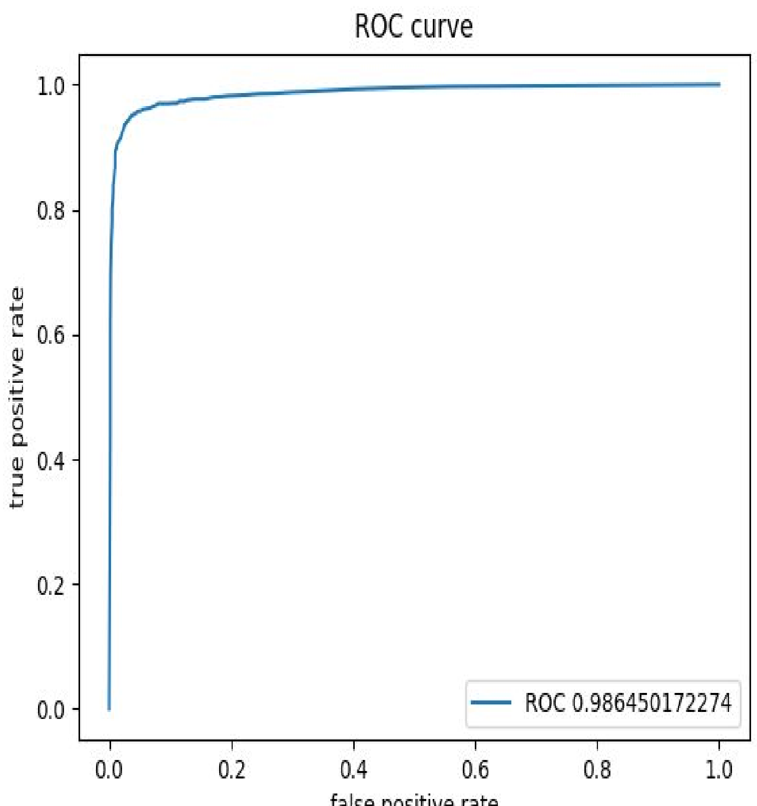


Figure 3 : Representation of ROC curve for preprocessed data.

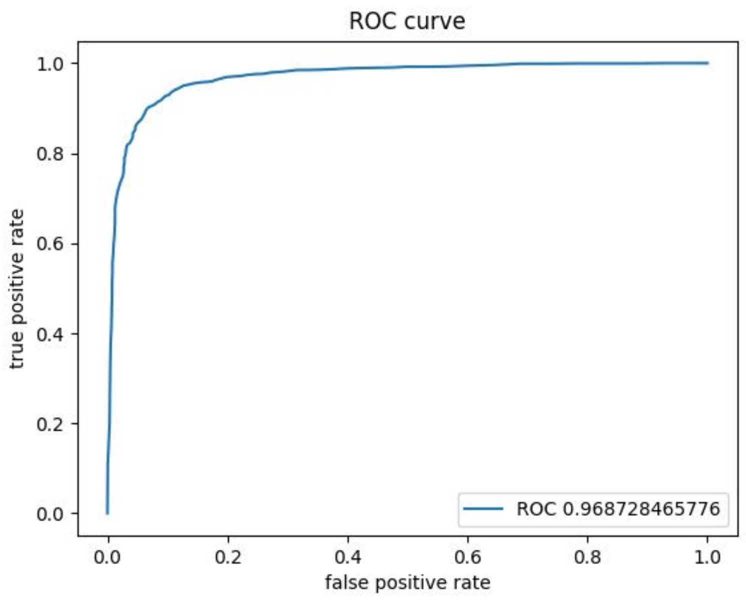


Figure 4 : Representation of ROC curve for RAW data.

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