Context and motivation :

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 the third world countries.

Malaria is an infectious sickness caused by a parasite called Plasmodium which affects red blood cells. If it isn't diagnosed and treated promptly, it can be fatal.

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.

Data and problem description :

This problem is a simple classification problem which aims to distinguish between infected and uninfected cells and then predict if a patient is sick or not. To do that, we divided the work into three parts : a subgroup for preprocessing, another for choosing the classification model and the last for visualization. Also, we used the area under the curve ROC (AUC) as our metric to evaluate our model, which allows us to know the percentage of correctly classified examples by our model. The statistics of the dataset are summarized in **Table 1**.

Approach chosen and Member's tasks :

1. **Preprocessing :** "Alexandre PHAM and Clément ELLIKER"

We are preprocessing a dataset that is already preprocessed. Our goal here was to make it better and learn everything we could to eventually face the RAW challenge.
First we wanted to get rid of the data that might negatively affect our results . Several algorithms exist to do this tasks (eg. LocalOutlierFactor, IsolationForest...). We choose IsolationForest because it was already implemented and seems more useful.
Then, we applied dimensional-reduction to our dataset. We tried 2 algorithms: PCA [2] and TSVD [1]. We also compare their result with and without outliers. Finally, for features selection, we used ExtraTreesClassifier.

2. **Model :** " Yacine MOKHTARI and Lynda ATTOUCHE "

Here, our goal is to find the best classifier model for our features within 5 different models. Following the results obtained in the TP2, we chose RandomForest, DescisionTree and Adaboost for their performance (score). And with reference to KNeighb and GaussianNB, we chose them to bring a certain variety. At this moment, we are working with the processed version of the project. So the results we had may not be the one we should use with the RAW data.

The main process is : gathering the processed data and then training a list of classifying algorithms on our dataTrain so that we can select the best one. And to do that, we used a Grid Search Cross Validation process (GridSeardCV)[4]. The 'Cross Validation' is a validation method that splits our dataset into two subsets, the first subset is used in the training process and the second one is used as a validation dataset and gives us the performance of our model. The Grid Search method not only does a cross validation, but also gives us the best hyperparameters of each model by testing different values. In other words, it allows us to know the best parameters in order to get the highest score.

3. **Visualization :** " Lilia IZRI and Anis SMATI "

Visualization is one of the most important parts in machine learning. Our task is to make the results obtained more understandable and meaningful for everyone especially for our partners so they can compare the performance of their algorithms.

For example :

- *For preprocessing* : We plot the data using only the labels (figure 1) so we can compare performance of Data clustering using gaussian mixture with and without outliers (figure 2)
    - In this exemple, to plot the data we selected 2 features : min_gray (which seems to be one of the most important) and max_gray.
- *For model* : We plot the performance of their algorithms to find the best ones (figure 3), and we plot the decision surface [3] of some of their algorithms to compare how they work (figure 4). As a bonus, we also plotted performance of RandomForest Classifier against its hyperparameters to see the variations according to hyperparameters (figures 5-6).
    - For decision surface, we use paired features with the most important ones.

For the future, our task is to assemble our functions in a Python Class and generalize them for every Data/model.

Note : Currently, tasks are shared betweens pairs, each pair is specialized in one task, the next step is to have each member being able to contribute in others tasks and assemble all our work.

Preliminary results :

Our first score results before preprocessing/model optimization are summarized in **Table 2.**

- First optimization results :

❖ Preprocessing :

Firstly, we can see that removing outliers impacts our results. In unsupervised learning (Figure 2), we can see how removing outliers helps. Besides, when testing a part of the dataset, 10% of the data were considered outliers, which is not a small part..
Then, we tested PCA and TSVD with and without outliers. If there are outliers in the dataset, we see that TSVD is really not efficient compared to PCA, indeed, TSVD does not center data unlike PCA so outliers have much more impact . However without outliers, both are not that different (Figure 7).
Finally, when using ExtraTreesClassifier on the same part of the dataset, we narrowed down our dataset to 4 dimensions (19 formerly).

❖ Model :

The results (figure 3,8) showed that, with the best hyperparameters, the models with the best score are AdaBoost classifier and Random Forest Classifier (as you can see in the illustration). To decide between these two models (because we had to choose only one classifier), we used a Randomized Search Cross Validation with other hyperparameters' combinations. We put the result of each combination in a data frame (**Table 3, 4**).
(*note* : we took an example of 10 combinations, you can find the others on github)
As we can see, the score of Random Forest raised while AdaBoost's score got lower, so we confirmed that The best model for our features is the Random Forest Model.

Goal :

Our goal now is to combine all of our work together and try to get the best scores and face the RAW data set challenge.

Preliminary visualizations :

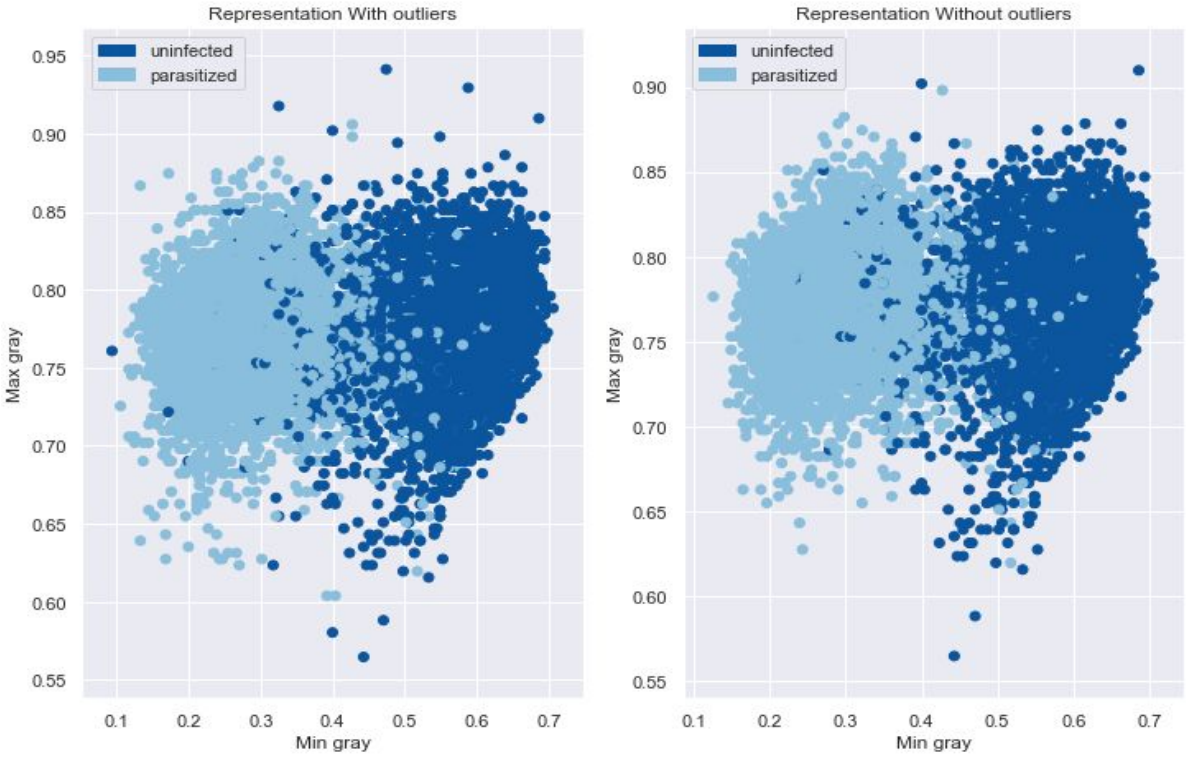

Figure 1 : Representation of Data with Labels.

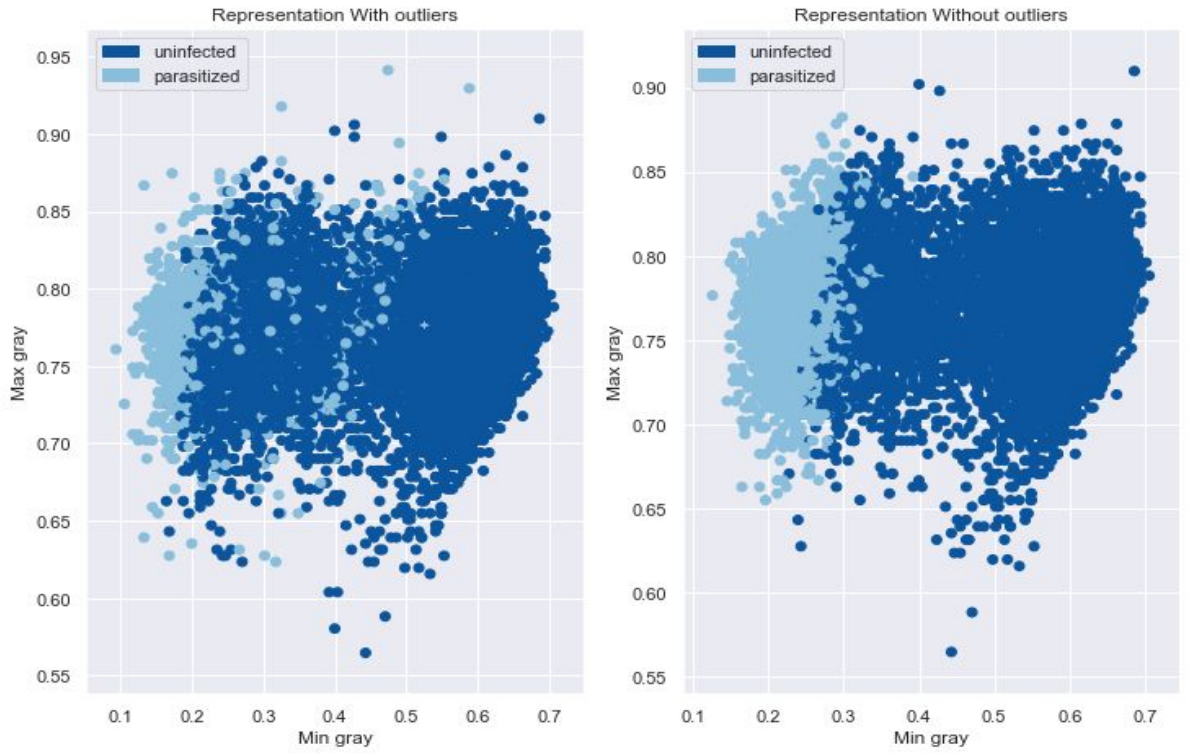

Figure 2 : Representation of Clustering Data with GaussianMixture.

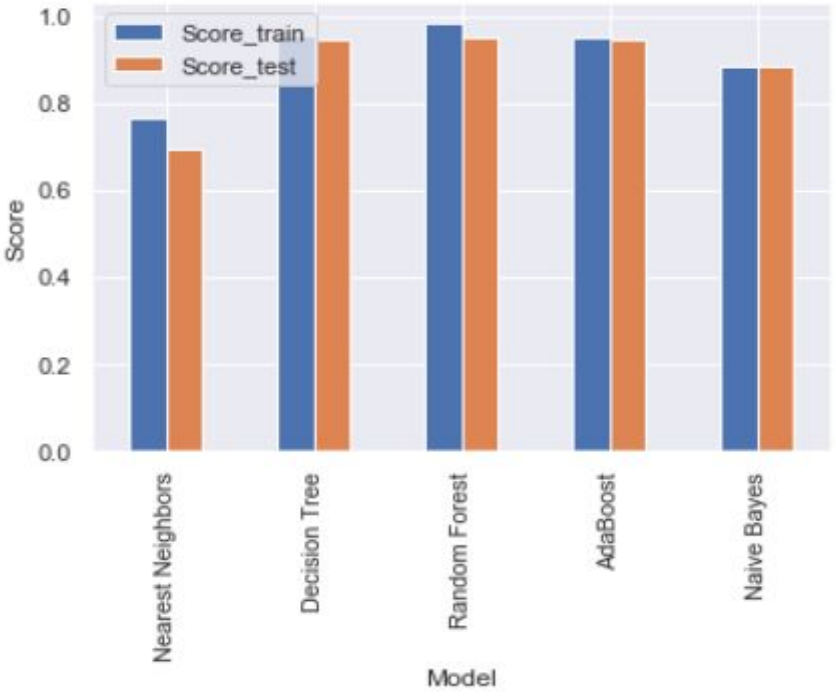

Figure 3 : Score of each model with its best hyperparameters.

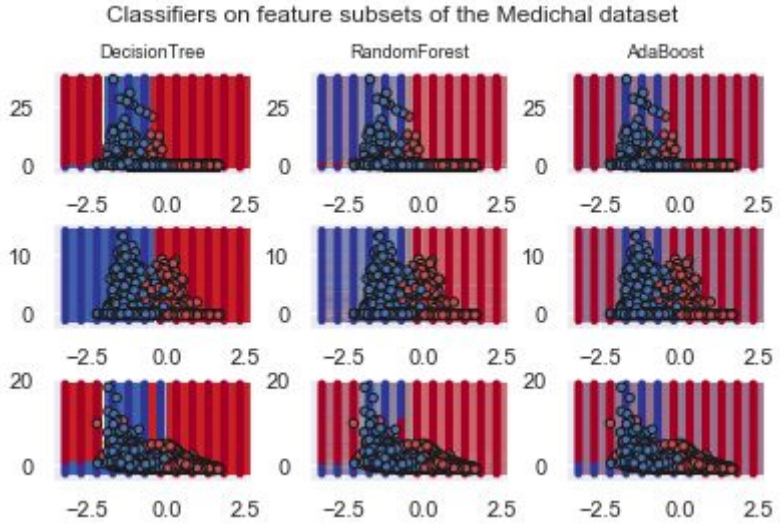

Figure 4 : Decision surface for 3 models.

***Bonus* :**

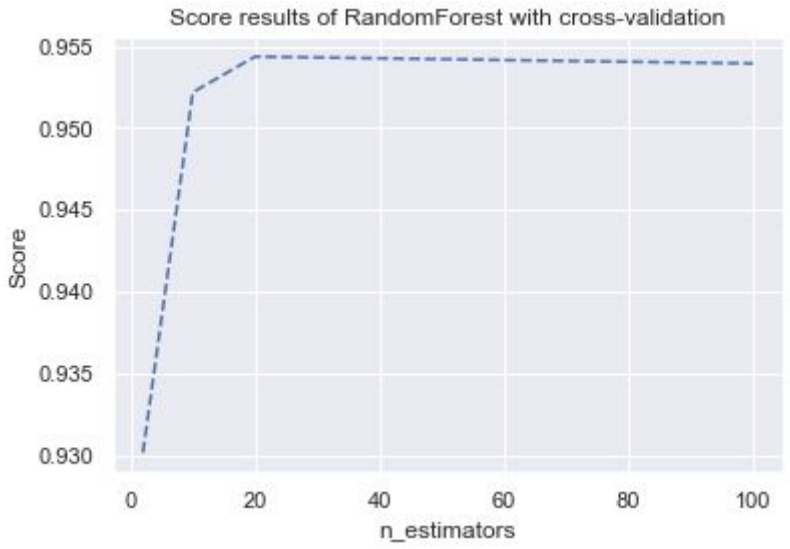

Figure 5 : Performance with hyperparameter number of estimators.

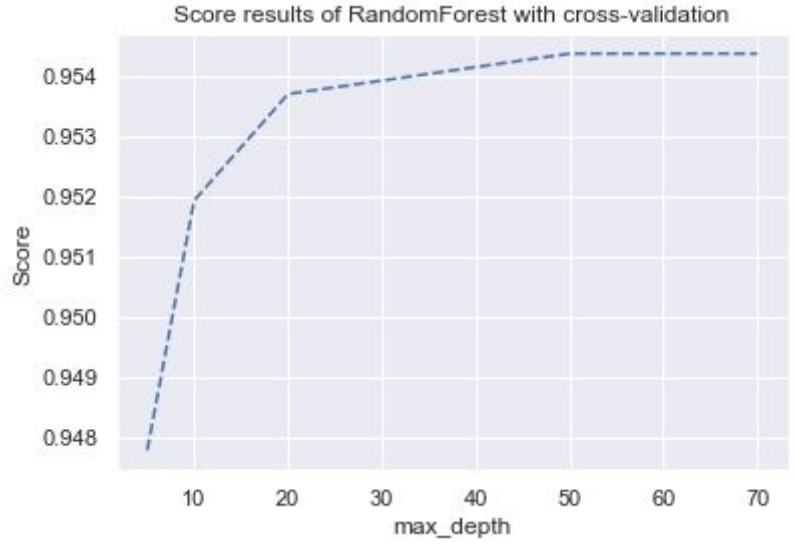

Figure 6 : Performance with hyperparameter maximum depth.

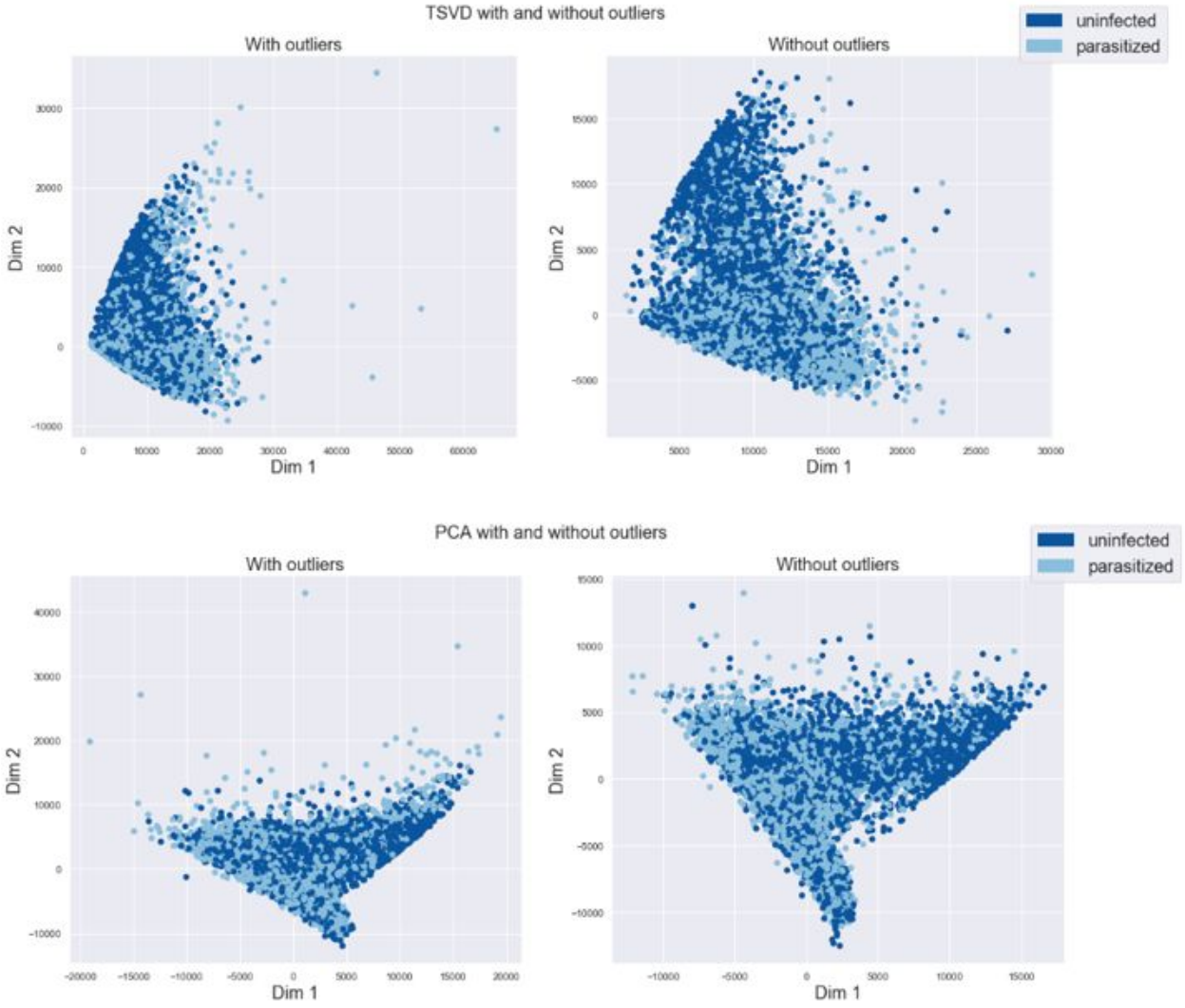

Figure 7 : PCA and TSVD

```
CV score (95 perc. CI):
-----------------------
Nearest Neighbors  : 0.70 (+/- 0.02)
Decision Tree  : 0.95 (+/- 0.00)
Random Forest  : 0.95 (+/- 0.00)
AdaBoost  : 0.95 (+/- 0.01)
Naive Bayes  : 0.88 (+/- 0.03)
```

Figure 8 : Scores obtained with the best hyperparameters combination for each classifier

Table 1 : Statistics of the Data before preprocessing optimization (Features reduce, PCA,...).

Method	Num. Examples	Num Features	Sparsity*	Has missing data ?	Num examples in each class
Training	10581	19	31.1%	No.	5352 uninfected 5229 parasitized
Validation	2646	19	31.1%	No.	1338 uninfected 1308 parasitized
Test	3307	19	30.6%	No.	1709 uninfected 1598 parasitized

* sparsity = $\frac{number\ of\ 0\ in\ data\ matrix}{number\ of\ elements\ in\ data\ matrix} \times 100$

Table 2 : Summary of preliminary score results before preprocessing optimization.

Method	Random Forest	Decision Tree	Nearest Neighbors	AdaBoost	NaiveBayes
Training	0.994	0.955	0.765	0.952	0.884
CV	0.954	0.947	0.697	0.948	0.883
Validation	0.952	0.950	0.702	0.947	0.815

*The score here is proportional to the number of images well classified.

Table 3 : Scores for each hyperparameter combination for the AdaBoost model.

	n_estimator	learning_rate	score	mean	std
0	100	0.30	precision	0.947018	(+/-0.005322366033627679)
1	100	0.10	precision	0.942059	(+/-0.007015846135236425)
2	50	0.01	precision	0.940426	(+/-0.006652957542034543)
3	200	0.30	precision	0.948349	(+/-0.005564291762429008)
4	200	0.10	precision	0.945688	(+/-0.005322366033627679)
5	100	0.01	precision	0.940426	(+/-0.006652957542034543)
6	200	0.05	precision	0.942603	(+/-0.00713680899963709)
7	100	1.00	precision	0.946111	(+/-0.006894883270835872)
8	50	0.05	precision	0.940002	(+/-0.006048143220031443)
9	150	0.05	precision	0.942422	(+/-0.007015846135236425)
10	100	0.30	recall	0.947018	(+/-0.005322366033627679)

Table 4 : Scores for each hyperparameter combination for the RandomForest model.

	n_estimator	max_depth	Precision/Recall	mean	std
0	150	5	precision	0.948288	(+/-0.0035079230676182682)
1	50	100	precision	0.954034	(+/-0.0048385145760252435)
2	200	40	precision	0.953974	(+/-0.004959477440425797)
3	200	5	precision	0.947744	(+/-0.0048385145760251325)
4	150	40	precision	0.954639	(+/-0.005564291762428897)
5	100	40	precision	0.954457	(+/-0.003749848796419486)
6	150	10	precision	0.953369	(+/-0.004717551711624579)
7	50	10	precision	0.953308	(+/-0.002177331559211293)
8	100	5	precision	0.947865	(+/-0.005322366033627679)
9	150	100	precision	0.954699	(+/-0.00447562598282325)
10	150	80	recall	0.954337	(+/-0.0042337002540220325)

References :

[1]: scikit-learn TSVD documentation ;
https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

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

[3]: scikit-learn decision surface documentation ;

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

[4]: scikit-learn GSCV ;

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html