Building a Supervised learning model with SKlearn

We tested scikit-learn (or sk learn), a python package for machine learning (https://scikit-learn.org/stable/) .
With that, you can try several different methods with custom parameters on your data to find the most suited model. We have annotated data, so we will use supervised learning.

Here, we converted the 60 x 60 images in a dataframe of 3600 values. There is only one value per pixel because the image comes from fluorescence, only the luminosity, coded from 0 to 255, is given. We normalize these values by dividing them by 255 to obtain numbers between 0 and 1.
All the images form a list, and we made a second list with corresponding labels.
We used the function `sklearn.model_selection.train_test_split` to make a train set with ⅔ of the data and a test set with ⅓ of data.

For the classifier models, we calculated the accuracy between prediction and true labels for test data, a measure that represents good classified elements.

Accuracy: elements in correct class/ total number of elements.

It takes values between 0 and 1. If Accuracy=1, all the images are classified in their correct class.

For the regressor, we calculated:
- Root mean square error (RMSE): it must be as little as possible
- R2: the percentage of variance explained by the model. Values between -1 and 1. If equal to 1, the model explains the data.

Note that we will calculate the 3 metrics for each model to be able to compare classifier and regressor performances.

Multiclass classifier

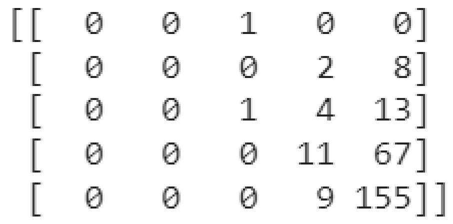
First we consider that every image must be classified in 6 classes, each for a specific number of receptors, from 1 to 6. We will test several models and compare the results:

With the 819 annotated images:

	Accuracy	RMSE	R2
MultinomialNB	0.579	0.926	-0.433
RandomForestClassifier	0.616	0.797	-0.233
DecisionTreeClassifier	0.480	1.170	-0.810

KNeighborsClassifier (n_neighbors=6)	0.554	0.970	-0.501
MLPClassifier	0.613	0.819	-0.267

But the results are very bad, we obtain average accuracy only because of the high number of images with 6 receptors: the model learns to always say “there are 6 receptors” to have good results.



```
[[  0   0   1   0   0]
 [  0   0   0   2   8]
 [  0   0   1   4  13]
 [  0   0   0  11  67]
 [  0   0   0   9 155]]
```

We need data augmentation, by rotating and mirroring the images.
The new dataset contains 12000 images, with 2000 in each class, from one to six receptors.

With the 12000 images, after data augmentation:

	Accuracy	RMSE	R2
MultinomialNB	0.385	2.535	0.142
RandomForestClassifier	0.796	0.528	0.822
DecisionTreeClassifier	0.651	1.090	0.631
KNeighborsClassifier (n_neighbors=6)	0.686	0.943	0.681
MLPClassifier	0.729	0.568	0.808

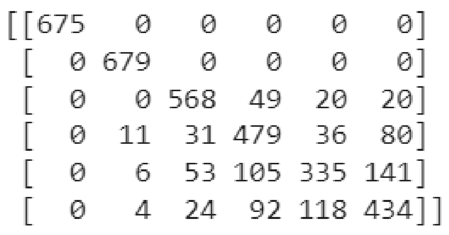
The better method is the random forest: we tried to optimize this model by using different combinations of parameters, outside the defaults values. To proceed, we used the GridSearch function of the sklearn package.

RandomForestClassifier v2 (max_depth=30)	0.799	0.516	0.825
RandomForestClassifier v3 (max_depth=30,	0.810	0.490	0.834

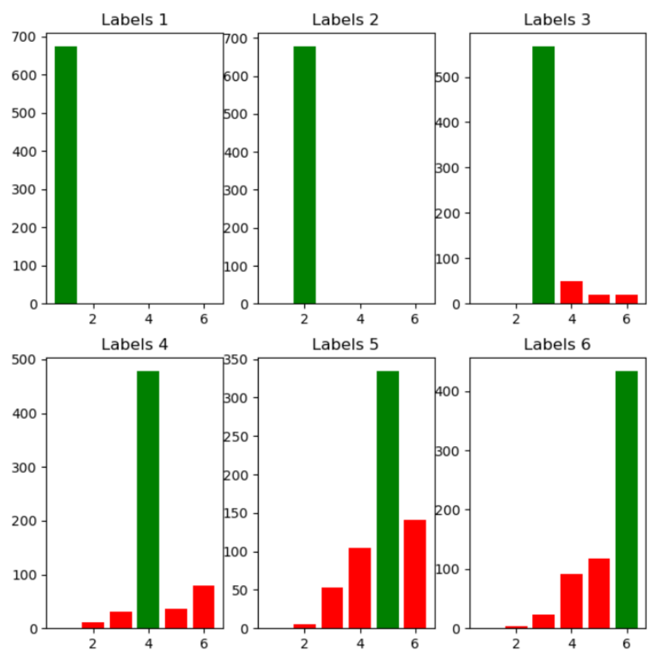
n_estimators=200, criterion='entropy')			
one_vs_all (RandomForestClassifier v3)	0.820	0.463	0.843
one_vs_one (RandomForestClassifier v3)	0.803	0.491	0.834

Once we had an improved version of our model, RandomForestClassifier v3, we used it in the specialized models for multiclass classifiers: one_vs_all and one_vs_one. The accuracy improved again with one_vs_all, the better classifier tested until now.

If we look at the confusion matrix obtained from the test set result on this models, we can discuss about several points:



```
[[675   0   0   0   0   0]
 [  0 679   0   0   0   0]
 [  0   0 568  49  20  20]
 [  0  11  31 479  36  80]
 [  0   6  53 105 335 141]
 [  0   4  24  92 118 434]]
```



- on one line, all the images with the same label, and the column correspond to the label found by the model.
- The diagonal represents the well-classified images. For each class, the majority of the images are well classified.
- all the images with 1 and 2 receptors are correctly labeled. It’s probably because all images with one receptor were augmented from only one image, and those with 2 receptors from 4 images. There is very little diversity, and the model can have difficulties identifying ommatidium with 1 or 2 receptors during its usage.

But this model has a flaw on this type of data: if the classifier is wrong about an image, it does not take account of the fact that a mistake between class 1 (1 receptor) and class 5 (5 receptors) is more important than an error between class 1 and class 3. To change that, we will consider another type of model: the regressor .

Regressor

The regressor, given an image, return a float number that correspond of the numbers of receptors, following the relation of a model between data and label.

	RMSE	R2	Accuracy (round)
LinearRegression	2.138	0.277	0.339
LogisticRegression	1.413	0.522	0.564
DecisionTreeRegressor	1.037	0.649	0.632
DecisionTreeRegressor v2 (max_depth=20, max_leaf_nodes=30, min_samples_leaf=30)	0.882	0.702	0.486
DecisionTreeRegressor v3 (max_depth=40, max_leaf_nodes=40, min_samples_leaf=40)	0.855	0.711	0.490
RandomForestRegressor	0.443	0.850	0.635
RandomForestRegressor v2 (max_depth=30, n_estimators=200)	0.444	0.850	0.636

The better results are obtained with random forest regressor, but when we round the number to correspond to an integer number of receptors, and calculate the accuracy, there are very bad results. We don’t know exactly why, but we prefer to use the random forest classifier.

Deliverable

- The notebook used to test the differents models and make the research
- The pickle object that contains the best model. (not furnish in the Git Lab, the file is to heavy (50 Mo) to be in the repository, but we will send it to the project owners
- The notebook to use the pickle objects