Exercises with ensembles

Pablo García Fernández
(pablo.garcia.fernandez2@rai.usc.es)

Thursday 20th January, 2022

Abstract

The aim of this report is to present the work carried out during the execution of the exercises of the unit 8 assignment, as well as the obtained results. In addition to this report, we also attach the developed .py code, the obtained figures and a .txt file with the results.

Exercise 2: Divide hepatitis in train/test and calculate measures.

As in all reports, the starting point is the hepatitis and wine datasets. In this exercise we asses Random Forest and AdaBoost performance using as evaluation methodology the separation of the data into two different training/test sets. The coefficients of the models are fit on the training set and the trained model is evaluated on the never-used test set. Note that there is no hyperparameter tuning or cross-validation.

The statement asks us to report for each dataset a varied number of evaluation metrics. They are present in "Tab. 1" (accuracy, kappa, precision, recall and F1) and "Fig. 1" (confusion matrix, AUC and ROC). We also include time measurement.

Finally, in order to perform a correct separation of the samples in the train/test splits, it has been done in a stratified way, i.e. maintaining the relative populations of the classes. The percentage of total sampels destined to test is 50 %. The number of estimators of each model is 100. Moreover, due to the random component of these algorithms, the results may vary between runs.

In the results we can see how Random Forest performs slightly better than AdaBoost. The computational cost of both, in terms of time consumed, is practically the same.

Table 1: RF and AdaBoost. Test measures

Hepatitis dataset

Classifier	**Metrics**					
	time (s)	**acc. (%)**	**kappa (%)**	**precision (%)**	**recall (%)**	**f1 (%)**
RF	0.0608	84.62	48.00	87.88	93.55	90.62
AdaBoost	0.0648	82.05	42.28	87.50	90.32	88.89

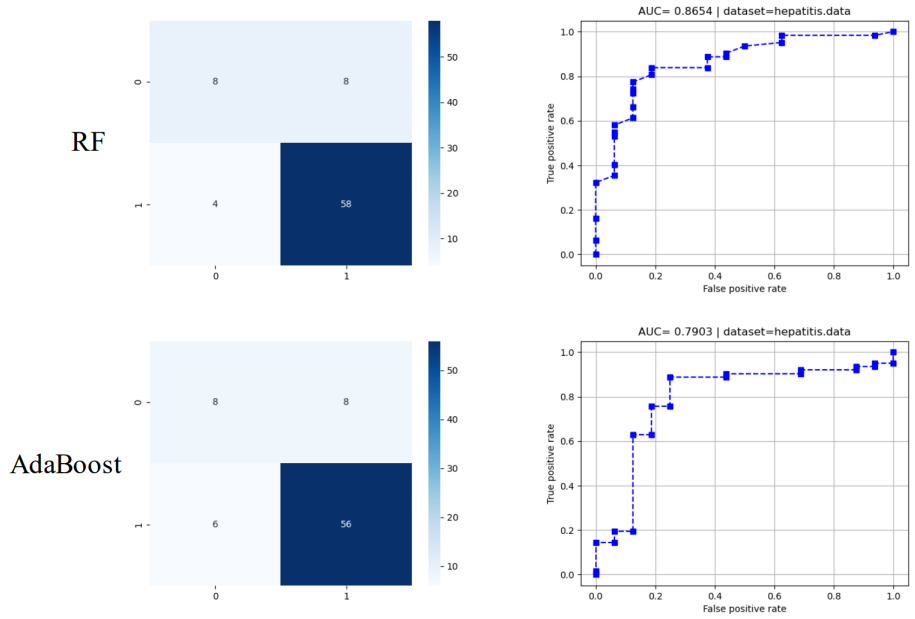

Figure 1: RF and AdaBoost. Confusion matrix, AOC, ROC

Exercise 3: Use RF and AdaBoost with scikit-learn cross validation (4,5,10). For AdaBoost tune *n_estimators*.

Some consideratiosn about the exercise:

- **RF:** To evaluate the Random forest model we use (as requested in the statement) a cross validation with the `cross_val_predict` method and 4, 5 and 10 folds. As there is no parameter tuning, we set the number of estimators to 100.
- **AdaBoost:** To evaluate the AdaBoost model, we use the same approach. However, in this case it is necessary to tune the number of estimators. To do this, we use the `GridSearchCV` function on the entire data set with f1-score (this is the second of the tuning approaches mentioned in the practical guide). Tuning results are presented in "Fig. 2" (best for hepatitis: 60, 70.9% f1; best for wine: 100, 98.4% f1).

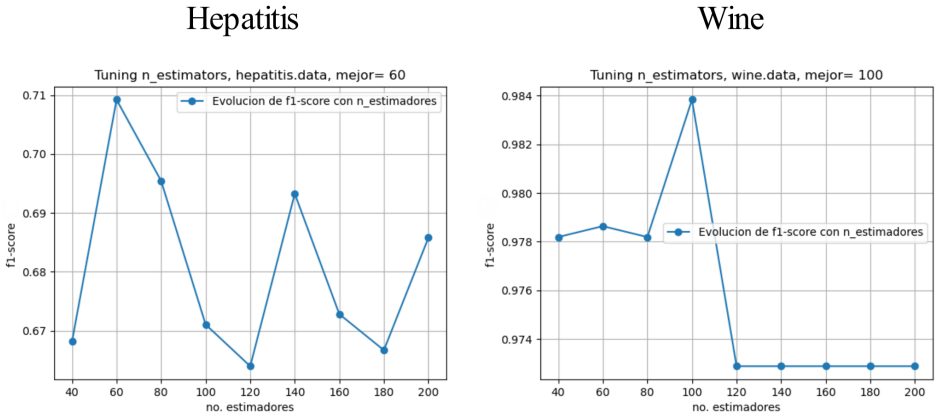

Figure 2: Tuning step AdaBoost

"Tab. 2" shows the results of the cross-validation test with different number of folds. "Tab. 3" the obtained confusion matrices.

Table 2: Test results. RF and AdaBoost with cross validation (AdaBoost with tuned parameters)

Hepatitis dataset

Classifier	**Metrics**		
	acc. (%)	**kappa (%)**	**Time (s)**
RF 4-folds	83.23	42.11	0.2490
RF 5-folds	85.16	51.93	0.3110
RF 10-folds	81.94	40.76	0.6379
AdaBoost 4-folds	83.87	49.01	13.4856
AdaBoost 5-folds	82.58	43.58	16.9057
AdaBoost 10-folds	78.71	27.48	34.1159

Wine dataset

Classifier	**Metrics**		
	acc. (%)	**kappa (%)**	**Time (s)**
RF 4-folds	97.75	96.60	0.2507
RF 5-folds	98.31	97.45	0.3140
RF 10-folds	97.19	95.74	0.6326
AdaBoost 4-folds	95.51	93.21	13.3938
AdaBoost 5-folds	97.75	96.59	17.1853
AdaBoost 10-folds	98.31	97.45	34.7962

Table 3: Confusion matrices

Hepatitis dataset

Classifier	**k=4**	**k=5**	**k=10**
RF	$\begin{bmatrix} 14 & 18 \\ 8 & 115 \end{bmatrix}$	$\begin{bmatrix} 18 & 14 \\ 9 & 114 \end{bmatrix}$	$\begin{bmatrix} 15 & 17 \\ 11 & 112 \end{bmatrix}$
AdaBoost	$\begin{bmatrix} 18 & 14 \\ 11 & 112 \end{bmatrix}$	$\begin{bmatrix} 16 & 16 \\ 11 & 112 \end{bmatrix}$	$\begin{bmatrix} 11 & 21 \\ 12 & 111 \end{bmatrix}$

Wine dataset

Classifier	**k=4**	**k=5**	**k=10**
RF	$\begin{bmatrix} 59 & 0 & 0 \\ 1 & 67 & 3 \\ 0 & 0 & 48 \end{bmatrix}$	$\begin{bmatrix} 59 & 0 & 0 \\ 1 & 68 & 2 \\ 0 & 0 & 48 \end{bmatrix}$	$\begin{bmatrix} 57 & 2 & 0 \\ 1 & 68 & 2 \\ 0 & 0 & 48 \end{bmatrix}$
AdaBoost	$\begin{bmatrix} 57 & 2 & 0 \\ 1 & 65 & 5 \\ 0 & 0 & 48 \end{bmatrix}$	$\begin{bmatrix} 58 & 1 & 0 \\ 1 & 68 & 2 \\ 0 & 0 & 48 \end{bmatrix}$	$\begin{bmatrix} 59 & 0 & 0 \\ 1 & 68 & 2 \\ 0 & 0 & 48 \end{bmatrix}$

Exercise 4

In this exercise we are asked to use RF and AdBoost by tuning the parameters through the 3 approaches mentioned in the practice guide:

- **Approach 1:** Split the entire data into training/test sets. Tune the hyperparameters in the training set using `GridSearchCV()`. Once the best values are calculated, test the classifier on the test set.
- **Approach 2:** Use the whole dataset to tune the hyper-parameters using the function `GridSearchCV()`. Once the best values are calculated, test the classifier on the whole data using cross-validation (with `cross_val_predict()`).
- **Approach 3:** Tuning with `crea_folds` function. This is the well-known methodology used in previous practices.

In approaches 2 and 3, where it is necessary to use cross-validation, we use 4 folds. In approach 1, where it is necessary to manually divide the data into training and test sets, we do int in a stratified manner using a test size of 50%.

For RF we tune the number of estimators (between 40-200 with 20 step) and the number of maximum features (between 3-13 with 2 step). For AdaBoost we only tune the number of estimators (between 40-200 with 20 step).

First we present the tuning results for both datasets. "Fig. 3" for hepatitis and "Fig. 4" for wine. It should be noted that since RF requires tuning 2 hyperparameters, the graph is displayed as a heatmap. Detailed results can be found in the attached files (*5-ensembles_resultados.txt*). Anyway, the figures are self-contained because they show the best configuration of the title.

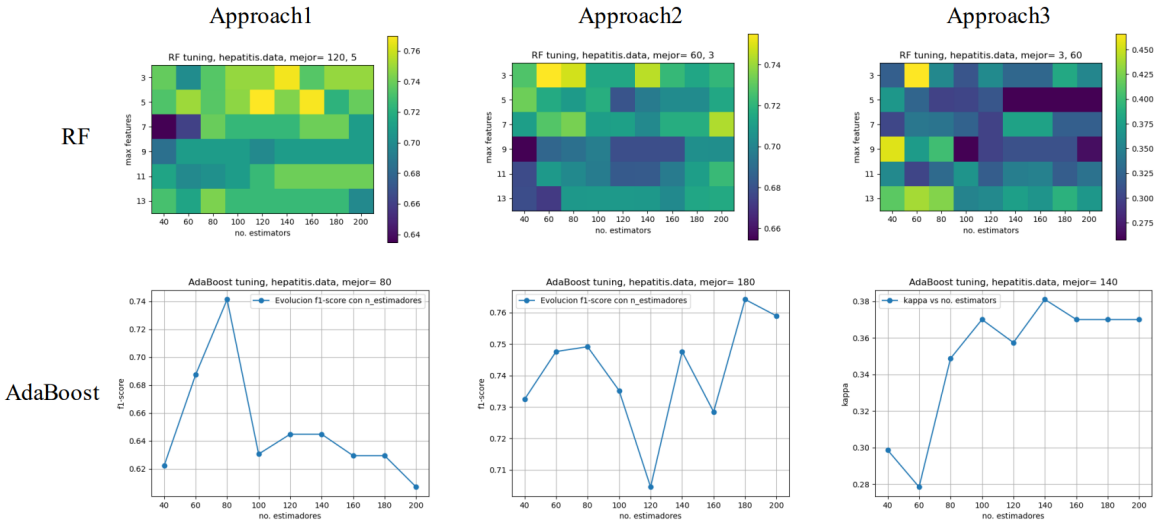

Figure 3: Tuning step on hepatitis dataset. RF and AdaBoost with the 3 approaches

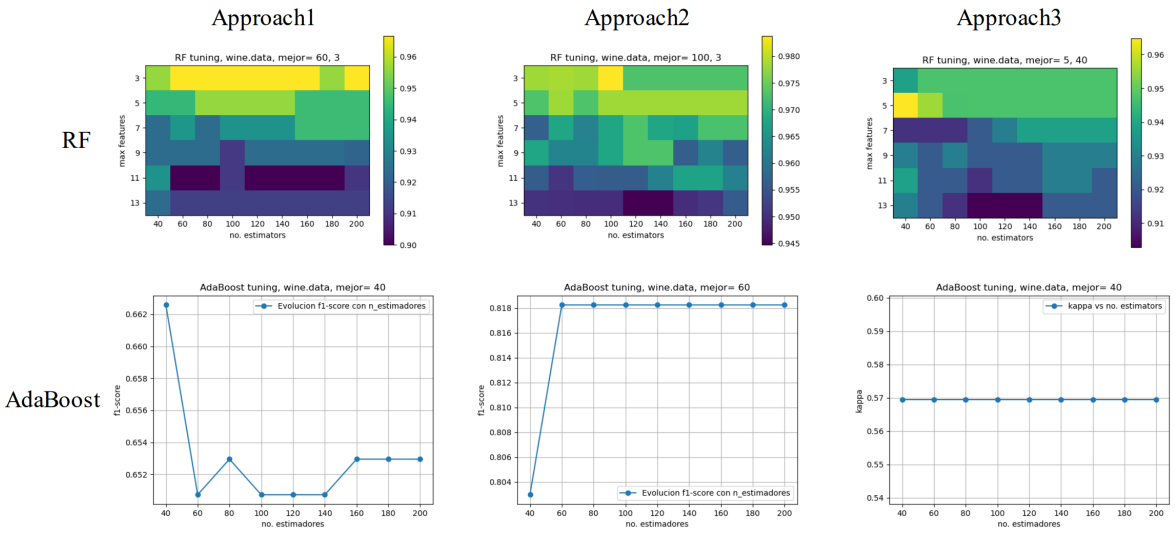

Figure 4: Tuning step on wine dataset. RF and AdaBoost with the 3 approaches

Finally, "Tab. **??**" displays the results of the test on the best tuned configuration for each dataset.

Table 4: Test results over hepatitis.

Hepatitis dataset

Classifier	**Results**	
	acc. (%)	**kappa (%)**
Random Forest approach-1	84.62	48.00
Random Forest approach-2	84.52	45.13
Random Forest approach-3	83.54	44.10
AdaBoost approach-1	80.77	36.62
AdaBoost approach-2	81.94	43.56
AdaBoost approach-3	81.10	35.74

Wine dataset

Classifier	**Results**	
	acc. (%)	**kappa (%)**
Random Forest approach-1	97.75	96.60
Random Forest approach-2	96.07	94.05
Random Forest approach-3	96.43	94.55
AdaBoost approach-1	59.55	36.85
AdaBoost approach-2	80.90	70.98
AdaBoost approach-3	93.88	90.62

Exercise 5: Use RF on CooCurr and LBP. Compare the results with the other classifiers

Some important remarks for a fair comparison between the different classifiers. In all of them:

- Cross-validation with 4 folds was used.
- Approach 3 (the one with `crea_folds` function) was used to tune the hyperparametres.
- Patterns are standardized with the mean and the standard deviation of the training set patterns.
- As measures, kappa, accuracy and run time (after the hyperparameters tuning) are shown.
- We do not present tuning results (they can be seen in the previous reports), only the used hyper-parameters.

"Tab. 5" shows the final comparison for both datasets. SVC is the best-performing classifier, closely followed by random forest.

Table 5: Final comparison

CooCur dataset

Classifier	**Results**		
	acc. (%)	**kappa (%)**	**Time (s)**
KNN ($k = 1$)	71.88	71.34	0.019
LDA	85.42	85.14	0.011
MLP ($conf = [30]$)	72.92	72.41	17.290
ELM ($conf = [40]$)	69.79	69.22	0.040
SVC OVA ($kernel = gaussian, \sigma = 0.00781, \lambda = 2^{15}$)	**86.46**	**86.20**	**0.247**
RF ($n\_estimators = 60, max\_features = 9$)	73.50	73.00	1.037

LBP dataset

Classifier	**Results**		
	acc. (%)	**kappa (%)**	**Time (s)**
KNN ($k = 1$)	86.57	86.32	0.018
LDA	85.65	85.38	0.008
MLP ($conf = [30]$)	83.91	83.61	2.10
ELM ($conf = [40]$)	77.66	77.24	0.01
SVC OVA ($kernel = gaussian, \sigma = 0.5, \lambda = 8$)	**90.62**	**90.45**	**0.220**
RF ($n\_estimators = 200, max\_features = 3$)	87.85	87.62	1.42