Ensemble methods

Exercises

Single Classifier vs Bagging vs Boosting

In this exercise you will compare the accuracy of various types of classifier ensembles:

- a single decision tree – `DecisionTreeClassifier`
- a bagged decision tree – `BaggingClassifier`
- the *AdaBoost* algorithm for decision trees – `AdaBoostClassifier`
- the *Gradient Boosting* algorithm for decision trees – `GradientBoostingClassifier`

In this exercise use the full dataset obtained using `load_wine()` function - `sklearn.datasets`.

Use the following parameters values for decision trees and classifier ensembles (look for the appropriate parameters in the `sklearn` documentation) :

- in all cases:
  - the minimum number of samples required to be at a leaf node: 3
- for all classifier ensembles:
  - the number of base estimators in the ensemble: should be constant for all methods (eg. 50 or 100)
- for all methods using boosting:
  - maximum depth of the individual estimator: 1
- `AdaBoostClassifier`
  - algorithm: SAMME
- `GradientBoostingClassifier`
  - learning rate: 1.0
  - subsampling: 0.5

Remember to manually set the random seed (parameter `random_state`) whenever it's possible – to ensure reproducibility of results!

For each model determine its average accuracy using 5-fold stratified cross-validation. You may use `cross_val_score` and `StratifiedKFold`
The expected values (for `random_state=1`) should look similar to these:

```
Decision tree scores: [0.94444444 0.80555556 0.80555556 0.91428571
0.97142857], (avg: 0.8882539682539681)
Bagging scores: [0.91666667 0.91666667 0.94444444 0.97142857 1.        ],
(avg: 0.9498412698412698)
AdaBoost scores: [0.83333333 0.91666667 0.91666667 1.         1.        ],
(avg: 0.9333333333333332)
Gradient boosting scores: [0.91666667 0.97222222 0.97222222 0.42857143
0.97142857], (avg: 0.8522222222222222)
```

Determine an approximated number of iterations after which the *Gradient Boosting* model yields almost no further improvement by following these steps:

1. Fit the model on the whole dataset.
2. Compute the cumulative improvement, i.e. the *OOB loss* (call `numpy.cumsum()` function with `GradientBoostingClassifier.oob_improvement_` attribute as its argument).
3. Prepare a plot of the above-mentioned OOB loss with respect to the number of iterations.