AUC and ensemble methods with PYTHON

Objective: The objective of today's exercise is to understand (i) how the ensemble methods bagging and boosting is used to improve the performance of classifiers, (ii) how the receiver operating characteristic (ROC) is used to evaluate the performance of two-class classification problems, and (iii) how artificial neural network and logistic regression can be generalized to multi-class classification.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* as well as the files in the exercise 9 folder available from Campusnet.

Discussion forum: You can get help on our online discussion forum:
https://piazza.com/dtu.dk/spring2023/02450

Software installation: Extract the Python toolbox from DTU Inside. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/` Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Regression	`y`	numpy.array	$N \times 1$	Dependent variable (output): For each data object, `y` contains an output value that we wish to predict.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\texttt{y}_n \in \{0, 1, \ldots, C-1\}$, where C is the total number of classes.
	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	integer	Scalar	Number of classes.
Cross-validation				All variables mentioned above appended with `_train` or `_test` represent the corresponding variable for the training or test set.
	`⋆_train`	—	—	Training data.
	`⋆_test`	—	—	Test data.

9.1 Ensemble methods

In this part of the exercise we will consider ensemble methods to improve the classification performance. In particular we will consider bagging and boosting methods.

In bagging we randomly sample with replacement the same number of samples as the size of the training data. This is also denoted bootstrapping and it can be shown that on average each bootstrap sample will contain approximately 63% of the samples in the data.

In boosting we adaptively change the distribution that we sample the training examples from so that the classifiers will focus on examples that are hard to classify. A particularly well known boosting method is *"AdaBoost"* which is described in section 15.4 of the lecture notes *Introduction to Machine Learning and Data Mining*.

In this part of the exercise we will work with a synthetic data set which has two classes and two attributes, x_1 and x_2. This data set cannot be classified correctly with any linear classifier, such as logistic regression or an artificial neural network with one hidden unit, as long as only x_1 and x_2 are used as features.

9.1.1 Load the artificial dataset (`Data/synth5` file) with `loadmat` function. Inspect the data by making a scatterplot. Why can this data set not be classified correctly using logistic regression?

Inspect and run the script `ex9_2_1.py`. The script fits an ensemble of logistic regression models to the data, using the bootstrap aggregation (bagging) algorithm. Use $L = 100$ bootstrap samples. This requires creating 100 bootstrapped training sets, fit a logistic regression model to each, and combine the results of their outputs to make the final classification. Explain how the error rate is computed (on the training set) and how the decision boundary is plotted.

Script details:

- *To generate the bootstrap sample, you need to draw N data objects with replacement from the data set.*
- *You can use the function* `bootstrap()` *function from the toolbox to generate random numbers from a discrete distribution. You can write something like* `X_bs, y_bs = bootstrap(X, y, N, weights)` *to make a bootstrap sample.*
- *To make the final classification, take a majority vote amongst the classifiers in the ensemble. You can use simple class* `BinClassifierEnsemble` *from the toolbox.*
- *To plot the decision boundary, you can use the function* `dbplot()` *or* `dbprobplot()` *from the toolbox. It requires as an input an object that implements* `predict(X)` *and* `predict_proba(X)` *methods. You can call it e.g. like this:* `dbprobplot(fitted_classifier, X, y, 'auto', resolution=200)`

Bagging is known to be most effective for non-linear classifiers. Show that bagging only leads to a limited performance improvement for the logistic regression classifier.

Try also the data set in `Data/synth6` which is the] one used as an example in the lecture.

9.1.2 In the script `ex9_2_2.py`, the script `ex9_2_1.py` has been modified so that boosting is used instead of bagging. The script fits an ensemble of logistic regression models to the data, using the AdaBoost algorithm. Notice the script uses $L = 100$ rounds of boosting. This requires creating a

randomly chosen training set, fit a logistic regression model to it, evaluate its performance and update the weights accordingly, and compute a classifier importance. This process is repeated $L = 100$ times, and ultimately the trained ensemble of classifiers is combined to make a final classification. Compute the error rate (on the training set) and make a plot of the decision boundary.

Script details:

- *Read the hints to the previous exercise.*
- *Note that you will need two sets of weights in the algorithm. One is a weight for each of the N data objects, that is adapted when the boosting algorithm proceeds (we could call these* `weights`*). The other is the importance weights for the L trained classifiers (we could call these* `alpha`*).*
- *You can use the function* `bootstrap()` *to generate random numbers from a discrete distribution, as before. You will need to update the 'weights' parameter in according to AdaBoost algorithm, in every iteration.*
- *To make the final classification, take a weighted majority vote amongst the classifiers in the ensemble (weighted by* `alpha`*). Note that* `alpha` *needs to be normalized so that it sums to one. ii*

when L increased from 100 to 500, error rate dropped from 4.8% to 1.4%

Show that, if you use enough rounds of boosting, the data set can be perfectly classified.

Try also the data set in `Data/synth6` which is the one used as an example in the lecture.

Let us return to the bagging algorithm, which we found to be of little use for logistic regression. Now, we will try the algorithm on a non-linear classifier: the decision tree. Bagging applied to decision trees is often called "random forests". The Python's package `sklearn.ensemble` has implemented bagging for random trees, see the class `RandomForestClassifier()`.

9.1.3 The data set `Data/synth5` we have used so far can trivially be fitted using a decision tree. Can you explain why? We will consider a different data set, which you can load from the file `Data/synth7`. Inspect and run the script `ex9_2_3.py`. Explore the data set and explain in what sense this is more challenging for a decision tree.

Notice the script fits a random forest (an ensemble of bagged decision trees) to the data using $L = 100$ rounds of bagging. Compute the error rate (on the training set) and make a plot of the decision boundary.

Script details:

- *Type* `help(sklearn.ensemble)` *to learn about the functions for fitting random forests with bagging.*

For comparison, you can try also to fit a regular decision tree (without bagging or pruning). Does bagging appear to improve on the classification, and if so, in what sense? Observe how the classification rate and decision boundaries decrease when you reduce number of bootstrap iterations.

Try the script on the other synthetic multi-class data sets you have studied before, (`Data/synth5` ... `Data/synth7`).

9.2 The receiver operating characteristic (ROC)

The receiver operating characteristic (ROC) is commonly used to evaluate and compare the performance of two-class classifiers, where one class is denoted "positive" and the other is denoted "negative." The ROC is a graphical approach for displaying the tradeoff between the true positive rate (y-axis) and the false positive rate (x-axis). For classifiers such as logistic regression and artificial neural networks that estimate the class labels by thresholding a continuous output variable, an ROC curve can be plotted by varying the threshold value.

9.2.1 Inspect and run the script `ex9_1_1.py`. The script loads the wine data (`Data/wine2`) into Python with the `loadmat` function. Notice how the data is divided into 50% for training and 50% for test using stratification such that training and test have roughly equal class proportions. Fit a logistic regression model to the training set to classifiy the wines as red or white. Consider the red wines as "positive" and white wines as "negative." Notice how the script makes a plot of the ROC curve showing that the AUC is around 0.99.

Script details:

- *You have fitted a logistic regression model to the wine data before in exercise 5.2.6.*
- *As before, use cross-validation. Use* `StratifiedKFold` *method to ensure that training and test sets have roughly equal class proportions (stratification).*
- *There is a function in the course toolbox called* `rocplot()` *that can make the ROC plot and compute the AUC score for you. Import it and type* `help(rocplot)` *to learn how to use it. Notice that the rocplot code assumes the score values are all distinct.*

9.2.2 Consult the script `ex9_1_2.py`. The experiment in exercise 9.1.1 is repeated, but this time it is examined how well the type of wine can be classified using only the "Alcohol" attribute. Explain how it can be seen that the alcohol contents of the wine is not very useful for classifying wine as red or white.

although the accuracy is 75%, ROC curve is close to diagonal which is close to random guessing.

Discussion:

- ◇ You have showed that using only the single attribute "Alcohol" to classify the wine as red or white performs worse than using all attributes. When using logistic regression, is it always best to use as many attributes as possible for the classification?

9.3 Tasks for the report

Continue working on the tasks for the reports as described in the previous exercise note.

1 Homework problems for this week

Problems

Question 1. Fall 2016 question 18: Considering the data in Table 1, we will use x_1 to classify whether a subject has inflammation of urinary bladder ($y = 1$) or not ($y = 0$). We will quantify how useful x_1 is for this purpose by calculating the area under curve (AUC) of the receiver operator characteristic (ROC). Which one of the ROC curves given in Figure 1 corresponds to using the feature x_1 to determine if a subject has inflammation of urinary bladder?

	x_1	x_2	x_3	x_4	x_5	y
P1	1	1	1	1	0	1
P2	0	0	0	0	0	0
P3	1	1	0	1	0	0
P4	0	1	1	0	1	0
P5	1	1	1	1	1	1
P6	0	0	0	0	0	0
P7	1	1	0	1	0	0
P8	0	1	1	0	1	0
P9	1	1	1	1	0	1
P10	0	1	1	0	1	0
P11	0	0	0	0	0	0
P12	1	1	0	1	0	0
P13	0	1	1	0	1	0
P14	0	1	1	0	1	0

Table 1: Provided in the above table are the last 14 observations of the acute inflammation data.

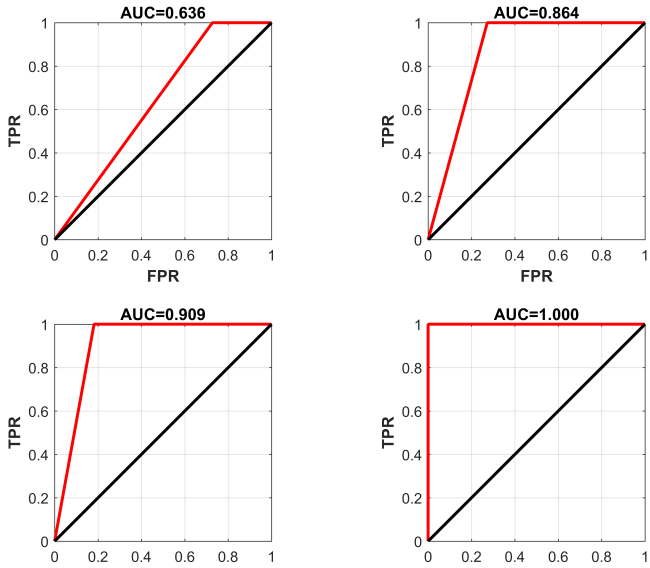

Figure 1: Four different receiver operating characteristic (ROC) curves and their corresponding area under the curve (AUC).

A The curve with AUC=0.636.

B The curve with AUC=0.864.

C The curve with AUC=0.909.

D The curve with AUC=1.000.

E Don't know.

Question 2. Fall 2018 question 13:

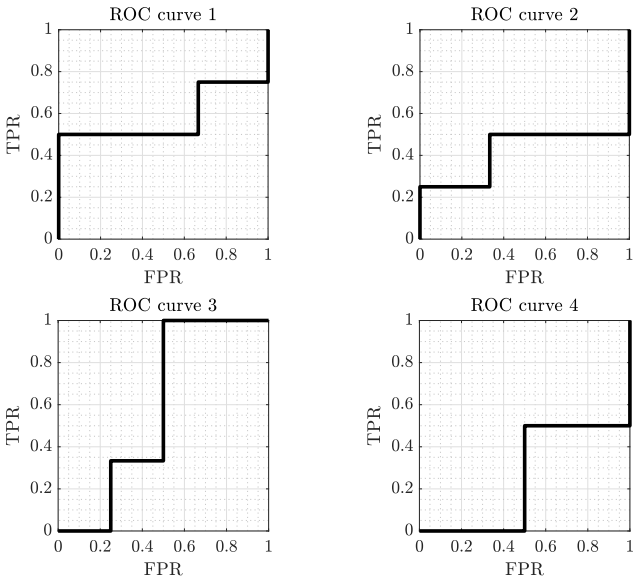

Figure 2: Proposed ROC curves for the logistic regression classifier in fig. 3.

To evaluate the classifier fig. 3, we will use the *area under curve* (AUC) of the *reciever operator characteristic* (ROC) curve as computed on the 7 observations in fig. 3. In fig. 2 is given four proposed ROC curves, which one of the curves corresponds to the classifier?

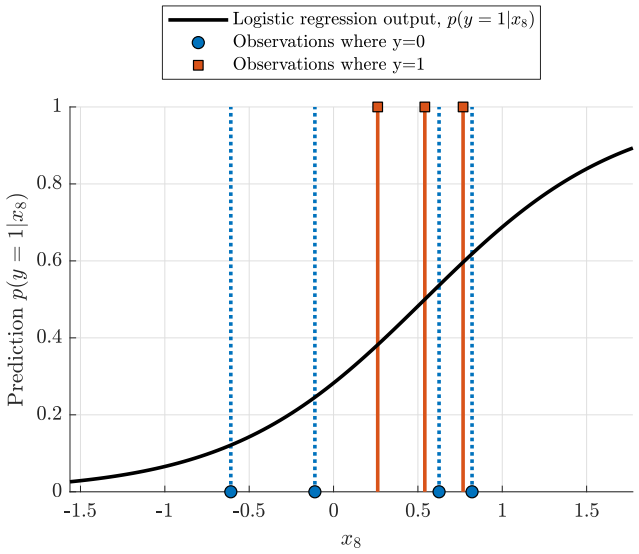

Figure 3: Output of a logistic regression classifier trained on 7 observations from the dataset.

A ROC curve 1

B ROC curve 2

C ROC curve 3

D ROC curve 4

E Don't know.

Question 3. Fall 2014 question 26: Suppose Jane wishes to apply a decision tree classifier to a binary classification problem of only $N = 4$ observations. Training and applying the decicion tree to the full dataset $\boldsymbol{X}$ and $y_1, \ldots, y_4$ gives predictions $\hat{y}_1, \ldots, \hat{y}_4$ shown in table 2.

y	$\hat{y}$
1	1
1	0
0	0
0	0

Table 2: True values y_j and predictions $\hat{y}_j$ for a decision tree classifier trained on the full data set with observed values $y_1, \ldots, y_4$.

To improve performance Jane decides to apply AdaBoost, however Jane implements AdaBoost such that instead of sampling the N elements of the training sets D_i *with* replacement, Jane samples the training sets *without* replacement, i.e. the training set D_i is simply the full dataset. Suppose Jane applies AdaBoost for $k = 1$ round of boosting, what is the resulting (approximate) value for the weights w?

A $w = \begin{bmatrix} 0.123 & 0.630 & 0.123 & 0.123 \end{bmatrix}$

B $w = \begin{bmatrix} 0.167 & 0.5 & 0.167 & 0.167 \end{bmatrix}$

C $w = \begin{bmatrix} 0.081 & 0.756 & 0.081 & 0.081 \end{bmatrix}$

D $w = \begin{bmatrix} 0.077 & 0.769 & 0.077 & 0.077 \end{bmatrix}$

E Don't know.

References