Assignment 6

Computer Science Department, University of Crete

MACHINE LEARNING - CS 577, Fall 2022

Deadline: 7/12/2022, 23:55 on e-learn (`https://elearn.uoc.gr`).

Deliverable files: Submit a zip/tar etc. file containing:

- a report in PDF with ALL answers to theoretical tasks, observations and figures produced by scripts.
- **all** Python script files written by you in the scope of the assignment.

NOTE: 10% penalty on your assignment grade will be applied in failing to follow the above instructions for deliverables or not explaining your code/ having comments!

Model Selection (Programming) [100 points]

In this exercise you will have to implement the full procedure for a learning method, from the beginning to the end. You will train multiple classifiers, exploring several hyper-parameter values for each algorithm, select the best model, and report its performance.

Data: the data in the file *Dataset*6.*A_XY.csv* (the last column contains the label, the other columns contain the features).

Classifiers to train: 2 of your choice among the ones you saw in class. You shall use existing Python implementations e.g. from sklearn for the classifiers.

Range of hyper-parameters: a range of your choice, but at least 3 values per hyper-parameter[1].

Data Preprocessing: You should run the classification models with and without data standarization (see StandardScaler from sklearn)

[1] If you use Logistic Regression play with the regularization parameter and for NB play with the alpha parameter (smoothing), denoted in sklearn

Performance Metric: ROC AUC (use sklearn implementation).

PART A [60 points]

Exploratory Data Analysis

First, you will have to explore the data: are they categorical or continuous? How many unique values exist per features? Figuring this out might give you insights to select the correct classifiers setups. Moreover, explore the features with some plots, and check how they are distributed; report on their basic statistics (e.g. mean, std for continuous and percentage of values for categorical features). In your report:

- Draw the histograms of 5 features that you consider interesting (e.g. you could plot features with high/low std/mean for the continuous ones, imbalanced features for the categorical ones or highly dependent features with the target variable Y). Explain the rationale behind your choices.
- Have a table with the statistics for the features that you selected to plot

Model Selection

In this step you should code from scratch the model selection protocol. The protocol that you will implement is the **stratified**[2], k-fold cross validation (see the recitation on model selection for further details and do not use existing implementations of cross validation).

Learning procedure

1. Split the data into stratified k folds.
2. Use cross-validation to find the best configuration: combination of classifier with its hyper-parameters.
3. Report the average performance for the best configuration across the validation (tune) folds.
4. Construct a final model trained on *all* data, using the best configuration determined in the previous step.

You are free to chose how many folds (the "k" in k-fold). In your report:

- indicate which configuration had the best performance.

[2]Search on the web what does stratification mean

- have a table with the hyper-parameters per classifier
- report how many models were trained

You must implement the following functions in **a single python file**:

def create_folds(data, k)
This function splits the data matrix into k stratified folds and returns a list of k arrays of indices that will be used for validation set in each fold. In each iteration, the remaining indices will be used as the train folds. E.g.: for k = 3 and 6 samples the output is [[0,5],[1,2],[3,4]]

def CV(data, validation_indices, configurations)
This function selects the best configuration and computes its performance via the cross validation procedure over the k validation_indices. The configurations argument is a data structure that stores information about: (i) data preprocessing (apply standardization or not) (ii) which classifier is going to be trained and (iii) the hyper-parameters of each classifier. The function must return a model instance, built from the best configuration.

PART B [40 points]

Computing the out-of-sample performance

Building on the previous exercise, let's now suppose you had previously hold-out a test set (in the file: *Dataset6.B_XY.csv*). Use this set to assess the out-of-sample performance of your model (auc). Does the out-of-sample performance differ from the performance obtained by the cross validation? If yes, why?

The ROC curve

Produce the ROC curve for the selected best model (you can use existing implementations). Compare against the trivial (naive) classifier (NOTE: this is *not* the random classifier).

Calculating the 95% Confidence Intervals

When we obtain the predictive performance for a model, we are also interested in how accurate this performance actually is. To do that, we use a technique called Confidence Intervals. In this step, we will calculate the confidence intervals of the best model by bootstrapping the hold-out set. To do so, store the performance obtained at each bootstrapped sampling of the hold-out set (you do not train additional models, you calculate the performance on the bootstrapped prediction set), and then create a histogram of the results: you can now define the 0.025 and 0.975 quantiles[3] to obtain the 95% confidence intervals. In your report, draw a plot containing:

[3] `https://numpy.org/doc/stable/reference/generated/numpy.quantile.html`

- the histogram showing the out-of-sample performances
- two vertical lines indicating the performance at the 0.025 quantile and at the 0.975 quantile
- a vertical line (of different color) indicating the performance obtained in the original hold-out set (without bootstrapping)

How do you interpret the confidence intervals that you have built?

NOTE: This estimation variant has not been shown in the class.