Assignment 5

Computer Science Department, University of Crete

MACHINE LEARNING - CS 577, Fall 2024

Deadline: 27/1/2025, 23:55 on e-learn (`https://elearn.uoc.gr`).

Deliverable files: Submit a zip/tar 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.

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 *Dataset5.A_XY.csv* (the last column contains the label, the other columns contain the features).

Classifiers to train: Two 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)

Performance Metric: ROC AUC (use sklearn implementation).

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

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 k-fold cross validation (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.
- 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(arguments)
This function splits the data matrix into k stratified folds.

def CV(arguments)

This function selects the best configuration and computes its performance via the cross validation procedure. Hint: each configuration can be 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.

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: *Dataset5.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).