MLCB 2023
25 March 2023

Assignment #2

Hepatitis C is a viral infection that causes inflammation of the liver. It is caused by the hepatitis C virus (HCV), which is primarily spread through blood-to-blood contact and can lead to serious health problems, including liver damage, cirrhosis, and liver cancer. The treatment for hepatitis C has improved significantly over the years, but there is still a need for better prediction of disease progression and treatment outcomes. In this context, Machine Learning (ML) methods offer a promising approach to analyzing various features and laboratory test values to effectively diagnose Hepatitis C patients and generate laboratory diagnostic protocols.

In the provided dataset (Hepatitis_C.csv), you will find 204 samples of Hepatitis C patients and healthy blood donors. For each patient 12 features are available. The dataset index corresponds to the patient's ID. The features are the following:

1. Age (in years),
2. Sex (f=1, m=0),
3. The rest of the features correspond to laboratory test data (ALB, ALP, ALT, AST, BIL, CHE, CHOL, CREA, GGT, PROT).

The goal of this assignment is to create a complete ML pipeline to process the data and classify successfully future patients according to the label column. 'label'=1 corresponds to a Hepatitis C patient (positive class), and 'label'=0 corresponds to a healthy blood donor (negative class). In detail, the steps to follow for this exercise are the following:

1) Build a *nested Cross Validation* (nCV) pipeline to compare systematically the expected performance of the following classification algorithms on unseen data: Logistic Regression (LR), Gaussian Naive Bayes (GNB), K-Nearest Neighbors (kNN), Linear Discriminant Analysis (LDA), Support Vector Machines (SVM). For the outer loop use K=5 folds and for the inner loop L=3 folds. Select a suitable metric for training and model selection (you may experiment with more than one) in the inner loop. For the evaluation of algorithms (outer loop), use the Mathews Correlation Coefficient (MCC) to be able to compare. When you evaluate models derived from unbalanced datasets, except MCC you should also compute other important metrics such as the Balanced Accuracy, F1 score, F2 score, Sensitivity (Recall), Precision, Average Precision, Specificity, Negative Predictive Value, etc.. Declare as "winner" the classification algorithm that achieves the highest average test MCC performance in 10 trials of nCV (that means the average over 10x5=50 outer loop test folds).

2) After finding the winner algorithm using nCV (step 1), *use the whole dataset* and simple cross-validation with 5 folds (CV(5)) to determine the “final model” (with optimal hyperparameters) to deploy in the field. We will use your final models and a *hold-out* set (not accessible to you) to declare the “class-best” model! *Its creator will be awarded automatically 50% bonus points for winning the competition (see below)!*

What to submit: By the deadline, you should submit using eclass a compressed file (with filename *YourLastName_ID#_Assignment1.zip*) that should include:

1) A *technical report* in the form of a journal paper with at least the following sections:
   a. Abstract
   b. Introduction (problem statement, significance),
   c. Methods (dataset description, preprocessing, pipeline structure, pipeline stages description),
   d. Results and Discussion (main results and discussion). If you have tried the Bonus parts report their results in separate Results subsections
   e. Conclusions (summary of main findings, lessons learned, limitations of the analysis suggested further research),
   f. References,
   g. Supplementary Material (secondary but noteworthy results).
   Use well-designed Figures and Tables as needed. For example, you should include a Figure with the boxplots of the evaluation metrics over the 50 nCV outer loop folds for all algorithms. All Figures and Tables should have titles and clearly labeled axes.

2) The *notebook(s)* with your well-commented code (*it should be in R or python*). Your notebook(s) should easily reproduce the results in your Figures. It should be straightforward to reproduce all your results (*this will be an important part of your evaluation!*). Please include all necessary instructions on dependencies etc. required to run your notebook(s) in the Supplementary Material section of your technical report as an Appendix.

Bonus parts:

1) (for 50% more points) After you complete the main part to establish a baseline, apply different *feature selection* methods (at least two) to choose the top 5 features and evaluate if they can improve the results of the main part. Discuss your findings.

2) (for 50% more points): Apply different data oversampling methods (at least two) for the examples of the minority class and evaluate if they can improve the results of the main part (baseline). Discuss your findings

3) (for 50% more points) Repeat the baseline analysis (main part of the assignment) using a second programming language (R or python).

Bonus parts of assignments are optional. Not doing them will not affect your final course grade. However, if you consistently accumulate “Bonus points” in assignments (i.e., you demonstrate consistent extra effort) your course grade may be boosted by as much as a whole mark (e.g., a 7.2 may become 8, a 4.2 may become 5, etc.)

Deadline: Submit the deliverables by **Monday 24 April** using eclass. Late assignments will not be graded.

Hints and Tips

- If you use Python and you are not very familiar with scikit-learn for machine learning we recommend that you use the ATOM package which was developed to make things easier for newcomers to ML pipeline building (https://tvdboom.github.io/ATOM/v5.1/). If you decide to use ATOM you still need to create the outer loop of nCV using scikit-learn. Keep in mind that there is no developed online community to solve certain issues.

- If you choose to develop your nCV implementation using scikit-learn directly, consider using Optuna (https://optuna.org/#key_features) for hyperparameters tuning in the inner loop. Please note that new ATOM versions (>5.0v) use Optuna by default and that makes hyperparameter tuning very easy. On the other hand, scikit-learn gives you full flexibility e.g., more hyperparameters to tune, etc. One of the benefits of using Optuna is the significant acceleration it provides. Using up to 50 trials for hyperparameters tuning should suffice.