MLCB 2024
2/4/2023

Assignment #2

The dataset "Diabetes.csv," available via cloning (as in the previous assignment) in the repository "https://github.com/MLCB2024Class/Assignment_2.git," contains medical records of 508 patients. They contain a binary target variable called "Outcome" along with eight (8) features. You should use all provided medical records in building your predictive models.

The objective of this assignment is to explore and analyze the dataset and then implement a complete *nested Cross Validation* (nCV) machine learning pipeline following object-oriented programming (OOP) principles. Your pipeline should identify the best machine learning (ML) algorithm that classifies successfully future patients according to the Outcome variable. Subjects with 'Outcome'=1 correspond to Diabetes patients (positive class), and those with 'Outcome'=0 correspond to healthy controls (negative class). The steps to follow for this exercise are detailed below:

Task 1. Exploratory Data Analysis [20 points] :

Create a notebook in Python (or R markdown) and perform a brief but concise exploratory data analysis. This should include (at least):

1) Dataset Overview and Descriptive Statistics: For example, examine the shape and format of the dataset, column data types, duplicate rows and/or missing values, outliers, and *class balance.*
2) Feature Assessment and Visualization: For example*, Interpret the "physics" of the features, Box Plots of their distributions, correlation analysis between features, dimensionality reduction with PCA, and plotting to get a rudimentary sense of the feature space and class separability.*
3) Data Quality Evaluation: For example, *assess possible complicating factors from the results of the previous two stages that might compromise an ML task built upon this dataset and how you propose to deal with them.*

Task 2. Intuition & Preparation for the nested Cross Validation implementation [10 points]:

Before implementing your nCV pipeline (or any class object), it is beneficial to grab a pen and paper. Imagine you have your components, that is, the 508x8 dataset and a series of untrained classifiers. Sketch out the abstraction of the nCV pipeline logic (Showcase the stages involved in terms of input, output, and role. Simple boxes (for pipeline steps) and arrows (for the flow of the logic) with descriptive text are enough.

a) Sketch the abstraction of one iteration of nCV. You should use 5 splits in the outer loop and 3 in the inner loop. Imagine the journey your data goes through and pay extra attention to avoid **data leakage**.
b) If you prefer a more polished approach than pen and paper, you can create diagrams using tools like https://app.diagrams.net/.

Task 3. Nested Cross Validation implementation [70 points]:

1) Create a Python (or R) file to construct a nested Cross-Validation (nCV) class equipped with essential methods to systematically train and estimate the generalization performance of classification algorithms on unseen data. This pipeline should accept as input at least the dataset, a list of classifiers along with their corresponding hyper-parameter spaces for tuning, as well as the number of outer and inner loop folds, called K and L, respectively. Choose a suitable metric for training and model selection in the inner loop. You can experiment with different optimization metrics for hyperparameter tuning and different methods for best trial selection in the inner loop. Justify your choices. For the assessment of classifiers using imbalanced datasets, you should compute in the outer nCV loop test set statistics of crucial metrics such as the Matthews Correlation Coefficient (MCC), Balanced Accuracy, F1 score, F2 score, Sensitivity (Recall), Specificity, Precision, Average Precision, Negative Predictive Value, Precision-Recall curves, etc. *Keep in mind that since the dataset may, in general, be imbalanced, the folds you create should all reflect the dataset's class imbalance.*

2) Import your nested cross-validation (nCV) class into a new Python (or R) notebook and perform M=10 iterations of nCV. For each nCV iteration (round), utilize K=5 folds for the outer loop and L=3 folds for the inner loop to estimate the generalization performance of the folloing algorithms: Logistic Regression (LR), Gaussian Naive Bayes (GNB), k-Nearest Neighbors (kNN), Linear Discriminant Analysis (LDA), Support Vector Machines (SVM). You can use regularized versions of these algorithms. You should define your own hyperparameters' space for each algorithm. Assess the classification algorithms based on statistics of performance metrics aligned with the intended purpose of the ML classifier in the field**,** such as e.g. high Recall, and declare as "winner" the algorithm that achieves the best purpose-aligned balance of these metric(s) over 10 repetitions (rounds) of nCV. That means considering the median metric(s) over 10x5=50 outer loop test folds. Fully justify your approach in performing algorithms' comparison. In cases where the median performance of two (or more) top algorithms is similar, use sound statistical practices, e.g. considering the 95% Confidence Interval of the medians in your "winner" classifier selection.

3) After finding the winner algorithm using nCV (step 2), *use the whole dataset* and simple cross-validation with 5 folds (CV(5)) to determine its "final model instance" (with optimal hyperparameters) to deploy in the field and save it in a .pkl file in a ./models directory. We will use your final trained winner model instance and a (secret) *hold-out* data set 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 e-class a compressed file (with filename *YourLastName_ID#_Assignment2.zip*) that should include:

1) A *technical report* in the form of a scientific paper. You can use LibreOffice writer, Word, or LaTeX, a powerful typesetting software, for document preparation. The report should contain at least at least the following sections:
   a. Abstract (concise summary of the problem statement and the results of the technical report)
   b. Introduction (problem statement, significance, related work, technical approach),
   c. Materials and Methods (figure from task 2., dataset description, data exploration methods, preprocessing, nCV pipeline structure, pipeline stages description),
   d. Results and Discussion (Exploratory Data Analysis, nCV results and discussion). If you have tried any Bonus part, report its results in a separate subsection.
   e. Conclusions (summary of main findings, limitations of the analysis, lessons learned, suggested further research),
   f. References (to relevant scientific articles, resources, web pages, using the IEEE articles style)
   g. Supplementary Material (secondary but noteworthy results (Figures, Tables etc) to which you refer in the report).

   Use well-designed Figures and Tables as needed. For example, you should include a Figure for each evaluation metric with the boxplots of the different algorithms over the 50 nCV outer loop folds, as well as your graph from Task 2. All Figures and Tables should have titles, clearly labeled axes, self explanatory captions and should be numbered. Follow good statistics practices in presenting and reporting your results. E.g. If you use barplots to show medians (means) and IQRs of performance metrics, you should also include in the Figure captions the sample size used and the CIs and Standard Errors of the medians (means).

2) The *notebook(s) and script(s)* with your well-commented code (*it should be in Python or R*). 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 consideration of your evaluation! Think that you are submitting a paper for publication, and you want the reviewers to be able to easily reproduce and evaluate your work. Make it easy for them to get impressed!*). Always use seeds to avoid stochasticity and ensure reproducibility across runs. 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.

3) Your selected trained final model instance in .pkl format in the ./models directory. **Ensure that your finalized model, when loaded, is able to execute all preprocessing steps seamlessly on any incoming unseen data, with the format provided in "Diabetes.csv". If this functionality is not present, you should develop an evaluation script in either Python or R that performs the necessary preprocessing steps on the given data and then loads your trained chosen model for testing purposes.**

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 whether they can improve the main part's results. Discuss your findings and compare the methods used.

2) (for 50% more points): Apply different methods to balance the classes in the dataset (at least two) and evaluate if they can significantly improve the results of the main part (baseline). Discuss your findings and compare the methods used.

Bonus parts of assignments are optional. Not doing them will not affect your final course grade. However, if you consistently accumulate "Bonus points" in the class assignments (i.e., you demonstrate consistent extra effort), your course grade will be boosted by as much as a whole mark (e.g., a 7.2 may become 8, a 4.2 may become 5, etc.) and if you ask for a reference letter in the future this extra effort will be noted.

An important note on LLM usage for coding: Learning programming/Data Science isn't about mastering a single thing. It's mostly about gathering bits and pieces of knowledge and fitting them together like a puzzle. Once you have built "muscle memory" from practicing with assignments like this one, you can easily use your knowledge to tackle similar problems in the future. However, if you rely on LLMs without 100% understanding of the code they provide, you enter an infinite cycle where you will keep returning to it for the same tasks over and over again, and each time it is varied, often error-prone solutions will only lead to further confusion. It is much better to use them as a teaching assistant if needed, although the sci-kit learn package provides great documentation with in-depth explanations for all of the tools it provides, and it is a great idea to familiarize yourself with its use as early as possible. After all, assignment solutions that utilize copy/pasted LLM code are most often distinguishable and we have the experience to spot them!

If you use LLM assistance you should disclose it, as it is now becoming a standard practice also when submitting publications. In that case, please add a small section at the end of your report explaining which LLM you used and for what purposes (tasks). Feel free to discuss your experience.

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.

- To create confidence intervals for non-parametric statistics such as the median, you can utilize Bootstrap resampling. You can either implement it yourself as part of your class or utilize a library such as scipy for Python.

Deadline: Submit the deliverables by **Monday, April 29th** using e-class. *Late assignments will not be graded. Start as soon as possible!*