Machine Learning in Computational Biology

7/4/2025

Assignment #2

The dataset "breast_cancer.csv" is available via forking the repository: https://github.com/MLCB2025Class/Assignment-2.git (as per the previous assignment). It contains 512 tumor samples. The dataset contains a binary target called "diagnosis" along with thirty (30) features. All the provided samples should be used in building your predictive models.

The objectives of this assignment are to:

1. Explore and analyze the dataset.
2. Implement a complete repeated Nested Cross-Validation (rnCV) machine learning pipeline following object-oriented programming (OOP) principles (required for this assignment).
3. Your ML pipeline should identify the best machine learning (ML) algorithm that is expected to correctly classify malignant vs. benign breast cancer tumor cells based on the "diagnosis" variable.

The steps to follow for this assignment are detailed below:

Task 1 - Exploratory Data Analysis

About the dataset

The provided dataset consists of measurements derived from digitized images of fine needle aspirates (FNA) of breast masses. Each sample is represented by 30 numerical features describing the physical characteristics of **cell nuclei**, aiding in the differentiation between **malignant (cancerous)** and **benign (non-cancerous)** samples. These features are based on ten fundamental properties: ***radius*** (mean distance from center to perimeter points), ***texture*** (standard deviation of gray-scale values), ***perimeter***, ***area***, ***smoothness*** (local variation in radius lengths), ***compactness*** (calculated as perimeter^2/area - 1.0), ***concavity*** (severity of concave portions of the contour), ***concave points*** (number of concave portions of the contour), ***symmetry***, and ***fractal dimension*** (a measure of complexity using coastline approximation). Each of these properties is characterized in three ways: ***mean value***, ***standard error***, and ***worst*** **(largest mean of the three largest values)**, resulting in a total of 10x3=30 computed features per sample. The dataset provides an opportunity to explore statistical patterns, apply classification models, and assess predictive accuracy in medical diagnostics.

About exploration

Create a notebook in Python and perform a brief but concise exploratory data analysis. This should, *at least*, include:

1. Dataset Overview and Descriptive Statistics

Examine the shape and format of the dataset, column data types, duplicate rows and/or missing values, outliers, and *class imbalance*. How can you handle missing values?

2. Feature Assessment and Visualization:

Explore the features of the dataset, create box plots of their distributions, heatmaps, perform correlation analysis between features, and use dimensionality reduction techniques such as PCA. Plots should be used to get a rudimentary sense of the feature space and class separability.

Task 2 – Intuition & Preparation for the nCV implementation

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 512x30 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.

1. 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.
2. 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

Step 1 - Create the repeated nested cross-validation class

Create a Python file to construct a Repeated Nested Cross Validation class equipped with essential methods to **systematically** train and estimate the **generalization performance** of **classification** algorithms on unseen data. The pipeline should accept as inputs, at least:

1) The dataset
2) A list of estimators
3) The respective hyperparameter spaces (defined by you) for each estimator that will be used for tuning
4) Number of rounds for the nCV (R=10)
5) Number of outer fold loops (N=5)
6) Number of inner fold loops (K=3)

[*Keep in mind that:*

1. *In case a dataset is imbalanced, the folds you create should all reflect the whole dataset's class imbalance.*
2. *Your results need to be reproducible! Use specific seeds in order to avoid stochasticity.]*

Regarding the inner loop, choose a suitable metric for training and model selection. You can experiment with different *optimization metrics* for hyperparameter tuning and *different methods for best trial selection*.

Regarding classifier assessment using *imbalanced* datasets, you should compute in the outer nCV loop test set statistics of crucial metrics such as Matthews Correlation Coefficient (MCC), Area Under the Curve (AUC), Balanced Accuracy (BA), F1 score, F2 score, Recall, Specificity, Precision, Precision Recall-AUC (PRAUC), Negative Predictive Value (NPV), etc. It is up to you which metrics you will include in your analysis. Keep in mind that by using multiple metrics, you can get a clear idea of the algorithm's performance.

Step 2 - Assess generalization performance and select the best model type / winner algorithm.

Import your nested cross-validation class into a new Python *notebook* and perform R=10 iterations (also called "rounds") of nCV (repeated nested CV, or rnCV). For each rnCV round, utilize N=5 folds for the outer loop and K=3 folds for the inner loop. This will serve to estimate the expected **generalization performance** of the following algorithms:

Logistic Regression (LR) with Elastic Net Regularization, Gaussian Naive Bayes (GNB), Linear Discriminant Analysis (LDA), Support Vector Machines (SVM), Random Forests (RF), and LightGBM.

You may use regularized versions of these algorithms, and you should define your own hyperparameters' space for each one of them (as per the previous assignment). **Assess the classification algorithms** based on **statistics** of performance metrics aligned with the intended purpose of ML classifier in the field, such as e.g., "high AUC" or "high MCC" 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 metrics over all outer loop test folds (5-fold nested CV x 10 repetitions = 50 points for each metric). Fully justify your approach when comparing the algorithms. 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.

Task 4 - Train the final model instance

Step 1 - Select the best hyperparameters of the winner algorithm using simple CV

After identifying the "winner algorithm" (the algorithm with the best **generalization performance**) using rnCV (task 3 - step 2), perform 5-fold cross-validation (CV(5)) on the entire dataset to determine the **optimal set of hyperparameters**.

Step 2 - Train the final model instance with the whole dataset

Train the final model instance using the optimal hyperparameters **on all available data**, and save it as a .pkl file in the ./models directory for deployment in the field. It should be ready to perform *all needed transformations on unseen data* and *give predictions* (as per the previous assignment).

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)!*

Bonus parts

1. (for 50% more points) After you complete the main part to establish a baseline, **apply different feature selection (FS) methods** (at least two) to choose the top 5-10 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 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 the instructor for a reference letter in the future this extra effort will be emphasized.

Deliverables

By the deadline, you should submit a **10 - 15 page report** (11pt, 1.15 space) using e-class that includes:

1. The link to your repository. The architecture of the repository should be the same as in the previous assignment. A folder for notebooks, a *src* folder that will include your nested cross-validation class along with other Python files that you deem necessary, and a models directory to store your final selected model instance in .pkl format.
2. A technical report in the form of a brief scientific paper. You can use LibreOffice Writer, MS Word, or LaTeX, a powerful typesetting software, for document preparation. The report should contain the following sections:
    a. Abstract (concise, max 200 words summary of the problem statement and the results of the technical report and their impact)
    b. Introduction (problem statement, significance, related work, technical approach, contributions),
    c. Materials and Methods (figure from task 2, dataset description, dataset exploration, preprocessing, rnCV pipeline structure, pipeline stages description),
    d. Results and Discussion (Exploratory Data Analysis, Algorithms comparison results and discussion). If you try any Bonus part, report its results in a separate subsection. You can add up to 5 pages to the report for every bonus part you try.
    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. AI usage disclosure statement (see below)

Hints and Tips

- If you use Python and you are not yet 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 hyperparameter tuning in the inner loop. Please note that new ATOM versions (>5.0v) use Optuna by default, which 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 hyperparameter 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.

- Many classifiers may return warnings (e.g., convergence warnings) during training. Once you have verified that your code works correctly, you can safely suppress these warnings to keep your notebook clean and readable for both yourself and whoever will review it. This helps ensure that everything runs smoothly without unnecessary clutter in the output.

Grading Approach: Code, including documentation and results generation 70%, completeness and quality of the report 30%.

Deadline: Submit the deliverables in a zip file with a filename format <Assignment2_YourLastFirstName_ID> by **May 9th, using e-class.** Late assignments will not be graded. Start as soon as possible!

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 will find it easy to use your knowledge and tackle similar problems in the future. However, if you rely from your first steps on LLMs without a 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 LLMs as a teaching assistant as needed, although the sci-kit learn package provides great documentation with in-depth explanations for all 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 we feel we need to evaluate your work further, you may be called for an oral examination based on the assignment.

If you use LLM assistance, you should disclose it, as it is now becoming a standard practice also when submitting a publication. 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. If this statement is missing from your report it will imply that LLMs were not used at all for any task.

Keep in mind that this is not a group assignment. Each student should work independently. You are allowed to discuss your technical approaches, but you should not share code. If we find evidence of code sharing, all parties involved will receive a zero grade.