A new pipeline for cross-validation fold-aware machine learning prediction of clinical outcomes addresses hidden data-leakage in omics based 'predictors'.

Marcelo Hurtado[1,2], Vera Pancaldi[1,2]

[1] Centre de Recherches en Cancérologie de Toulouse, CRCT, Université de Toulouse, INSERM, CNRS, Toulouse, France.

[2] Equipe labellisée Ligue Contre le Cancer

Abstract

Machine learning approaches are increasingly applied to high-dimensional biological data in which features are often dataset-dependent. In many omics workflows, features are computed using information derived from the entire dataset, such as correlations between variables, clustering structures, or enrichment scores. We refer to these as *global dataset features,* defined as features whose computation depends on properties of the full dataset, including the number of samples, the relationships between samples, or global statistical summaries. In such cases, standard validation strategies can fail, especially when evaluating on independent datasets, due to information leakage that leads to overly optimistic performance estimates. This issue is particularly relevant in omics analyses, where features are derived from correlation-based methods, enrichment analyses, or model-driven transformations that implicitly use information from the full dataset. To address this challenge, we present *pipeML*, a flexible and modular machine learning framework designed to support leakage-free model training through custom cross-validation fold construction. *pipeML* enables users to recompute global dataset features independently within each cross-validation fold, ensuring strict separation between training and test data, while preserving compatibility with a wide range of machine learning algorithms for both classification and survival tasks. The framework integrates feature selection, repeated and stratified cross-validation, model stacking, and comprehensive performance evaluation, supporting advanced validation schemes such as leave-one-dataset-out analysis. Using multiple real-world biological datasets, we demonstrate that *pipeML* enables leakage-free model evaluation when global dataset features are used. We argue that overestimation of model performance in the cross-validation setting can lead to overoptimistic expectations for validation on independent datasets. By explicitly addressing data leakage and offering a transparent, modular workflow, *pipeML* provides a robust solution for developing and validating machine learning models in complex biological settings.

The *pipeML* R package as well as a tutorial and documentation page is available at ***https://github.com/VeraPancaldiLab/pipeML***. The code to reproduce the analysis and figures is available on github at ***https://github.com/VeraPancaldiLab/pipeML_paper***.

Keywords: machine learning, cross validation, data leakage, overfitting, global dataset features

Contact: marcelo.hurtado@inserm.fr or vera.pancaldi@inserm.fr

Introduction

Machine learning methods are increasingly applied to high-dimensional biological data to identify predictive signatures and stratify patients. In many biological analyses, however, features are not independent measurements but are derived from relationships across multiple samples. Examples include pathway activity scores, gene set enrichment statistics, transcription factor activity estimates, or cell-state aggregates inferred from gene expression profiles. These features are often computed using correlations, enrichment statistics, clustering structures, or other transformations learned from the data itself. We refer to such representations as *global dataset features*, defined as features whose computation depends on properties of the entire dataset, including the number of samples, their relationships, or global statistics summaries. The use of *global dataset features* creates important challenges for machine learning model evaluation. When such features are computed on the full dataset prior to model training, information from samples that later appear in test partitions can inadvertently influence the feature representation. As a result, standard cross-validation strategies no longer maintain strict independence between training and validation data, leading to overly optimistic performance estimates. This issue is particularly critical in biomedical applications, where datasets are often small, heterogeneous, and collected across multiple cohorts, making reliable performance estimation essential.

Despite its importance, leakage-aware feature construction is rarely addressed in existing machine learning pipelines, which typically assume that features are fixed and independent across samples. As a consequence, feature construction is frequently treated as a preprocessing step applied before cross-validation, implicitly allowing information from validation samples to influence model training. One possible strategy to avoid information leakage is to learn global dataset features using a fully independent dataset and then project them onto the data used for model training and evaluation (Lapuente-Santana, 2021, 2024). While this approach is effective, it requires large, representative, and biologically aligned independent datasets with the target cohort, conditions that are often not met in practice, particularly in omics studies characterized by limited sample sizes, batch effects, and context-specific biological signals.

There is therefore a pressing need for flexible machine learning frameworks that explicitly support

fold-aware, leakage-free feature generation while maintaining robust model training, evaluation, and interpretability. To address these issues, we introduce *pipeML*, an R package that provides a flexible and modular framework for leakage-aware machine learning analysis of high-dimensional biological data. *pipeML* supports fold-aware feature construction, enabling *global dataset features* to be recomputed independently within each training fold during cross-validation. This design preserves strict separation between training and validation data while maintaining compatibility with a wide range of machine learning algorithms for both classification and survival tasks. It integrates feature selection, repeated and stratified cross-validation, model training and validation, benchmark of algorithms, hyperparameter tuning, prediction, and interpretability of features through SHAP analyses within a unified pipeline. It further supports realistic model evaluation and advanced validation strategies, including cross-dataset generalization such as leave-one-dataset-out analysis (***Figure 1***).

While several machine learning frameworks, including scikit-learn, AutoML and FLAML provide powerful tools for automated model development, many are primarily developed in Python and do not natively integrate with the R/Bioconductor ecosystem, where a large proportion of bioinformatics tools for omics data analysis are implemented. As a result, researchers working within R-based bioinformatics workflows often face challenges when incorporating leakage-aware machine learning pipelines into their analyses. *pipeML* addresses this gap by providing a framework designed specifically for the R environment, facilitating seamless integration with existing bioinformatics packages.

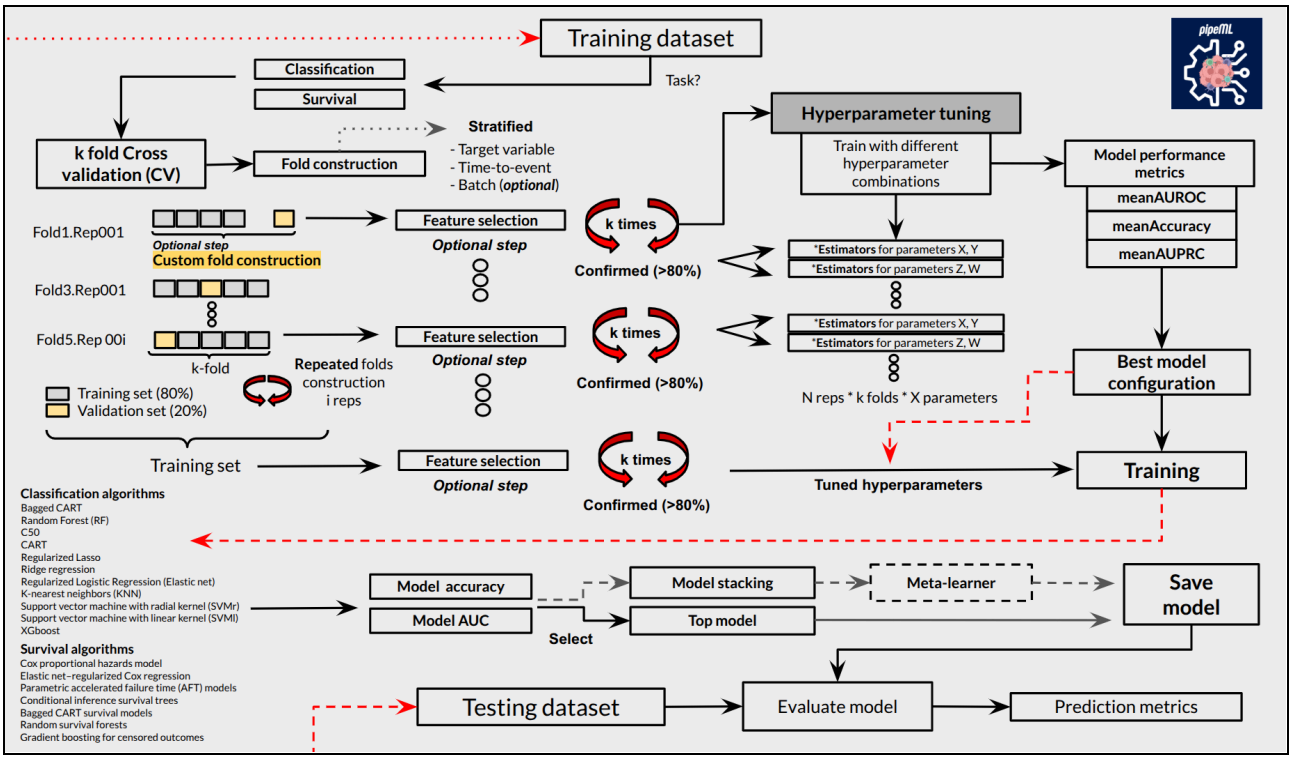


Figure 1. pipeML framework

Results and discussion

pipeML implements repeated and stratified k-fold cross-validation, optional model stacking with meta-learners, and cohort-aware validation strategies such as leave-one-dataset-out (LODO) analysis. Feature selection can be performed using repeated Boruta runs to identify robust predictors, while model training leverages the *caret*, *tidymodels*, *parsnip* and *censored* ecosystems for hyperparameter tuning and performance optimization across multiple classification and survival algorithms. *pipeML* further supports customized cross validation fold construction functions, enabling feature engineering steps that must be repeated within each fold, as well as tunable feature-generation parameters that can be optimized jointly with model hyperparameters. Finally, the package provides built-in tools for prediction, threshold optimization, performance visualization (RO, PR and Kaplan-Meier curves), and model interpretability through variable importance via SHAP values, making it suitable for transparent and reproducible machine learning analyses in translational and systems biology studies.

Model selection and pipeline construction for classification and survival tasks.

To demonstrate the application of *pipeML* for classification tasks, we applied it to the Breast Cancer

Wisconsin Dataset (Wolberg et al. 1990; Zhang J. 1992; Blake CL, et al. 1998) available in the *mlbench* R package (v2.1.6). This dataset contains clinical measurements from breast tumor samples and is commonly used for evaluating binary classification algorithms. The predictors include nine cytological characteristics computed from digitized images of fine needle aspirates of breast masses, including clump thickness, uniformity of cell size and shape, marginal adhesion, epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. The target variable represents tumor diagnosis, with two classes: benign and malignant.

Prior to model training, a binary target variable was created, where malignant tumors were encoded as 1 and benign tumors as 0. For example purposes, the dataset was subsequently divided into training and test sets using a stratified partition, with 70% of the observations used for model training and the remaining 30% reserved for testing.

Using *pipeML*, we performed automated model training and evaluation on the training dataset. Model selection was conducted using repeated k-fold cross-validation (5 folds with 10 repetitions) to obtain robust performance estimates and assess model stability. Candidate models were evaluated using the area under the receiver operating characteristic curve (AUROC) as the primary performance metric, and hyperparameters were tuned automatically within the pipeline procedure (***Figure 2; Supplementary Table 1***).

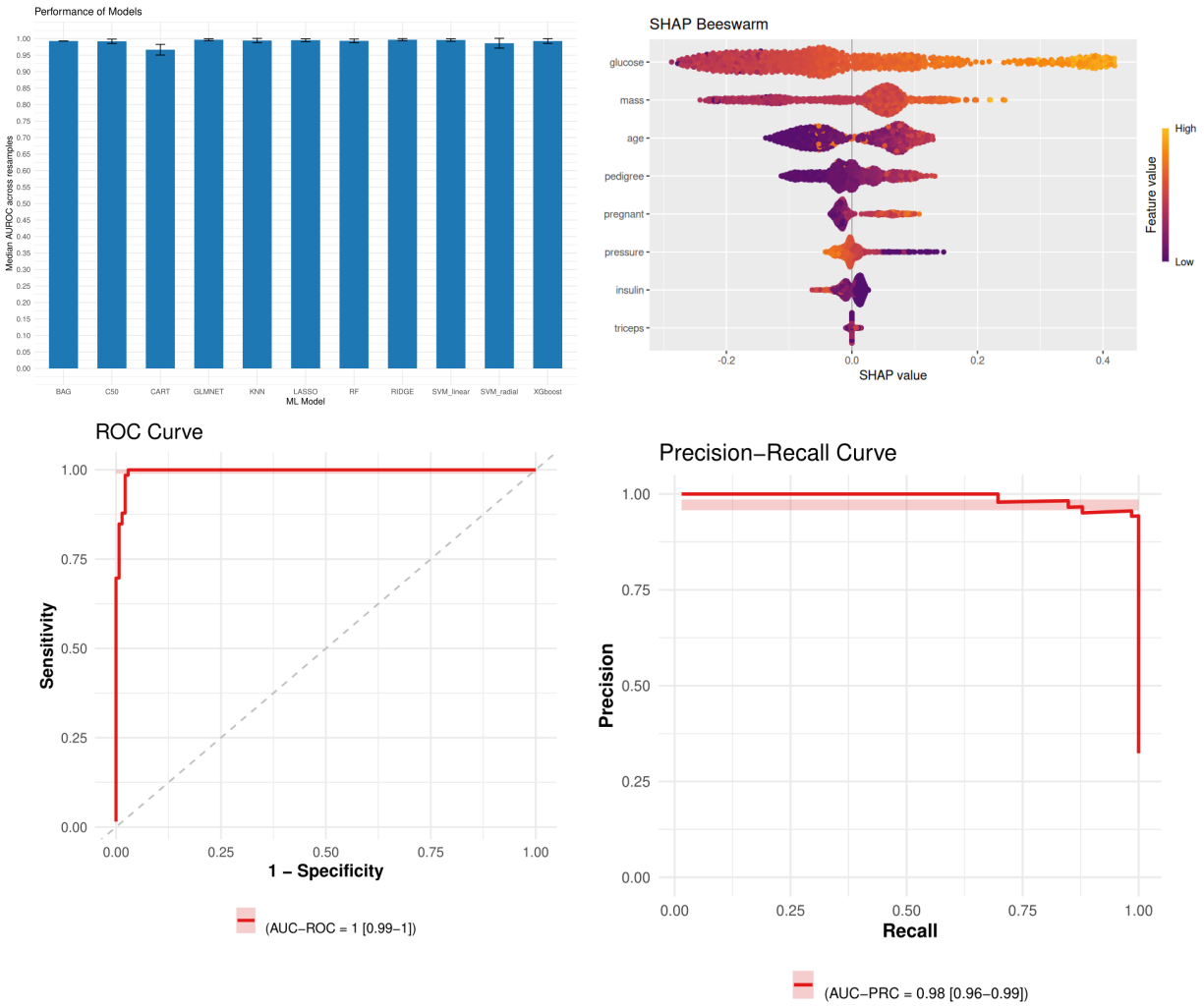


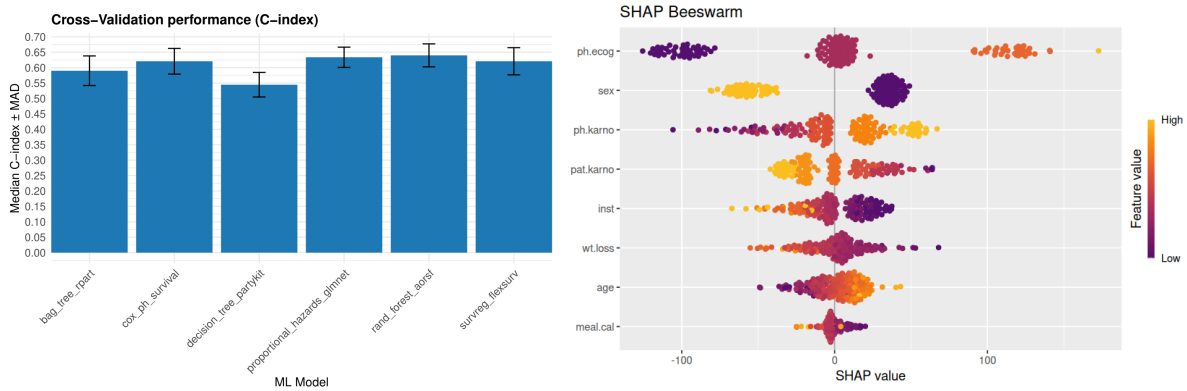
Figure 2. Performance and interpretation of classification models trained on the Breast Cancer Wisconsin dataset using a 70% training and 30% test split. (A) Cross-validated AUROC values for different classification algorithms evaluated on the training set. (B) SHAP values showing the relative contribution of predictive variables to the model predictions. (C) Receiver operating characteristic (ROC) and precision–recall (PR) curves evaluating model performance on the independent test set.

For survival analysis tasks, we applied *pipeML* to the lung dataset (Loprinzi CL, et al. 1994) from the *survival* R package (v3.8.3). This dataset contains 228 patients with advanced lung cancer, including clinical variables such as age, sex, ECOG performance score, and other measurements relevant to prognosis. Prior to analysis, observations containing missing values were removed. The survival outcome

is defined by the time to event (time) and the censoring indicator (status, 1 = death, 0 = censored). The dataset was subsequently divided into training and test sets using a stratified partition based on the event indicator, with 70% of the observations used for training and 20% reserved for testing.

Using *pipeML*, automated model training and evaluation were performed on the training data. Model selection was carried out using repeated k-fold cross-validation (5 folds with 10 repetitions) to obtain robust estimates of model performance and stability. Hyperparameters for candidate models were tuned automatically during training, and model performance was evaluated using the concordance index (C-index), a standard metric for assessing predictive accuracy in survival analysis (***Figure 3; Supplementary Table 1***).

Both models, classification and survival were analyzed using SHAP values to retrieve the feature importance for the final model obtained from the *pipeML* training procedure using every fold of the cross validation (***see Materials and methods***).



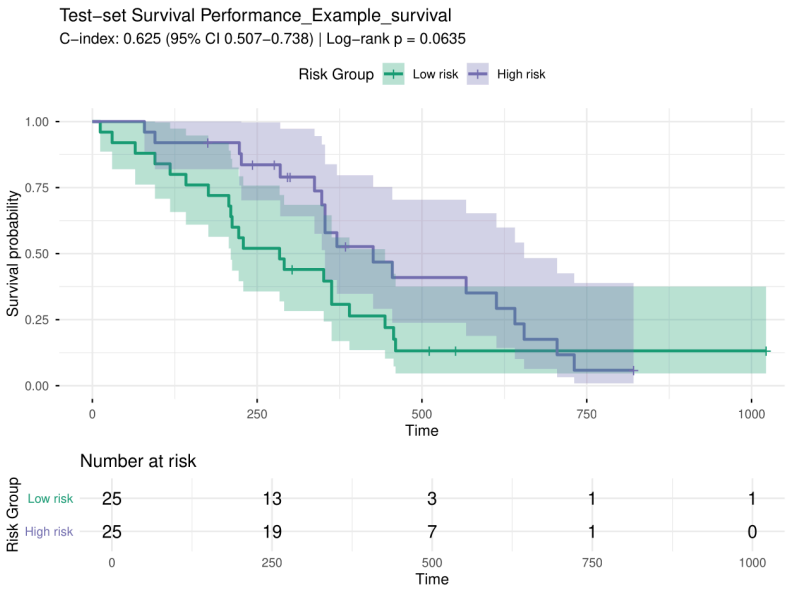


Figure 3. Performance and interpretation of survival models trained using a 70% training and 30% test split. (A) Cross-validated concordance index (C-index) for different survival analysis algorithms evaluated on the training set. (B) SHAP values showing the contribution of predictive variables to the model predictions. (C) Kaplan–Meier survival curves illustrating risk stratification performance on the independent test set.

***pipeML* achieves comparable median AUROC and AUPRC to standard machine learning frameworks.**

We compared the predictive performance of common machine learning models across three frameworks: *pipeML*, H2O AutoML (v3.44.0.3), and a Python scikit-learn implementation. Using the Sonar dataset from the R package *mlbench* (comprising 111 positive cases and 97 negative cases), we evaluated three models representing different families: GLM (Generalized Linear Model) as the linear model, Random Forest (RF) as a nonlinear ensemble tree-based method, and XGBoost as a gradient boosting algorithm capable of high predictive performance. Median AUROC and AUPRC were assessed over repeated cross-validation experiments (5 folds with 10 repetitions). *pipeML* achieved comparable median performance to both H2O AutoML and scikit-learn across all model families. Error bars, representing the median absolute deviation, indicate consistent performance across repeats (***Figure 4***). These results demonstrate that *pipeML* provides reliable predictions and in accordance with well-established machine learning pipelines.

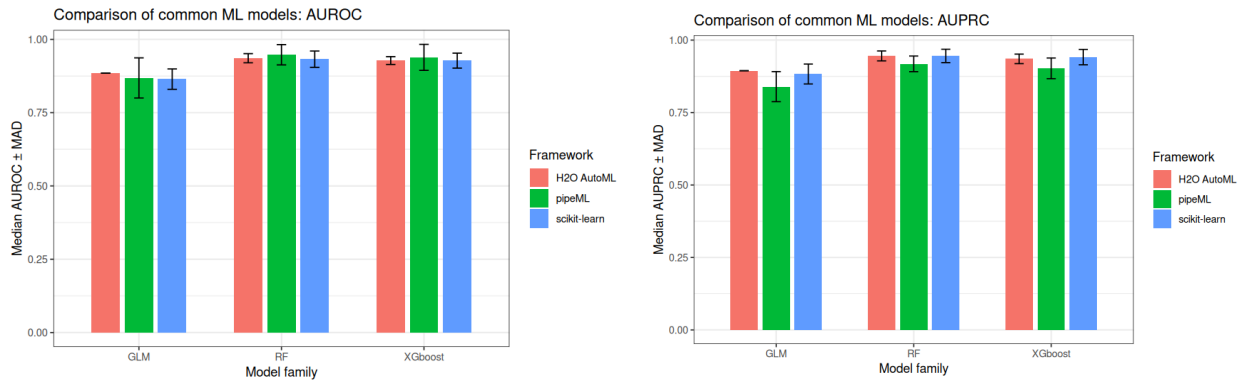


Figure 4. Comparison of predictive performance between established machine learning frameworks and pipeML across three model families: generalized linear models (GLM), random forests (RF), and XGBoost. (A) Area under the receiver operating characteristic curve (AUROC). (B) Area under the precision–recall curve (AUPRC).

"*global dataset features*" inflates predictive performance under cross-validation training settings.

To evaluate the impact of global dataset feature construction on model performance estimation, we designed a controlled experiment using the Sonar benchmark dataset (Blake CL, et al. 1998; Gorman RP, et al. 1998). We generated a set of global dataset features by clustering original input features based on their pairwise correlations across samples and aggregating each cluster into a single feature via within-cluster averaging. This procedure yielded correlation-informed representations that depend on the global structure of the dataset.

We compared two cross-validation (CV) strategies. In the global feature construction CV setting, hereafter referred to as 'standard CV', global dataset features were computed once using the full dataset prior to model training and evaluation. In contrast, in the cross-validation fold-wise feature construction CV setting, hereafter referred to as 'custom CV', feature construction was embedded within the cross-validation loop using *pipeML*, such that correlation structure and feature aggregation were recomputed independently within each training cross-validation fold and then applied to the corresponding test samples, ensuring no leakage of information contributing to feature construction.

Across a diverse set of machine-learning models, standard CV consistently yielded higher median

AUROC values compared to the custom CV strategy. Moreover, naive cross-validation exhibited systematically lower variability across cross-validation folds, as measured by the median absolute deviation (MAD). In contrast, the custom CV approach resulted in reduced and more heterogeneous performance estimates, reflecting the removal of information leakage arising from global dataset features construction (***Figure 5***). These results demonstrate that when new 'aggregated' features are derived from the original features using the global dataset structure, performing feature construction outside the cross-validation loop can lead to optimistically biased performance estimates. Properly including global dataset features construction within cross-validation is therefore important for reporting unbiased and reproducible estimates of predictive performance.

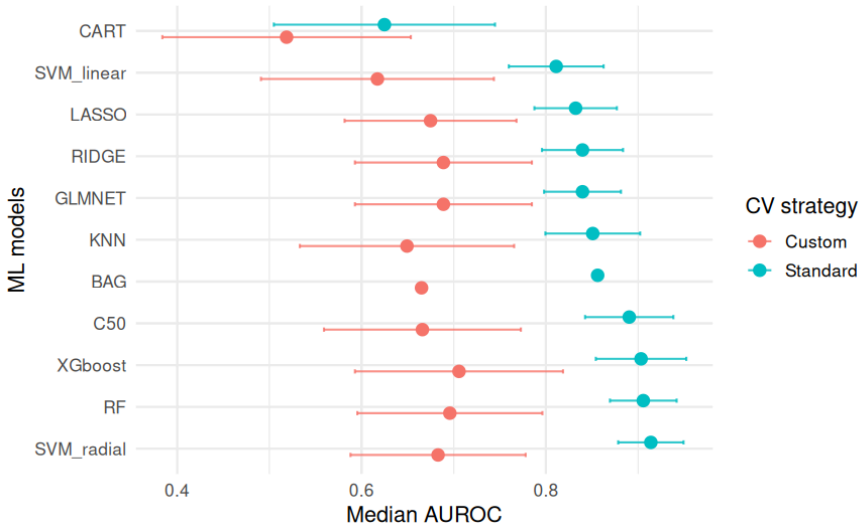


Figure 5. Median AUROC comparing standard and custom cross-validation (CV) strategies across 11 machine learning models implemented in pipeML. Error bars indicate the median absolute deviation (MAD) across CV resamples.

Use case: predicting immunotherapy response using public melanoma cohorts

To illustrate the utility of our pipeline and highlight the caveats of data leakage when computing global dataset features prior to model training in a real-world scenario, we applied our workflow to six independent melanoma cohorts profiled by bulk RNA-sequencing, with gene expression matrices structured as genes × samples and annotated for immunotherapy response (***Supplementary Table 2***).

Using multiple independent cohorts allows us to reduce the risk that model performance is driven by characteristics specific to a single dataset, thereby providing a more reliable assessment of generalization.

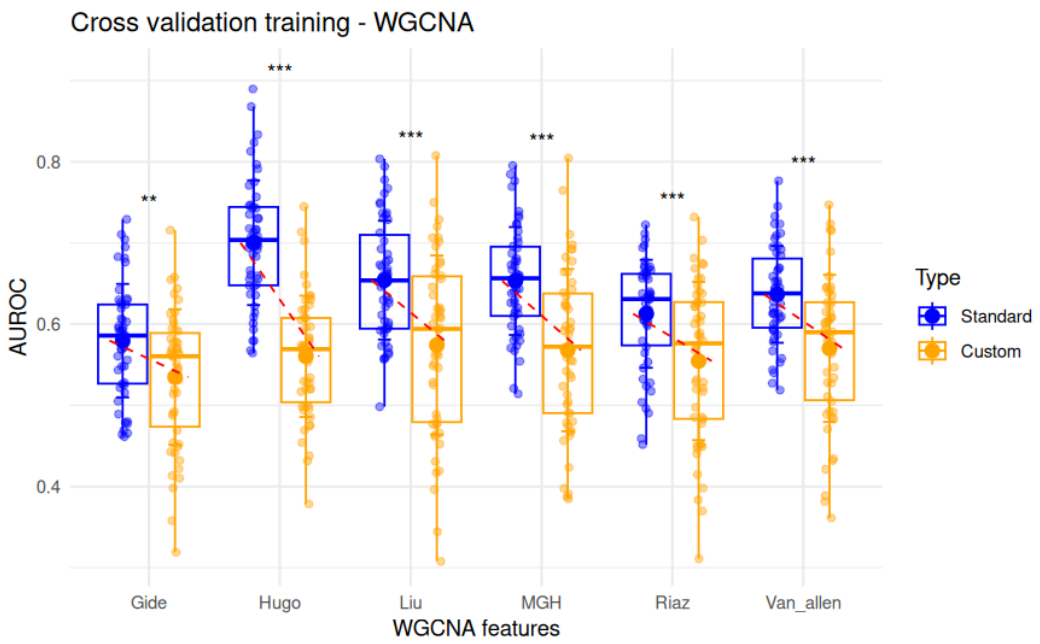
In particular, incorporating several cohorts helps mitigate cohort-specific technical and biological biases and enables evaluation of whether a no-data-leakage pipeline yields more realistic estimates of predictive performance across studies. To further ensure robustness, we adopted a Leave-One-Dataset-Out (LODO) validation strategy. In each iteration, models were trained on a subset of cohorts and evaluated on a held-out dataset that was not used during training. This design ensures that the test set remains fully independent and allows us to assess how well the trained models generalize to unseen cohorts, while minimizing biases arising from dataset-specific effects.

We started by filtering our gene expression matrix to retain the 5000 most variable genes across the full combined cohort. Within each LODO iteration, multiple feature representations were derived, including pathway activity scores using GSVA, gene cluster features obtained via k-medoids clustering, and co-expression module features derived from WGCNA (***see Materials and methods***). We choose these methods as they are usually commonly used in Bioinformatics pipelines, and they tend to leverage data-leakage during ML training.

We compared the two cross-validation strategies (custom and standard CV) that differ in when feature construction is performed relative to cross-validation splitting. In the standard pipeline, features were computed once using the entire set of training cohorts prior to model training. The resulting feature matrices were then provided directly to *pipeML*, and cross-validation was applied to these precomputed features during model training and hyperparameter tuning. For the held-out cohort, features were computed independently using the same transformation procedure, or by projecting the data using parameters learned from the training cohorts (***see Materials and methods; Supplementary Table 3***).

In contrast, the custom CV strategy integrates feature construction within the cross-validation procedure, implemented in *pipeML*. In this setting, gene expression data is provided directly to the training pipeline, and feature extraction is recomputed independently within each training fold during cross-validation. Specifically, GSVA enrichment scores, k-medoids gene clusters, and WGCNA co-expression modules are derived using only the samples belonging to the training portion of each fold. The learned feature transformations (e.g., cluster assignments or module definitions) are then applied to the corresponding validation samples within that fold. This design ensures that feature construction does not use information from validation samples during model training, thereby preventing potential data leakage (***Supplementary Table 4***).

Predictive performance was evaluated on the held-out LODO cohort using the area under the receiver operating characteristic curve (AUROC), enabling comparison between the standard and custom CV strategies in terms of their ability to generalize to independent datasets (***Figure 6; Supplementary Table 5 and 6***).



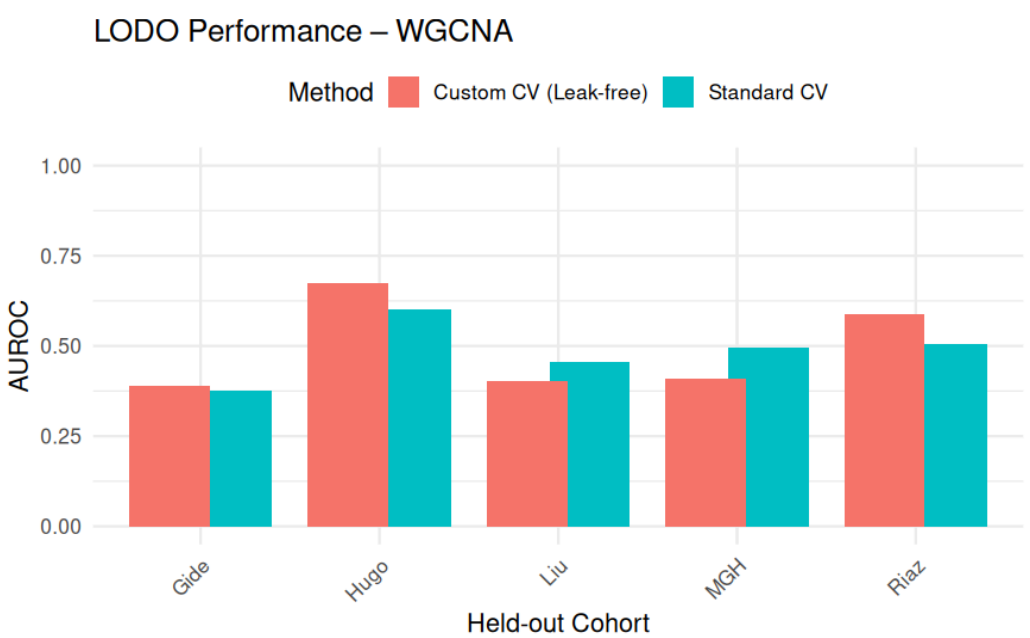


Figure 6. AUROC predictive performance for held-out cohorts comparing standard cross-validation (CV) and custom (leak-free) CV strategies using WGCNA features. (A) Boxplots of AUROC per cohort for each CV strategy across all models; paired differences were assessed using a paired t-test. Statistical significance is indicated as follows: *** p < 0.001, ** p < 0.01, * p < 0.05, ns = not significant. (B) Median AUROC for each cohort using standard and custom CV strategies, highlighting overall performance differences between CV methods.

Prediction performance is highly affected by the choice of hyperparameters during custom functions for cross-validation folds construction.

Custom cross-validation fold functions can involve multiple user-defined parameters. Identifying the optimal combination of these parameters depends largely on the analysis and the predictive metric of interest. To address this, *pipeML* provides a framework to tune these hyperparameters and optimize the custom fold function based on a performance metric such as AUROC, AUPRC and C-index.

Similar to standard hyperparameter tuning for ML models, *pipeML* evaluates multiple parameter combinations, calculates performance across cross-validation resamples, and identifies the combination that produces the best predictive output.

To illustrate this, we made use of the algorithm WGCNA previously explained (***see Materials and methods***), as this algorithm considers many parameters that can be set up by the user including soft-thresholding power, minimum module size, module merging threshold, and module splitting sensitivity. During cross-validation, *pipeML* evaluates the predictive performance of models trained with features derived from each parameter combination, allowing the identification of the configuration that maximizes the chosen metric.

Figure 7 shows the resulting performance (AUROC) across all tested parameter combinations. Each point represents one combination of hyperparameters, connected by a line to show trends.

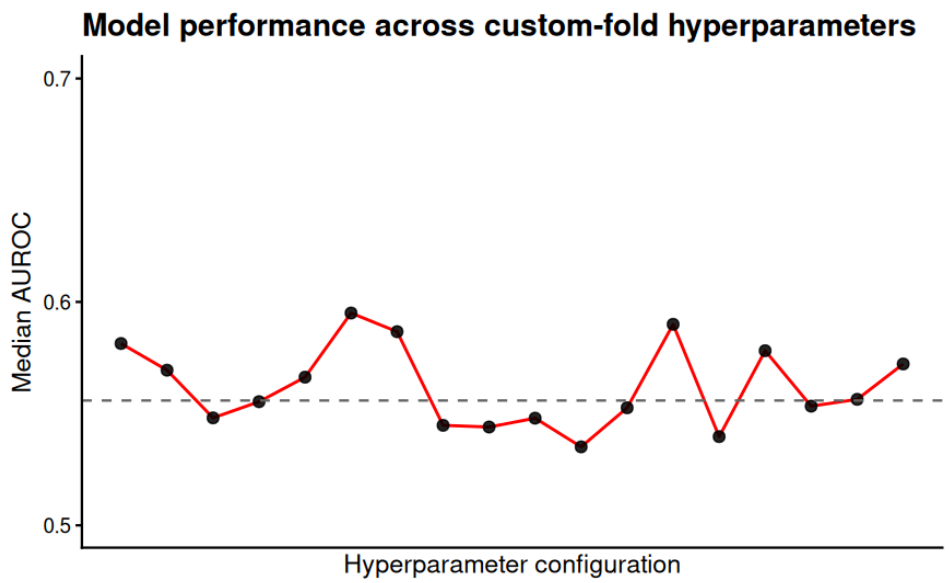


Figure 7. Model performance (AUROC) across different hyperparameter combinations of custom cross-validation (CV) folds. Each point represents the median AUROC for a given fold configuration, highlighting how hyperparameter choices affect predictive performance.

Materials and methods

pipeML implements a machine learning pipeline for classification and survival tasks using extensively the R packages *caret* (v6.0.94), *tidymodels* (v1.4.1), *parsnip* (v1.3.3) and *censored* (v0.3.3).

Machine learning pipeline for binary classification tasks

For classification tasks, we implemented a diverse set of classification algorithms that are benchmarked on the fly, including tree-based methods, regularized linear models, support vector machines with radial and linear kernels, and instance-based learning.

Model performance is assessed using repeated stratified k-fold cross-validation to ensure robust estimation and preserve class distributions across folds. In this framework, the dataset is partitioned into k folds and the cross-validation procedure is repeated multiple times with different fold assignments. Hyperparameter tuning is performed within the cross-validation loop using grid search. Model selection can be optimized according to a user-defined performance metric, including the area under the receiver operating characteristic curve (AUROC), the area under the precision–recall curve (AUPRC), or classification accuracy, depending on the task.

The current implementation supports multiple classification algorithms, including:

- *Bagged classification trees*
- *Random forest*
- *C5.0 decision trees*
- *Regularized logistic regression (elastic net)*
- *k-nearest neighbors*
- *Classification and regression trees (CART)*
- *Lasso regression*
- *Ridge regression*
- *Support vector machines with linear and radial kernels*
- *Extreme gradient boosting (XGBoost)*

Machine learning pipeline for survival tasks

For time-to-event outcomes, we implemented a unified survival modeling framework based on the parsnip and workflows ecosystems, enabling consistent training, tuning, and evaluation across multiple survival model families. Models are specified using a censored regression formulation and fitted using appropriate computational engines for each algorithm.

When applicable, model-specific hyperparameters (e.g., regularization strength, mixing parameters, or number of trees) are tuned via grid search within the training data. All models are trained using cross-validation strategies consistent with the main pipeline, ensuring that feature construction and hyperparameter selection were confined to the training folds.

Model performance is then evaluated on held-out data using the Concordance index (C-index), which measures the ability of the model to correctly rank survival times under right-censoring.

The current implementation supports multiple survival algorithms, including:

- *Cox proportional hazards model*
- *Elastic net–regularized Cox regression*
- *Parametric accelerated failure time (AFT) models*
- *Conditional inference survival trees*
- *Bagged CART survival models*
- *Random survival forests*
- *Gradient boosting for censored outcomes*

Machine learning model selection

The pipeline includes an automated procedure to identify the best-performing model according to a user-defined metric (AUROC or AUPRC). Models are ranked based on the median performance obtained during cross-validation, and the model with the highest score is selected as the top model. This model can subsequently be used as the final predictive model when model stacking is not applied.

Hyperparameter tuning strategy

Hyperparameter tuning was embedded directly within the cross-validation framework to ensure unbiased model selection and prevent information leakage. For each repeated and stratified k-cross-validation fold split, candidate hyperparameter configurations were evaluated exclusively on the training cross-validation folds. When no custom feature cross-validation fold-construction was specified, model-specific tuning grids defined by the caret framework were used, and optimal hyperparameters were selected based on cross-validated performance using the primary optimization metric (Accuracy, AUROC, or AUPRC). When a custom cross-validation fold construction function was provided (e.g., for global dataset features feature generation such as WGCNA in this paper), hyperparameter tuning was extended to jointly evaluate both model-level parameters (e.g., number of trees, regularization strength, kernel parameters) and feature-construction parameters. In this setting, each cross-validation fold contained multiple sub-cross-validation folds corresponding to different feature-construction configurations, and all machine learning models were trained and evaluated independently for each configuration. Performance metrics were aggregated across all resamples, and the optimal combination of model hyperparameters and feature-construction parameters was selected based on the median cross-validated metric. Finally, models were retrained on the full training dataset using the selected optimal parameters.

Machine learning with Leave-One-Dataset-Out (LODO) cross-validation

To evaluate the generalizability of our model across multiple melanoma cohorts, we implemented a Leave-One-Dataset-Out (LODO) cross-validation strategy in the ML pipeline described above. For each iteration, one cohort was held out as an independent test set, while the remaining cohorts were combined to form the training set. To ensure robust model optimization within each training set, we grouped the training data by cohort and generated K stratified cross-validation folds within each group based on the target label, ensuring class balance within and across cross-validation folds. cross-validation folds from each cohort were then merged such that each cross-validation fold retained a representative distribution of each class label from all available training cohorts. This process was repeated 100 times, and the resulting cross-validation folds were used internally by the model for hyperparameter tuning and performance

estimation. The trained model from each LODO iteration was then evaluated on the held-out cohort. Together, this LODO framework enabled us to assess how well models trained on combined cohorts generalize to completely unseen datasets while accounting for both inter-cohort variability and class imbalance.

Feature important analysis

Model interpretability is assessed using SHAP (SHapley Additive exPlanations) (Lundberg & Lee, 2017) values to quantify the contribution of individual features to model predictions. SHAP values are estimated using the R package *fastshap* (v0.1.1) (Greenwell B, 2024), which provides a model-agnostic Monte Carlo approximation of Shapley values. For each trained model, SHAP values are computed separately within each cross-validation resample to account for variability in model fitting across training folds. Within each resample, the model was retrained using the optimal hyperparameters identified during model tuning, and SHAP values were estimated for the corresponding hold-out samples.

The *fastshap::explain* function is then applied using the training data of the corresponding fold as the background dataset and the hold-out samples as the evaluation dataset. The SHAP values therefore represent the contribution of each feature to the predicted probability for each individual sample relative to the model's baseline prediction. SHAP values obtained across all resamples are subsequently combined to produce a single matrix of per-sample feature contributions. Global feature importance is summarized by computing the mean absolute SHAP value across samples, providing a ranking of features according to their overall influence on model predictions.

For visualization and interpretation, users can then use the R package *shapviz* (v0.10.3), which enables graphical summaries such as feature importance plots, SHAP beeswarm plots, and dependence plots. These visualizations combine SHAP values with the corresponding feature values, allowing assessment of both the magnitude and direction of feature effects on model predictions.

Feature selection using repeated Boruta

Feature selection can be performed using the *Boruta* R package (v8.0.0). To improve the stability of feature selection, *pipeML* executes it multiple times using different random seeds. In each iteration, Boruta compares the importance of original variables with that of randomly permuted "shadow" features to determine whether a variable carries information beyond random noise. Features are assigned one of three labels: *Confirmed, Tentative, or Rejected*. To obtain robust feature importance estimates, results

from all iterations are aggregated by computing the median importance of each feature and summarizing the frequency of Boruta decisions across runs. Features are retained if they are classified as Confirmed (or optionally Tentative) in more than a user-defined proportion of iterations.

Model stacking

pipeML optionally implements model stacking to combine predictions from multiple base learners. In this approach, each machine learning algorithm generates probabilistic predictions for the outcome variable. These predictions are then used as input features for a meta-learner, which learns how to optimally combine them. It also enables the calculation of feature importance scores based on the contribution of each base learner to the final stacked model.

Independent test evaluation

After cross-validation and model selection, the best-performing model configuration is refitted on the full training dataset. The trained model can be then applied to an independent test dataset to generate predictions. Model performance can be evaluated using receiver operating characteristic curves (ROC), precision–recall curves (PRC), and additional classification metrics derived from an optimized classification threshold (F1, MCC).

Sensitivity–specificity calculation across thresholds

Sensitivity and specificity values were computed across all possible classification thresholds using predicted probabilities from the trained model. Predictions were first sorted in decreasing order of the probability assigned to the positive class. For each threshold, cumulative counts of true positives (TP) and false positives (FP) were calculated using cumulative sums. Sensitivity (true positive rate) was computed as the proportion of correctly identified positive samples relative to all positive samples, while specificity was calculated as one minus the false positive rate (FPR).

Accuracy estimation

Classification accuracy was calculated as the proportion of correctly classified samples relative to the total number of samples. Accuracy was derived from the confusion matrix values and computed as the sum of true positives and true negatives divided by the total number of observations.

Precision estimation

Precision was calculated as the proportion of true positive predictions among all positive predictions. This metric quantifies how many predicted positive instances correspond to true positives and is particularly informative when evaluating models on imbalanced datasets.

Recall estimation

Recall, also referred to as sensitivity, was calculated as the proportion of true positive predictions among all actual positive samples. Recall measures the ability of the classifier to correctly identify positive instances.

F1 score calculation

The F1 score was calculated as the harmonic mean of precision and recall. This metric provides a balanced measure of model performance when both false positives and false negatives are relevant.

Matthews correlation coefficient

The Matthews correlation coefficient (MCC) was computed as a balanced performance metric that incorporates all four confusion matrix components (TP, TN, FP, FN). MCC ranges from −1 to 1, where 1 represents perfect prediction, 0 indicates random prediction, and −1 represents complete disagreement between predictions and observations.

Receiver operating characteristic curve and AUROC calculation

Receiver operating characteristic (ROC) curves were generated using the sensitivity and false positive rate values calculated across classification thresholds. The area under the ROC curve (AUROC) was computed using the trapezoidal rule, integrating the area between successive points along the ROC curve. This metric summarizes the model's ability to discriminate between positive and negative classes across all possible thresholds.

Precision–recall curve and AUPRC calculation

Precision–recall curves were constructed using precision and recall values computed across classification thresholds. The area under the precision–recall curve (AUPRC) was calculated using the trapezoidal rule.

AUPRC is particularly informative for evaluating models trained on imbalanced datasets where the positive class is underrepresented.

Use case: predicting immunotherapy response using public melanoma cohorts

GSVA pathway features

Pathway-level features were computed using Gene set variation analysis (GSVA) using the *GSVA* R package (v1.50.5). KEGG pathway gene sets were obtained from the *msigdbr* R package (v7.5.1), using the C2:CP:KEGG collection for Homo sapiens. Gene symbols associated with each pathway were grouped to construct gene sets. Gene set enrichment scores were then calculated for each sample using the GSVA algorithm with a Gaussian kernel cumulative distribution function, appropriate for normalized continuous expression data. GSVA transforms the gene expression matrix (genes × samples) into a sample-wise pathway activity matrix, where each value represents the relative enrichment of a pathway within a given sample.

K-medoids gene cluster features

Gene-level clustering features were derived using the k-medoids clustering algorithm implemented in the *cluster* R package (v2.1.4) via the CLARA (Clustering Large Applications) method. Genes were partitioned into k = 50 clusters based on their expression profiles across samples. For each cluster, a representative feature was computed by taking the mean expression value of all genes assigned to that cluster for each sample. This produced a reduced feature matrix where each column corresponds to the average expression of a gene cluster. During training, cluster assignments were estimated from the training data and subsequently reused when computing features for new datasets to ensure consistency between training and validation data.

Weighted Gene Correlation Network Analysis (WGCNA) module features

Co-expression module features were computed using the *WGCNA* R package (v1.73) (Langfelder P & Horvath S, 2008). The gene expression matrix was transposed to obtain a samples × genes format, and a weighted gene co-expression network was constructed using a soft-thresholding power of 6. Modules of co-expressed genes were identified using the *blockwiseModules* function with an unsigned topological overlap matrix (TOM) and a minimum module size of 30 genes. Each module was then summarized using its module eigengene, defined as the first principal component of the expression matrix of genes belonging to that module. This eigengene represents the dominant expression pattern of the module across

samples. As with the clustering-based features, module assignments learned from the training data were reused to compute module eigengenes for external datasets.

Conclusion

pipeML provides a flexible and robust framework for machine learning analyses in biological settings where features are dataset-dependent and traditional cross-validation strategies can introduce information leakage. By enabling custom cross-validation fold construction and fold-aware feature recomputation, *pipeML* ensures that model training and evaluation reflect realistic predictions on unseen data. This approach allows users to integrate complex, biologically informed feature engineering steps, including enrichment-based, correlation-based, or network-derived features, without compromising model validity. Implemented in R, *pipeML* seamlessly integrates with the Bioconductor ecosystem and existing bioinformatics workflows, supporting both classification and survival analyses. As predictive modeling in biomedical research increasingly relies on global dataset feature representations, *pipeML* offers a scalable, modular, and leakage-aware solution that delivers reliable performance estimates while remaining fully adaptable to user-defined feature construction strategies and current machine learning workflows.

Author contributions

M.H. conceived the project. M.H. developed the package and methods. M.H performed the benchmark and the analyses. V.P. and M.H wrote and reviewed the manuscript. V.P. supervised the project.

Acknowledgements

The authors thank Raphaël Mourad and Hafida Hamdache for their helpful feedback on the package methods. We also thank Andrei Zinovyev for his valuable critical reading of the previous version of the manuscript and providing insightful comments.

Funding

Work in the Pancaldi lab was funded by the Chair of Bioinformatics in Oncology of the CRCT (INSERM; Fondation Toulouse Cancer Santé and Pierre Fabre Research Institute) and Ligue Nationale Contre le Cancer. This study has been partially supported through the grant EUR CARe N°ANR-18-EURE-0003 in the framework of the Programme des Investissements d'Avenir and an Eiffel Excellence doctoral fellowship to M. H.

Data availability

No new experimental data was generated as part of this study. All the datasets used are publicly available as described in ***Supplementary Table 2***.

Key points

- *pipeML* provides an end-to-end, leakage-aware machine learning framework for classification and survival tasks that integrates feature selection, model training, validation, prediction, and interpretation within a single reproducible pipeline.
- *pipeML* provides custom fold-aware feature construction where global dataset features are recomputed independently within each cross-validation fold, preventing information leakage and overoptimistic performance estimates.
- *pipeML* supports robust model evaluation across heterogeneous datasets through repeated stratified cross-validation, model stacking, and cohort-aware strategies such as leave-one-dataset-out (LODO) analysis.
- *pipeML* facilitates transparent model interpretation and reporting by providing built-in tools for performance visualization, variable importance, and SHAP-based explanations, making it well suited for translational and systems biology applications.

References

Lapuente-Santana Ó, van Genderen M, Hilbers PAJ, Finotello F, Eduati F. Interpretable systems biomarkers predict response to immune-checkpoint inhibitors. Patterns (N Y). 2021 Jun 30;2(8):100293. doi: 10.1016/j.patter.2021.100293. PMID: 34430923; PMCID: PMC8369166.

Lapuente-Santana, Ó., Sturm, G., Kant, J., Ausserhofer, M., Zackl, C., Zopoglou, M., … Finotello, F. (2024). Multimodal analysis unveils tumor microenvironment heterogeneity linked to immune activity and evasion. IScience, 27(8), 110529. https://doi.org/10.1016/j.isci.2024.110529

Wolberg WH, Mangasarian OL (1990). “Multisurface Method of Pattern Separation for Medical Diagnosis Applied to Breast Cytology.” Proceedings of the National Academy of Sciences, 87(23),

9193–9196. doi:10.1073/pnas.87.23.9193.

Zhang J (1992). “Selecting Typical Instances in Instance-Based Learning.” In Sleeman D, Edwards P (eds.), Machine Learning Proceedings 1992, 470–479. Morgan Kaufmann, San Francisco (CA). ISBN 978-1-55860-247-2. doi:10.1016/B9781558602472.500668.

Blake CL, Merz CJ (1998). “UCI Repository of Machine Learning Databases.” University of California, Irvine, Department of Information and Computer Science. Formerly available from ‘http://www.ics.uci.edu/~mlearn/MLRepository.html’.

Loprinzi CL. Laurie JA. Wieand HS. Krook JE. Novotny PJ. Kugler JW. Bartel J. Law M. Bateman M. Klatt NE. et al. Prospective evaluation of prognostic variables from patient-completed questionnaires. North Central Cancer Treatment Group. Journal of Clinical Oncology. 12(3):601-7, 1994.

Gorman RP, Sejnowski TJ (1988). “Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets.” Neural Networks, 1(1), 75–89. ISSN 0893-6080. doi:10.1016/0893-6080(88)900238.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17, 4768–4777. https://dl.acm.org/doi/10.5555/3295222.3295230

Greenwell B (2024). fastshap: Fast Approximate Shapley Values. R package version 0.1.1

Langfelder P, Horvath S. WGCNA: an R package for weighted correlation network analysis. BMC Bioinformatics. 2008 Dec 29;9:559. doi: 10.1186/1471-2105-9-559. PMID: 19114008; PMCID: PMC2631488.

Van Allen E.M., Miao D., Schilling B., Shukla S.A., Blank C., Zimmer L., Sucker A., Hillen U., Foppen M.H.G., Goldinger S.M., et al. Genomic correlates of response to CTLA-4 blockade in metastatic melanoma. Science. 2015;350:207–211. doi: 10.1126/science.aad0095.

Riaz N., Havel J.J., Makarov V., Desrichard A., Urba W.J., Sims J.S., Hodi F.S., Martín-Algarra S., Mandal R., Sharfman W.H., et al. Tumor and Microenvironment Evolution during Immunotherapy with Nivolumab. Cell. 2017;171:934–949. doi: 10.1016/j.cell.2017.09.028.

Hugo W., Zaretsky J.M., Sun L., Song C., Moreno B.H., Hu-Lieskovan S., Berent-Maoz B., Pang J., Chmielowski B., Cherry G., et al. Genomic and Transcriptomic Features of Response to Anti-PD-1 Therapy in Metastatic Melanoma. Cell. 2016;165:35–44. doi: 10.1016/j.cell.2016.02.065.

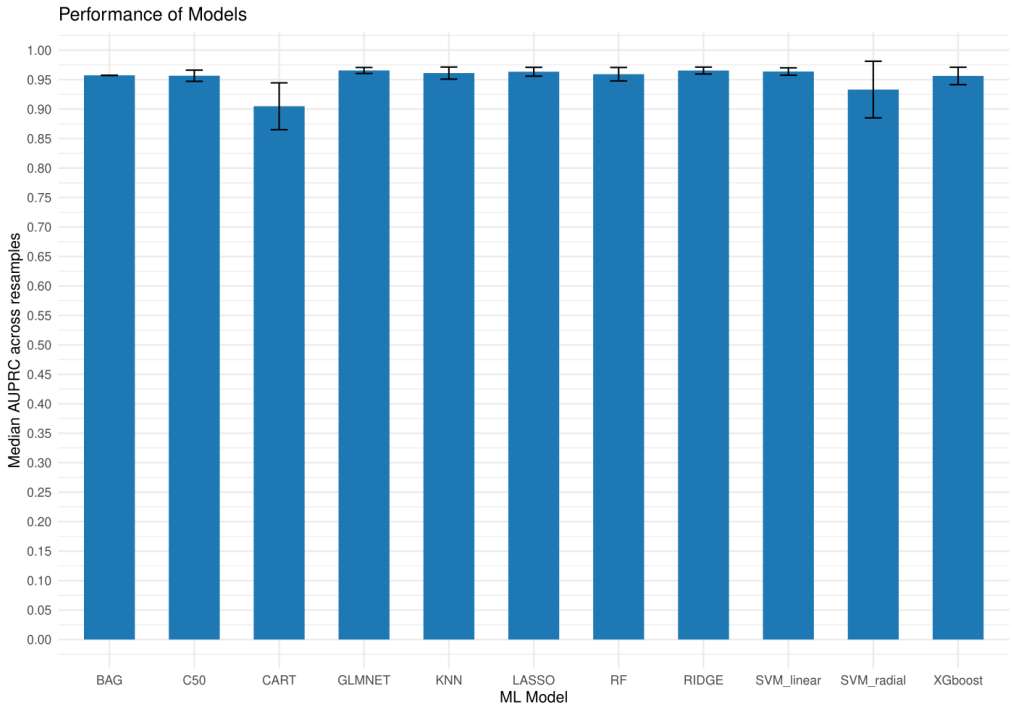
Gide T.N., Quek C., Menzies A.M., Tasker A.T., Shang P., Holst J., Madore J., Lim S.Y., Velickovic R., Wongchenko M., et al. Distinct Immune Cell Populations Define Response to Anti-PD-1 Monotherapy and Anti-PD-1/Anti-CTLA-4 Combined Therapy. Cancer Cell. 2019;35:238–255. doi: 10.1016/j.ccell.2019.01.003.

Liu D., Schilling B., Liu D., Sucker A., Livingstone E., Jerby-Arnon L., Zimmer L., Gutzmer R., Satzger I., Loquai C., et al. Integrative molecular and clinical modeling of clinical outcomes to PD1 blockade in patients with metastatic melanoma. Nat. Med. 2019;25:1916–1927. doi: 10.1038/s41591-019-0654-5.

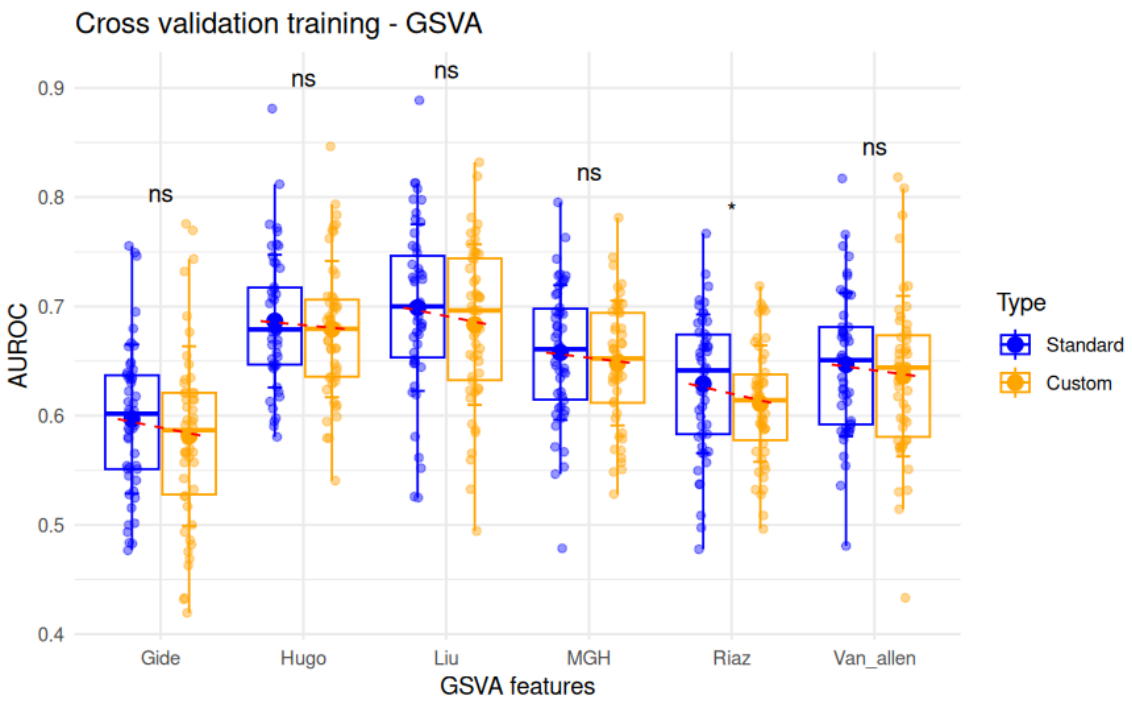
Freeman SS, Sade-Feldman M, Kim J, Stewart C, Gonye ALK, Ravi A, Arniella MB, Gushterova I, LaSalle TJ, et al. Combined tumor and immune signals from genomes or transcriptomes predict outcomes of checkpoint inhibition in melanoma. Cell Rep Med. 2022 Feb 15;3(2):100500. doi: 10.1016/j.xcrm.2021.100500. PMID: 35243413; PMCID: PMC8861826.

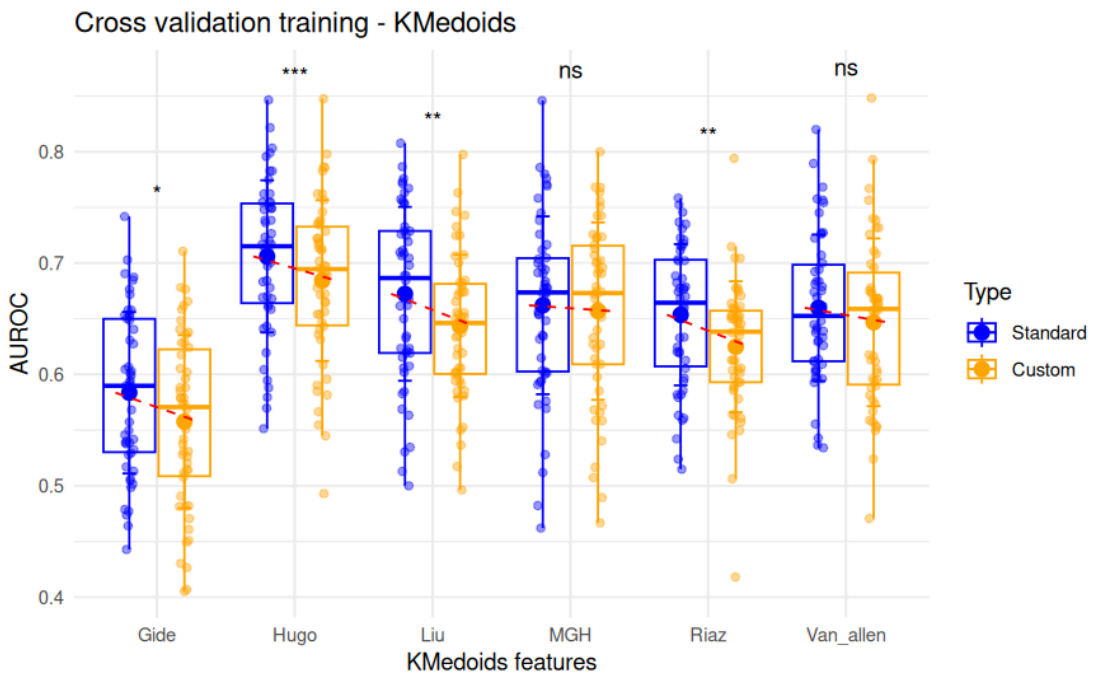
Supplementary Materials

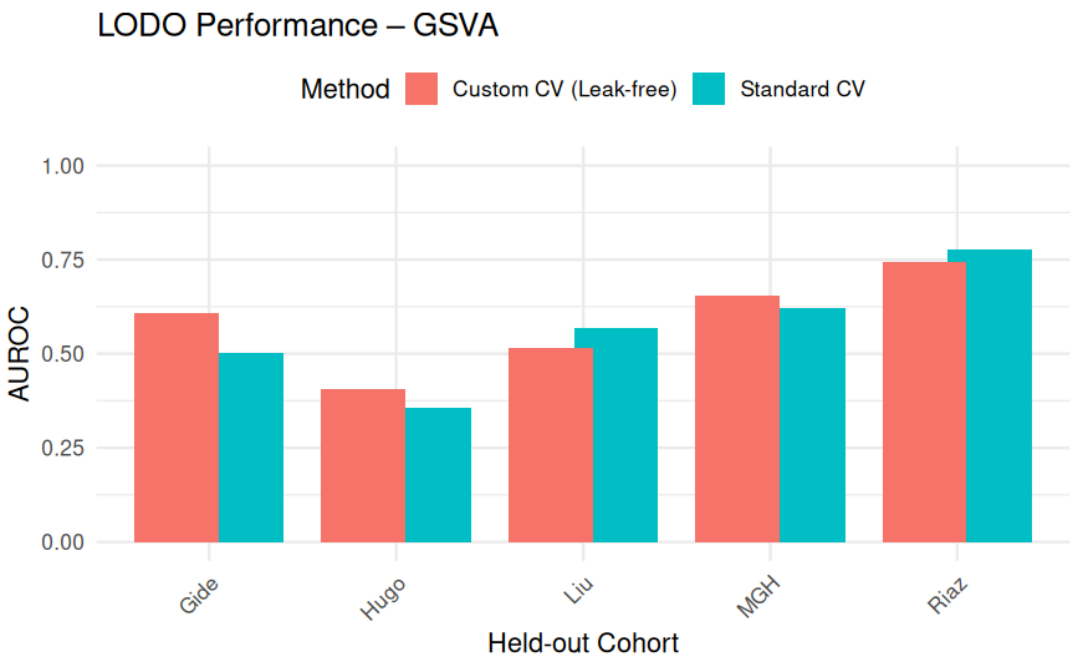
A new pipeline for cross-validation fold-aware machine learning prediction of clinical outcomes addresses hidden data-leakage in omics based 'predictors'.

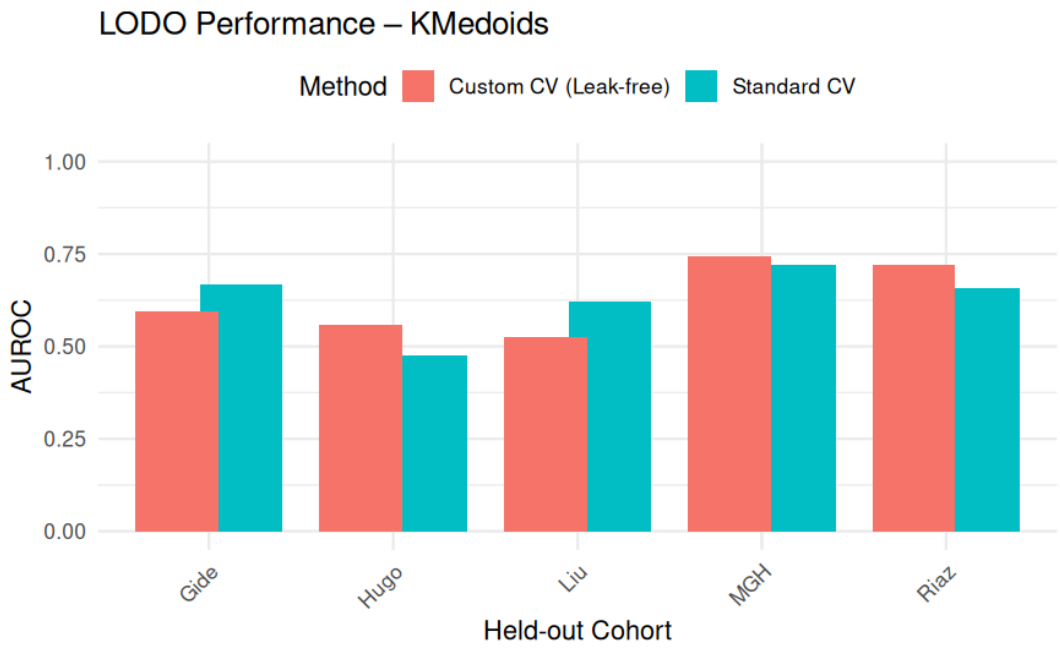


Supplementary Figure 1. Performance and interpretation of classification models trained on the Breast Cancer Wisconsin dataset using a 70% training and 30% test split. Cross-validated AUPRC values for different classification algorithms evaluated on the training set.


Cross validation training - GSVA
AUROC
0.9
0.8
0.7
0.6
0.5
0.4
ns
ns
ns
ns
*
ns
Gide
Hugo
Liu
MGH
Riaz
Van_allen
GSVA features
Type
Standard
Custom


Cross validation training - KMedoids
AUROC
0.8
0.7
0.6
0.5
0.4
*
***
**
ns
**
ns
Gide
Hugo
Liu
MGH
Riaz
Van_allen
KMedoids features
Type
Standard
Custom





Supplementary Figure 2. AUROC predictive performance for held-out cohorts comparing standard cross-validation (CV) and custom (leak-free) CV strategies. (A–B) Boxplots of AUROC per cohort for each CV strategy across all models using GSVA (A) and KMedoids (B) features. Paired differences were assessed using a paired t-test. Statistical significance is indicated as follows: *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$, ns = not significant. (C–D) Median AUROC for each cohort using standard and

custom CV strategies for GSVA (C) and KMedoids (D) features, highlighting overall performance differences between CV methods.

Supplementary Tables are provided as .xlsx files

Supplementary Table 1: Training and prediction metrics obtained for example datasets (Result 1) including AUROC, AUPRC, hyperparameters, Sensitivity, Specificity, fpr, Accuracy, Precision, Recall, F1, MCC scores.

Supplementary Table 2: Melanoma cohorts information used in use-case results.

Supplementary Table 3: Training metrics obtained for use case example per held-out cohort and feature set (GSVA, KMedoids and WGCNA) including AUROC, AUPRC, hyperparameters, Sensitivity, Specificity, fpr, Accuracy, Precision, Recall, F1, MCC scores for standard k-fold strategy.

Supplementary Table 4: Training metrics obtained for use case example per held-out cohort and feature set (GSVA, KMedoids and WGCNA) including AUROC, AUPRC, hyperparameters, Sensitivity, Specificity, fpr, Accuracy, Precision, Recall, F1, MCC scores for custom k-fold strategy.

Supplementary Table 5: Prediction metrics obtained for use case example per held-out cohort and feature set (GSVA, KMedoids and WGCNA) including AUROC, AUPRC, hyperparameters, Sensitivity, Specificity, fpr, Accuracy, Precision, Recall, F1, MCC scores for standard k-fold strategy.

Supplementary Table 6: Prediction metrics obtained for use case example per held-out cohort and feature set (GSVA, KMedoids and WGCNA) including AUROC, AUPRC, hyperparameters, Sensitivity, Specificity, fpr, Accuracy, Precision, Recall, F1, MCC scores for custom k-fold strategy.