All Models Are Wrong, but Some Are Interchangeably Right - Supplementary Materials

Supplementary materials and code are open-source and open-data on our online repository.

https://github.com/sub5716ijcai2026/RecoSHAP

Appendix

A Machine Learning Models

We tested the following 46 machine learning surrogate models and optimized their hyperparameters with a 4-folds cross-validation loop. Every model random state is set to the same seed for reproducibility and fair comparison.

- **Support Vector Machine (SVM).** This method is based on the ε-insensitive loss function, which seeks a function $f(x)$ that is as "flat" as possible while ensuring deviations from observed values remain within an ε-tube. The regularization parameter C balances model complexity and tolerance for deviations larger than ε, while the choice of kernel, either `linear` or `rbf`, determines whether the regression is carried out in the original feature space or in a higher-dimensional embedding via the kernel trick. In our experimental protocol, we tune $C \in \{0.1, 1\}$ and kernel type from the set {`linear`,`rbf`}. Note that the 2 models `SVC` and `SVR` have been tested for classification and regression, respectively.
- **Histogram-based Gradient Boosting.** An additive tree-ensemble algorithm that builds regression trees using histogram-based binning for efficiency. Hyperparameters include the maximum number of trees,`max_iter` $\in \{100, 200\}$, tree wideness controled via `max_leaf_nodes` $\in \{31, 63\}$, and shrinkage as a `learning_rate` $\in \{0.05, 0.1\}$. It is well-suited for large datasets and heterogeneous features. It was tested on Regression tasks.
- **Passive-Aggressive model.** The passive-aggressive algorithms are a family of algorithms for large-scale learning. They are similar to the Perceptron in that they do not require a learning rate. However, contrary to the Perceptron, they include a regularization parameter $C \in \{0.1, 1\}$. It handles streaming or large-scale data with fast, sparse updates. It was tested on Regression tasks.
- **Huber loss-based model.** This estimator minimises a hybrid loss that uses squared error for observations with small standardized residuals and absolute error for observations with large standardized residuals. The model jointly estimates the regression coefficients, the intercept, and a scale parameter that makes the robustness threshold invariant to global rescaling of the response; this scale parameter does not, however, correct for differing scales across input features. The Huber loss, therefore reduces sensitivity to outliers while still allowing extreme observations to influence the fit. In our tuning protocol we varied the key hyperparameters as follows: the threshold parameter `epsilon` was chosen from $\{1.35,\ 1.50\}$ where smaller values yield greater robustness to outliers and the regularization strength `alpha` from $\{10^{-4},\ 10^{-3}\}$, that is the L2-penalty on the weight vector. It was tested on Regression tasks.
- **ElasticNet.** This estimator is a linear regression model that combines L1 (lasso) and L2 (ridge) penalties. Here, the hyper-parameter $\alpha \in \{10^{-4},\ 10^{-3},\ 10^{-2}\}$ controls the overall regularization strength, and $\ell_1\_ratio \in \{0.15,\ 0.50,\ 0.85\}$ controls the trade-off between the L1 and L2 components. This formulation is particularly useful when there are many correlated features: the L1 part encourages sparsity, while the L2 part stabilises the selection of groups of correlated variables. It was tested on Regression tasks.
- **Poisson distribution-based model.** This estimator implements a Generalized Linear Model for non-negative count targets using a log link and a Poisson likelihood. It is appropriate when the variance of the response is approximately proportional to its mean (equidispersion). The L2 regularization strength is controlled by the hyper-parameter $\alpha \in \{0.0,\ 0.1\}$; note that $\alpha = 0$ corresponds to the unpenalised GLM and requires the design matrix to have full column rank, which can be numerically unstable in the presence of collinear predictors. The numerical solver is limited by the maximum number of iterations, which we vary as `max_iter` $\in \{100,\ 300\}$. In practice, non-negligible regularization ($\alpha > 0$) helps to stabilise estimates and mitigate overfitting. It was tested on Regression tasks.
- **Tweedie distribution-based model.** This estimator implements a generalized linear model with a Tweedie response distribution. In our experiments, we set the power parameter to $p = 0$, corresponding to a Normal distribution. regularization is controlled by the L2 penalty $\alpha \in \{0.0,\ 0.1\}$ ($\alpha = 0$ yields the unpenalised GLM but requires the design matrix to have full column rank), and we vary the solver iteration limit as `max_iter` $\in \{100,\ 300\}$. The Tweedie model is appropriate when the mean–variance relationship follows the Tweedie family; in cases of extreme overdispersion, zero-inflation, or other departures from the Tweedie assumptions, alternative models (e.g., negative-binomial or zero-inflated formulations) should be considered. It was tested on Regression tasks.
- **Ridge-based model.** This estimator performs linear regression augmented with an L2 penalty on the coefficients, which shrinks their magnitude to mitigate overfitting and instability, especially in the presence of multicollinearity or high dimensionality. The Tikhonov regularization strength is governed by the hyperparameter $\alpha \in \{0.1,\ 1.0,\ 10.0\}$; smaller values of α approximate ordinary least squares, while larger values reduce variance and improve numerical conditioning

at the cost of increased bias. Note that the 2 models `RidgeClassifier` and `RidgeRegressor` have been tested for classification and regression, respectively.

- **SGD-based model.** This estimator fits a linear model via stochastic gradient descent (SGD) by minimising a regularised empirical loss in an online fashion. Two models `SGDClassifier` and `SGDRegressor` have been tested for classification and regression, respectively. It supports multiple loss functions for classification, `loss` $\in$ {`hinge`, `log_loss`}, and for regression, `loss` $\in$ {`squared_error`, `huber`, `epsilon_insensitive`}, thus providing flexibility in handling large-scale data, outliers, and various target distributions. The regularization strength is controlled by $\alpha \in \{10^{-4}, 10^{-3}\}$ which scales the penalty term added to the loss; in our experiments. Because SGD updates parameters incrementally and uses a decreasing learning-rate schedule by default, the method scales reliably to large sample sizes but requires careful tuning of learning dynamics, feature scaling, and convergence criteria.
- **Linear SVM.** A large-scale linear support vector machine solver optimized for high-dimensional data. It first the hyperparameter $C \in \{0.1, 1, 10\}$ controls the trade-off between the regularization penalty and the tolerance to deviations beyond the ε-insensitive tube: larger values of C reduce regularization. Note that the 2 models `LinearSVC` and `LinearSVR` have been tested for classification and regression, respectively.
- **Decision Tree.** A single decision tree that partitions the feature space to capture nonlinearities and interactions; it is highly interpretable but can overfit if unconstrained. We tune `max_depth` $\in \{3, 5, \text{None}\}$ to control model complexity, where deeper trees can fit more detail at the risk of overfitting. Note that the 2 models `DecisionTreeClassifier` and `DecisionTreeRegressor` have been tested for classification and regression, respectively.
- **Random Forest.** This model constructs an ensemble of decision trees, each trained on bootstrap-resampled subsets of the data and random subsets of features. The predictions from all trees are then averaged to reduce variance and improve generalisation. Because each tree is slightly different, the ensemble mitigates the over-fitting inherent in a single tree. Key hyper-parameters include the number of trees `n_estimators` $\in \{100, 200\}$ and the maximum depth of individual trees `max_depth` $\in \{\text{None}, 10\}$. Note that the 2 models `RandomForestClassifier` and `RandomForestRegressor` have been tested for classification and regression, respectively.
- **ExtraTrees.** An ensemble of highly randomized decision trees similar to Random Forests, but with additional randomness in how split thresholds are chosen. This often reduces variance and can speed up training. We tune `n_estimators` $\in \{100, 200\}$ and `max_depth` $\in \{\text{None}, 10\}$ to balance bias, variance, and computational cost. Note that the 2 models `ExtraTreesClassifier` and `ExtraTreesRegressor` have been tested for classification and regression, respectively.
- **Gradient Boosting.** This model sequentially builds an ensemble of shallow regression trees, where each new tree fits to the residual errors (negative gradients) of the previous ensemble. Such stage-wise boosting allows for the flexible optimization of differentiable loss functions and often delivers excellent predictive accuracy. Key hyperparameters in our experiments are the number of boosting stages `n_estimators` $\in \{100, 200\}$ and the learning rate `learning_rate` $\in \{0.05, 0.10\}$. Note that the 2 models `GradientBoostingClassifier` and `GradientBoostingRegressor` have been tested for classification and regression, respectively.
- **AdaBoost.** This meta-ensemble method sequentially fits a series of weak regressors, where each successive model emphasises observations that previous ones poorly predicted. Key tunable hyperparameters in our experiments include the number of boosting rounds `n_estimators` $\in \{50, 100\}$. While this method can yield strong performance, it is known to be sensitive to noisy data and outliers because mis-predicted instances receive increasing emphasis. Note that the 2 models `AdaBoostClassifier` and `AdaBoostRegressor` have been tested for classification and regression, respectively.
- **XGBoost.** This method uses gradient-boosted trees consisting of efficient, regularised tree boosting with advanced training heuristics (e.g., shrinkage, column subsampling, and built-in handling of missing values). We only tune the number of boosting rounds `n_estimators` $\in \{100, 200\}$. Note that the 2 models `XGBoostClassifier` and `XGBoostRegressor` have been tested for classification and regression, respectively.
- **LightGBM.** A fast, histogram-based gradient boosting library optimized for large datasets. LightGBM supports leaf-wise tree growth (with depth control) and efficient handling of large feature sets. In our experiments we tune the number of boosting rounds `n_estimators` $\in \{100, 200\}$ and tree depth `max_depth` $\in \{10, \text{auto}\}$. Note that the 2 models `LightGBMClassifier` and `LightGBMRegressor` have been tested for classification and regression, respectively.
- **CatBoost.** CatBoost implements gradient-boosted decision trees with native support for categorical features and a novel "ordered boosting" algorithm to mitigate prediction shift and target leakage. We tuned both the number of boosting rounds `n_estimators` $\in \{200, 500\}$ and tree `depth` $\in \{4, 6\}$. Note that the 2 models `CatBoostClassifier` and `CatBoostRegressor` have been tested for classification and regression, respectively.
- **k-Nearest Neighbors (kNN).** A non-parametric, instance-based regressor that predicts by averaging the responses of the `k` $\in \{3, 5, 7\}$, nearest training samples in feature space. kNN is simple and locally interpretable but sensitive to feature scaling and the curse of dimensionality; its performance typically degrades as dimensionality increases or irrelevant

features are present. Note that the 2 models `KNNClassifier` and `KNNRegressor` have been tested for classification and regression, respectively.

- **Bayesian Ridge.** A Bayesian linear regression model that places conjugate priors on weight precision and noise precision, yielding closed-form posterior means. The prior strengths are controlled by four hyperparameters: α_1, α_2 (shape and rate for the weight-precision prior) and λ_1, λ_2 (shape and rate for the noise-precision prior); these govern the degree of shrinkage and the model's tolerance to noise. In our experiment,s we explore $\alpha_1, \alpha_2, \lambda_1, \lambda_2 \in \{10^{-6}, 10^{-5}\}$, which corresponds to weakly informative priors that impose mild regularization while allowing the data to dominate the posterior. It was tested on Regression tasks.
- **Gaussian Naive Bayes (GaussianNB).** A probabilistic classifier that assumes features are conditionally independent given the class and that continuous features follow a Gaussian distribution. It is extremely fast and performs well when the independence assumption is approximately satisfied. No hyperparameters were tuned for GaussianNB in our protocol.
- **Logistic Regression.** A linear discriminative classifier that models class probabilities using the logistic (sigmoid) function. We used the implementation with a maximum iteration limit of $\texttt{max\_iter} = 500$ and tuned the inverse regularization strength $C \in \{0.1, 1, 10\}$. It was tested on Classification tasks.
- **Multi-Layer Peceptron.** A feed-forward neural network with one or more hidden layers that flexibly captures non-linearities and feature interactions. We tested either one or two layers in the perception architecture, specified by $\texttt{hidden\_layer\_sizes} \in (\{50,), (100,), (50, 50)\})$, activation functions by $\texttt{activation} \in \{\texttt{ReLU}, \texttt{tanh}\}$, and weight decay $\texttt{alpha} \in \{10^{-4}, 10^{-3}\}$. Note that the 2 models `MLPClassifier` and `MLPRegressor` have been tested for classification and regression, respectively. It was tested on Classification tasks.
- **Kriging with Partial Least Squares.** This surrogate model implements a Gaussian-process (Kriging) framework enhanced by partial least squares (PLS) dimension reduction to improve efficiency and performance in high-dimensional input spaces. We tune the kernel correlation function $\texttt{corr} \in \{\texttt{abs\_exp}, \texttt{pow\_exp}\}$, the mean trend $\texttt{poly} \in \{\texttt{constant}, \texttt{linear}\}$, and the PLS projection dimension $\texttt{kpls\_dim} \in \{2, 3\}$. KPLS is particularly well-suited to moderate-sized problems with many input variables: the PLS projection reduces the effective input dimensionality, which stabilises hyperparameter estimation, lowers computational cost, and improves predictive performance when inputs are highly correlated. It was tested on Regression tasks.
- **Linear Regression.** The standard ordinary least squares regression model fits a linear relationship between inputs and outputs. Its simplicity and interpretability make it a useful baseline for comparing more complex models or non-linear interpolators. It was tested on Regression tasks.
- **Radial Basis Function.** A kernel-based interpolator that combines radial basis functions with optional polynomial trends. Key hyperparameters include the polynomial ($\texttt{poly} \in \{\texttt{without}, \texttt{constant}, \texttt{linear}\}$) and the regularization strength ($\texttt{reg} \in \{10^{-10}, 10^{-9}, 10^{-8}, 10^{-7}\}$). This method is particularly suitable for smooth function approximation and sparse data interpolation. It was tested on Regression tasks.
- **Quadratic Polynomial Surrogate.** A second-order polynomial model that approximates the response as a quadratic function of the inputs. It is particularly suitable for structured regression or interpolation problems with quadratic objectives or constraints. Model coefficients are typically estimated via ordinary least squares, with an additive error term capturing residual variability. It was tested on Regression tasks.
- **Inverse Distance Weighting** An interpolating method where unknown points are estimated as a weighted average of known sample points, with weights inversely proportional to the distance raised to a power $p \in \{1, 2, 3\}$ that controls the influence of nearby points, with larger values giving more weight to closer observations. IDW is simple, locally interpretable, and effective for smoothly varying spatial data, but can perform poorly when data are highly non-stationary or irregularly distributed.It was tested on Regression tasks.
- **Sparse Gaussian Process.** A sparse Gaussian Process regression model that approximates the full covariance using a subset of inducing points, reducing both computational and memory costs compared to standard GPs. The hyperparameter $\texttt{n\_inducing} \in \{20, 30, 50\}$ controls the number of inducing points, balancing accuracy and efficiency. This approach retains principled uncertainty estimates while scaling to larger datasets. It was tested on Regression tasks.
- **CIEL.** This method implements a context-aware ensemble framework in which multiple simple agent learners cooperate under a multi-agent system architecture. The hyperparameters include the expansion learning rate $\alpha \in \{0.1, 0.2\}$, L1 regularization strength $l_1 \in \{0.1, 0.2\}$, memory length of the context agents $\texttt{memory\_length} \in \{5, 10, 20\}$ to retain recent context, and the local agent radius $R \in \{0.3, 0.5, 0.8\}$. It was tested on Regression tasks.
- **Sparse Polynomial Chaos Expansion.** A surrogate model that represents the response as a finite expansion of orthogonal polynomials in the input variables. It is particularly suited for smooth functions and uncertainty-quantification settings, since the basis polynomials are chosen to match input distributions and allow direct access to output moments and sensitivities. We tune the maximum polynomial degree ($\texttt{degree} \in \{2, 3, 10\}$) and a sparsity-truncation threshold

(q_enum $\in \{1.0, 0.99\}$), enabling a compact set of basis terms and improved efficiency for moderate dimensionality problems. It was tested on Regression tasks.

- **Regularized Minimal-energy Tensor-product Splines.** A spline-based surrogate for low-to-moderate dimensional problems that computes coefficients by minimizing a regularized energy functional while fitting training data. Tensor-product structure enables very fast predictions that do not scale with the number of training points. We tune the spline approximation order approx_order $\in \{3, 4, 5\}$.

- **Tabular Prior-data Fitted Network.** TabPFN is a transformer-based foundation model for small to medium-sized tabular datasets. It is pre-trained on large synthetic datasets, enabling predictions without task-specific training or hyperparameter tuning. In our experiments, we vary the ensemble size n_estimators $\in \{8, 16, 24\}$ and aggregate the predictions from multiple forward passes. Each forward pass receives slightly perturbed input data, effectively forming an ensemble of n_estimators "prompts" to improve robustness and accuracy. It was tested on Regression tasks.

B Datasets description

Table 1: Description of the classification datasets used in the experiments, ranked by number of instances.

Dataset	**Number of**		
	Inst. (m)	**Dim.** (d)	**Classes** ($\lvert\mathcal{Y}\rvert$)
iris	150	5	3
wine	178	14	3
breast-cancer	286	10	2
diabetes	768	9	2
vehicle	846	19	4
Schelling	1000	5	2
cmc	1473	10	3
car	1728	7	4
hypothyroid	3163	26	2
chess	3196	37	2
splice	3188	61	3
churn	5000	21	2
Loan_Modelling	5000	13	2
mushroom	8124	23	2
Adult	48842	15	2
shuttle	58000	10	5

Table 2: Description of the regression datasets used in the experiments.

Dataset	**Inst.** (m)	**Dim.** (d)	**Problem size**
analcatdata_apnea1	475	3	small
ERA	44	4	small
LEV	92	4	small
ESL	199	4	small
pm10	500	7	medium
pollen	3 848	4	medium
Wine_Quality	1 018	11	medium
Abalone	4 177	8	big
puma8NH	8 192	8	big
cpu_small	8 192	12	big
wind	6 574	14	big
satellite_image	6 435	36	big
pol	14 958	48	very big
houses	20 640	8	very big
BNG_lowbwt	31 104	9	very big
Schelling	1000	5	-

C Extra results and figures

C.1 Schelling problem

For the Thomas Schelling's model of segregation, we predicted both the final sparsity, a regression problem and if the simulation will converge towards an equilibrium, a classification problem. We performed a global sensitivity analysis to evaluate how input variables affect model outputs. Random Forest feature importance based on MDI highlights intolerance and density as the most influential variables, while grid size has a weaker effect (Fig. 2). SHAP-based importance from k-Nearest Neighbors shows different patterns due to its local, instance-specific nature (Fig. 3). To assess the consistency of explanations across models, we computed pairwise NDCG on SHAP-value rankings. This analysis reveals clusters of models with similar importance patterns, distinguishing local from global surrogates (Fig. 1). Hierarchical clustering further confirms two main groups of models, reflecting differences in generalization versus sensitivity to local effects (Fig. 4). Note that we tested a new ML model, GENN (Gradient-Enhanced Neural Network).

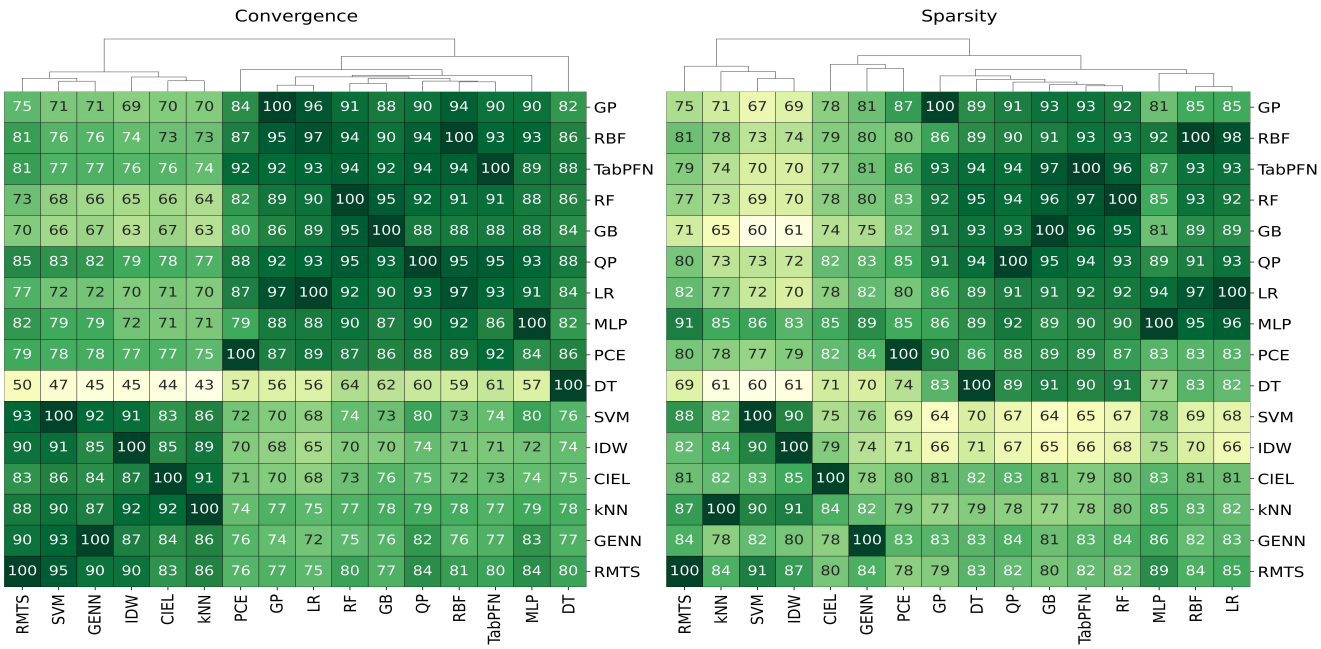


Figure 1: SHAP values NDCG (%) for every pairs of clustered models.

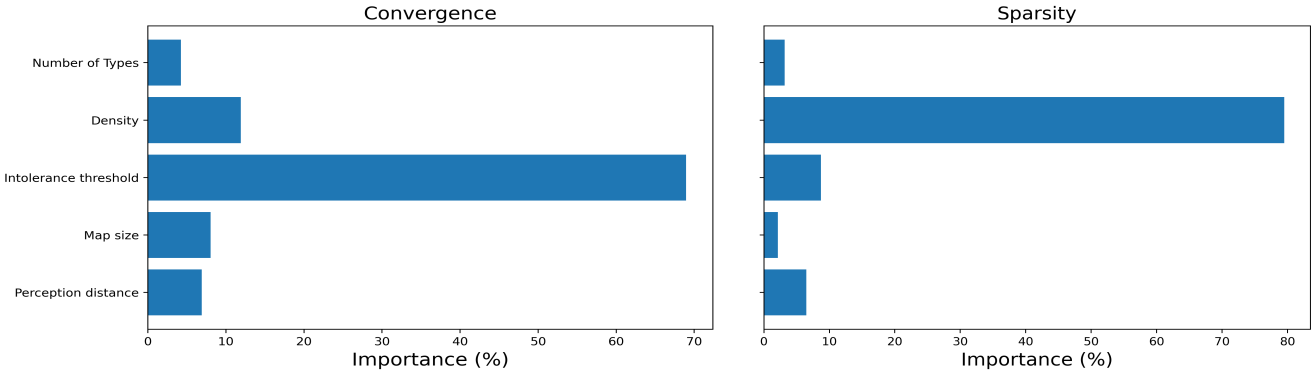


Figure 2: Variables importance in the random forest based on MDI.

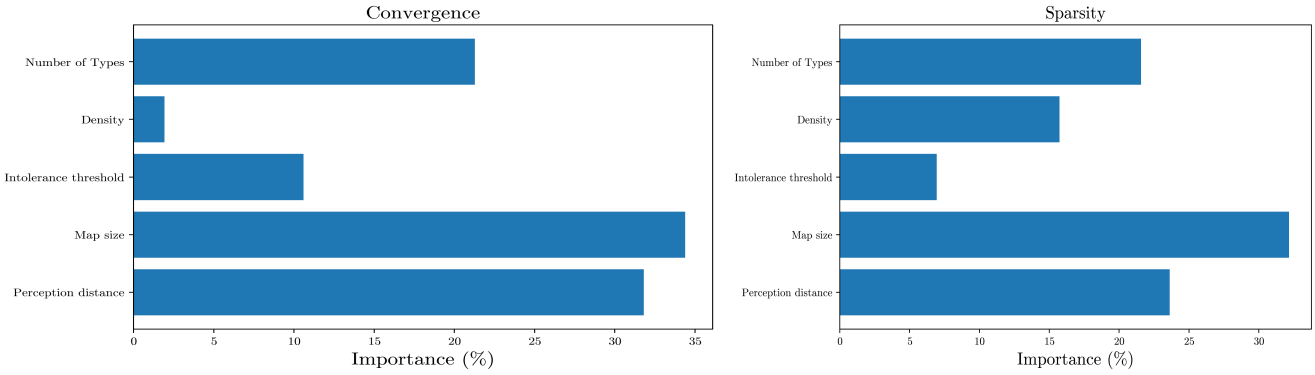


Figure 3: Variables importance in the k-Nearest Neighbors based on SHAP.

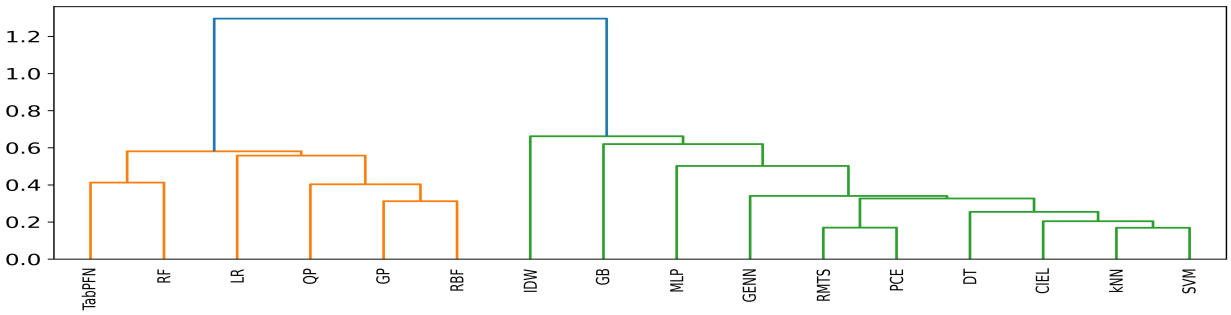
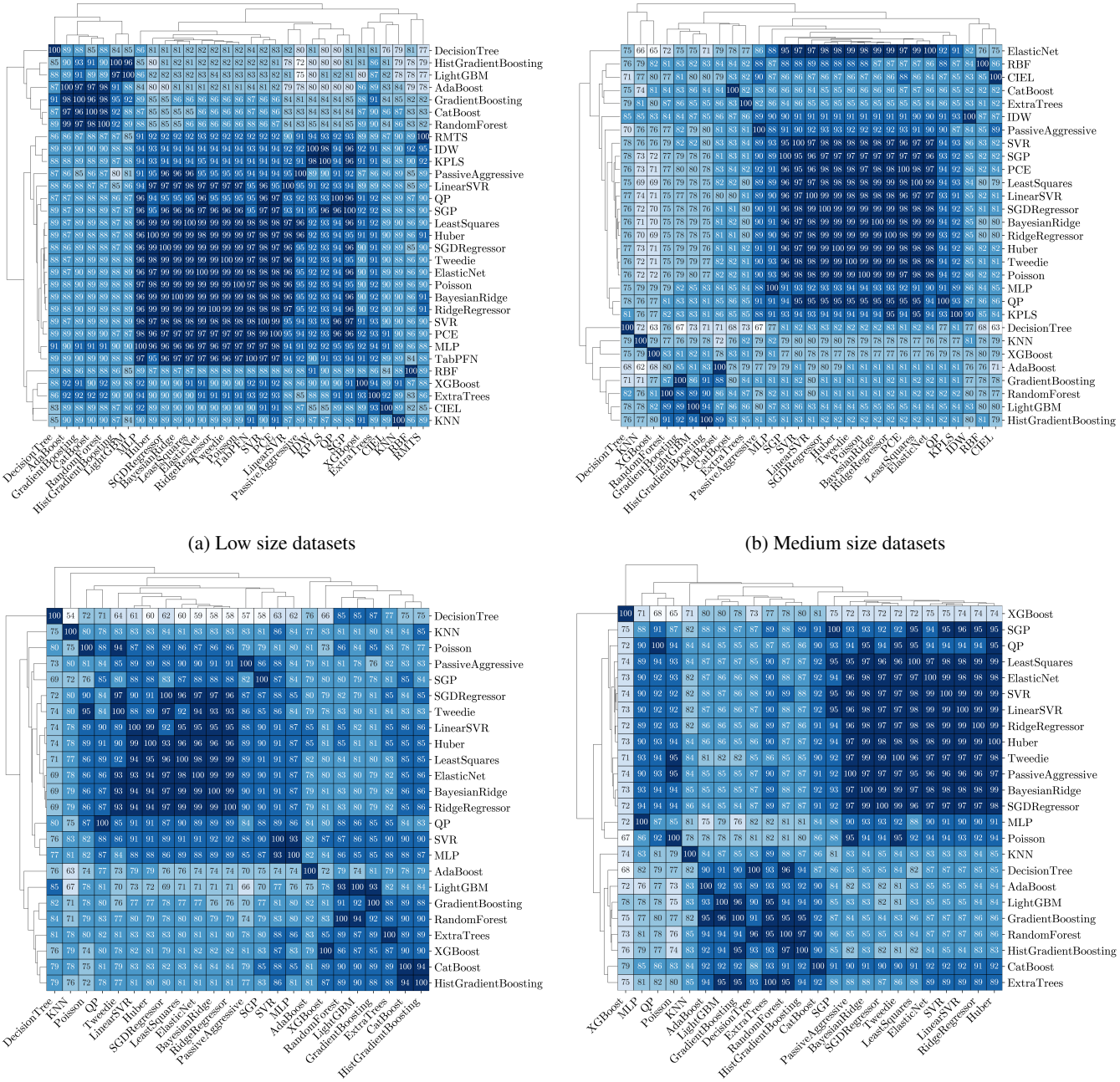


Figure 4: SHAP values NDCG based hierarchical clustering.

C.2 Regression datasets

Figure 5 displays, for all 4 sizes of Regression datasets (low, medium, high, and very high), the average NDCG adequacy of every pair of models in terms of SHAP values importance prediction on the validation test dataset. These SHAP prediction analyses reveal two subgroups of models whose diagonal elements belong to the subgroups with important internal NDCG scores and lower intergroup relationships, highlighting two groups of surrogate models sharing similar SHAP-based feature importance patterns, seemingly the local and the global models.



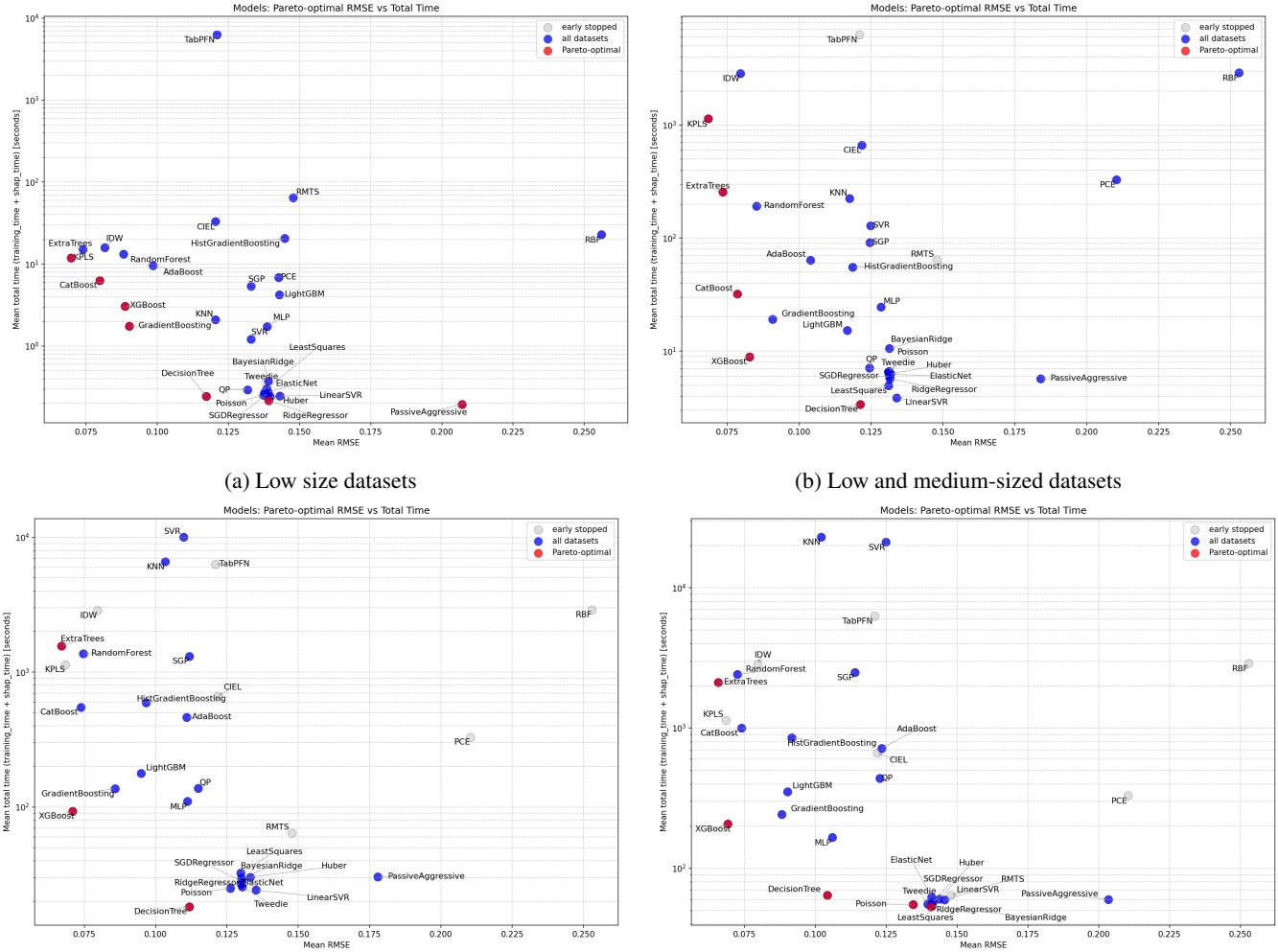
(a) Low size datasets

(b) Medium size datasets

(c) High size datasets

(d) Very-high size datasets

Figure 5: Average NDCG agreement for the four dataset-size groups in Regression.



(a) Low size datasets

(b) Low and medium-sized datasets

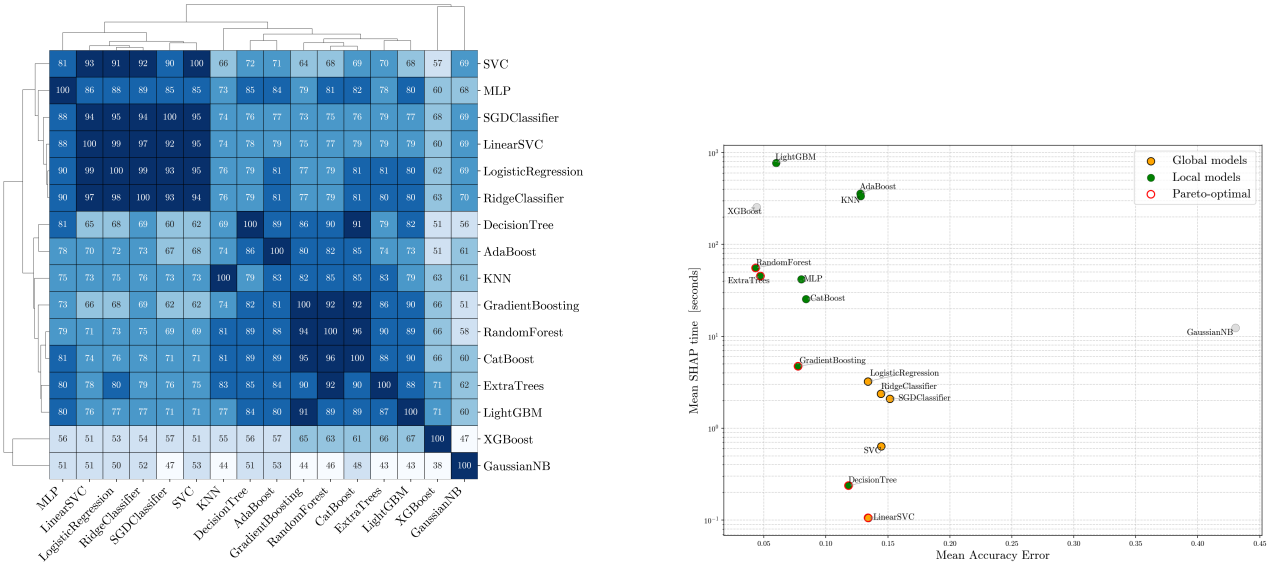
(c) Low, medium, and high size datasets

(d) All datasets

Figure 6: Computational time versus RMSE for the four dataset-size groups.

C.3 Classification datasets

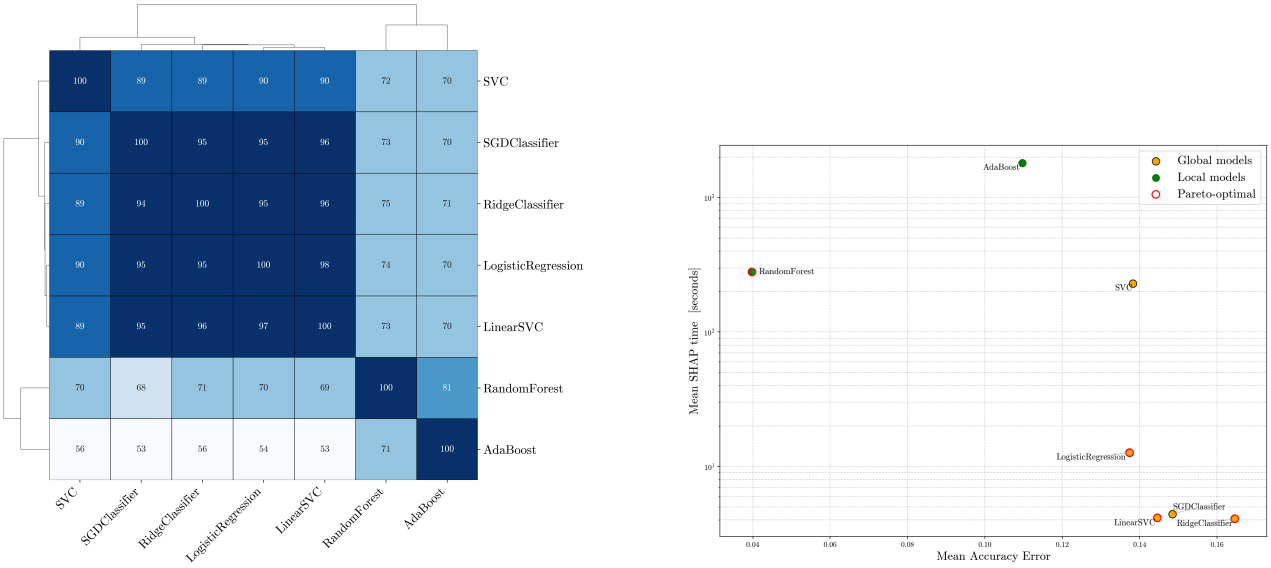
Figure 7 show the results on 4 small datasets (Breast cancer, diabetes, chess and hypothyroid) for all the 16 classification ML models. Both GaussianNB and XGBoost show odd results in terms of agreements, and, even if GaussianNB leads to a really high error, XGBoost seems highly accurate. Note that the intra-group agreements are higher on these small datasets that they were on median when considering more complex problems. Figure 8 show the results on the 15 datasets for 7 classification ML models. The agreement values are globally smaller which show that the more difficult the problem, the more interesting the diversity of ML models.



(a) NDCG matrix with all 16 models

(b) Pareto front with all 16 models

Figure 7: Average Classification results on small datasets.



(a) NDCG matrix with all 15 problems

(b) Pareto front with all 15 problems

Figure 8: Average Classification results on all datasets.