

Supplementary Material: A Feature Selection Layer Approach for Multi-Class Classification on Neural Networks

João Pedro Silveira e Silva^a, Márcio Dorn^{a,b,c}, Bruno Iochins Grisci^{a,1}

^a*Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil*

^b*Center for Biotechnology, Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil*

^c*National Institute of Science and Technology - Forensic Science, Porto Alegre, RS, Brazil*

Abstract

This supplementary material aims to present additional information and experiments not included in the original article. We begin with an introduction to feature weighting algorithms (Section 1) and how to evaluate them with a focus on multi-class problems (Section 2). Then we present complementary experiments results (Section 3). The complementary experiments start with an analysis of the weighted t-SNE using the synthetic datasets under an idealized scenario, where a “perfect” feature-weighting algorithm assigns a weight of 1 to all informative features and 0 to all non-informative ones (Figure 1). Then we present the t-SNE results using the microarray datasets without feature weights (Figure 2). These results serve as a benchmark for comparison with the outcomes produced by real algorithms. Figures from 3 to 9 and Tables 1 to 3 presents complementary experiments comparing FSL and the proposed extensions. Figures from 10 to 14 and Table 7 presents complementary experiments comparing Multiple FSL with other feature weighting algorithms. Table 8 summarizes the feature weighting methods evaluated in this study alongside their key characteristics. Table 9 presents the complete results of the Expected Calibration Error (ECE) comparison across all datasets and embedded feature weighting methods.

*Corresponding author

Email addresses: jpedross1999@gmail.com (João Pedro Silveira e Silva), mdorn@inf.ufrgs.br (Márcio Dorn), bigrisci@inf.ufrgs.br (Bruno Iochins Grisci)

1. Feature weighting algorithms

In this section, we introduce five consolidated feature weighting algorithms, the Kruskal–Wallis filter (Section 1.1), ReliefF (Section 1.2), Decision Tree (Section 1.3), Random Forest (Section 1.4) and LASSO (Section 1.5).

1.1. Kruskal–Wallis filter

The Kruskal-Wallis test [1] is a non-parametric statistical test that evaluates variation within and between groups to determine if a group of random variables originates from the same distribution [2]. Kruskal-Wallis test generates feature weights that are particularly useful as a filter which identifies features that significantly differentiate between labels within a dataset [3]. Given a labeled dataset with L labels, to test whether a feature f is relevant to the label, first f 's values are partitioned into L groups according to their label membership. Then, the Kruskal-Wallis statistic is calculated to determine if these L groups are significantly different [4]. The Kruskal-Wallis test is computed by calculating the H value using Equation 1 and 2. In these equations, L is the number of compared groups, n_i is the number of samples in the i -th group, N is the total number of samples across all groups, and R_i is the sum of the values in the i -th group [2].

$$N = \sum_{i=1}^L n_i \quad (1)$$

$$H = \frac{12}{N(N+1)} \sum_{i=1}^L \frac{R_i^2}{n_i} - 3(N+1) \quad (2)$$

1.2. ReliefF

ReliefF [5] is an extension of Relief [6] that can handle multi-class problems. It is also more robust and capable of dealing with incomplete and noisy data [7], being a popular choice of filter method through feature weighting [2]. The pseudo-code for ReliefF is described in Algorithm 1. It starts with all feature weights, W , set to zero. For each given instance R_i , it searches for k of its nearest neighbors from the same label, $\text{label}(R_i)$, called nearest hits H , and also k nearest neighbors from each of the

different labels, called nearest misses M_l . Then it updates the weights estimation W for all attributes A based on their values for R_i , hits H_j , and misses M_l . The function *diff* calculates the probability that two given instances have different values for a given feature, conditioned on a label value. The contribution for each label of the misses is weighted with the prior probability of that label, $P(L)$, which is estimated from the training set, divided by $1 - P(\text{label}(R_i))$ (the sum of probabilities for the misses labels) to keep the values between 0 and 1 [7, 3].

Algorithm 1: ReliefF

Inputs :

R: Set of all instances
A: Set of all features
L: Set of all labels
k: Number of neighbors to lookup

Output:

W: Vector of feature weights

```

1  $W_{0...a} \leftarrow 0.0$ 
2  $m \leftarrow |R|$ 
3 for  $i \leftarrow 1$  to  $m$  do
4    $H \leftarrow k\_nearest\_hits(k, R_i, R)$ 
5   for  $l \in L \neq \text{label}(R_i)$  do
6      $M_l \leftarrow k\_nearest\_misses(k, l, R_i, R)$ 
7   for  $a \leftarrow 1$  to  $|A|$  do
8     for  $hit \in H$  do
9        $W_a \leftarrow W_a - \frac{diff(a, R_i, hit)}{m * k}$ 
10    for  $l \in L \neq \text{label}(r)$  do
11      for  $miss \in M_l$  do
12         $W_a \leftarrow W_a + \frac{P(l)}{1 - P(\text{label}(R_i))} * \frac{diff(a, R_i, miss)}{m * k}$ 
13 return  $W$ 

```

The hyperparameter k is defined by the user and controls the locality of the estimates [7]. For most purposes it can be set to 10 [5].

1.3. Decision Tree

Decision Tree is a hierarchical, rule-based model that recursively splits the feature space into subdomains until only single-label regions remain [8]. It consists of branch nodes (multi-label) and leaf nodes (single-label). Predictions follow a path through branch nodes based on input features until reaching a leaf. Common splitting criteria

include Information Gain and Gini Index [9]. In Decision Tree models, feature relevance can be computed as the (normalized) total reduction of the criterion brought by that feature, also known as Gini importance.

1.4. Random Forest

Random Forest is an ensemble of Decision Trees designed to reduce overfitting by combining multiple models. Each tree votes on the label for a given input, improving accuracy but reducing interpretability [10]. Feature relevance is computed by averaging the importance scores from all trees, yielding a single feature weight vector [2].

1.5. Least absolute shrinkage and selection operator (LASSO)

LASSO [11] is a powerful embedded method used for regularization and feature weighting. It penalizes the coefficients of regression variables, shrinking some of them to zero through l_1 regularization. For feature selection, the features with non-zero coefficients are selected to be part of the model. Consider the linear regression model in Equation 3, where $x_1, x_2, \dots, x_p$ are the values of the p features for an input, y is the output, $\beta_1, \beta_2, \dots, \beta_p$ are the coefficients, and ϵ is the error. Given an input matrix $X_{n \times p}$, in which each row denotes a vector of p features for an input n , and a vector Y_n , in which each value represents the correct output for each input n . Then, for a given input with features $x_1, x_2, \dots, x_p$, the estimated response $\hat{y}$ is predicted by Equation 4.

$$y = \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p + \epsilon \quad (3)$$

$$\hat{y} = \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \dots + \hat{\beta}_p x_p. \quad (4)$$

The ordinary least squares estimate, $\hat{\beta} = \{\hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_p\}$, is obtained by minimizing the residual sum of squares, as formulated in Equation 5.

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p \beta_j x_{ij} \right)^2. \quad (5)$$

The LASSO algorithm introduces an extra penalty into Equation 5, generating Equation 6, where $\lambda \sum_{j=1}^p |\beta_j|$ is called the LASSO penalty, also known as l_1 regularization, and λ is a non-negative hyperparameter. The regularization causes the $\hat{\beta}$

values to continuously shrink towards zero as the parameter λ increases. The coefficients $\hat{\beta} = \{\hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_p\}$ shrink exactly to zero if λ is sufficiently large, which implies that all variables are eliminated [11, 12].

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| \quad (6)$$

There are many advantages in using LASSO. It provides good prediction accuracy by shrinking and removing coefficients, which can reduce variance without a substantial increase in bias, being especially useful in “large p, small n” problems. LASSO also enhances model interpretability by eliminating irrelevant features that are not associated with the response variable, thereby reducing overfitting.

2. Feature weighting evaluation

In Section 1, we presented various feature weighting algorithms, each employing distinct strategies and metrics to identify relevant features and assign weights accordingly. Given an arbitrary dataset and a feature weighting algorithm, evaluating the quality of the generated feature relevance is a complex task. To address this challenge, we propose different evaluation methods. First, we utilize synthetic datasets to assess the ability of feature weighting algorithms to identify truly relevant features (Section 2.1). Second, we employ feature erasure (Section 2.2) in conjunction with classification metrics to evaluate the impact of most relevant features, ordering by weight, on the model performance. Third, we assess the performance of new models trained on subsets of selected features with and without using the feature weights (Section 2.3). Fourth, we utilize weighted t-SNE (Section 2.4) to visualize how the feature weights can differentiate between different labels compared with the standard t-SNE. Finally, we measure and compare the stability of different feature weighting algorithms (Section 2.5).

2.1. Informative features

Feature weighting algorithms are particularly valuable for problems with a high number of features and a limited number of samples which makes difficult to obtain

good classification performance and interpretability on most models. As we lack complete control over the complex interactions and correlations between features and labels in high-dimensional real-world datasets, we use synthetic datasets where we can explicitly define the relevant features for each label. Given this precise knowledge of feature relevance, we can compute two key metrics:

- **Percentage of informative features selected (PIFS)**: Measures the proportion of informative features included in a feature subset. It is calculated by dividing the number of informative features present in the subset by the total number of informative features available in the dataset. A score of 1 indicates that all informative features are present in the subset, while a score of 0 indicates that none of the informative features are present.
- **Percentage of selected features that are informative (PSFI)**: Evaluates the proportion of informative features within a given feature subset. It is calculated by dividing the number of informative features present in the subset by the total number of features in the subset. A score of 1 indicates that all features in the subset are informative, while a score of 0 signifies that none of the features in the subset are informative.

Equation 7 and 8 outline the calculation of these metrics, where $S_{selected}$ is a set of selected features and $S_{informative}$ is the set of informative, or relevant, features for a given dataset [2]. For feature weighting algorithms, these metrics can be calculated multiple times by considering different top- n feature subsets defined by a threshold.

$$PIFS = \frac{|S_{selected} \cap S_{informative}|}{|S_{informative}|} \quad (7)$$

$$PSFI = \frac{|S_{selected} \cap S_{informative}|}{|S_{selected}|} \quad (8)$$

2.2. Feature erasure

Feature erasure is a method for assessing the relevance of features subsets in a trained model, adapted by Grisci et al. [3] for tabular data from the perturbation analysis

presented by Samek et al. [13]. It involves systematically masking specific features and observing the impact on the model’s predictive performance. This is achieved by setting the values of the target features to zero. The value zero is chosen because, given the initial data normalization, all input features have a mean equal to zero. If a feature is crucial for the model’s predictions, masking it should lead to a significant decrease in performance. Conversely, masking irrelevant features should have a minimal impact or even increase the predictive power.

This technique is primarily applicable to feature weighting algorithms that are integrated into the model training process or applied post-hoc. For algorithms that access feature weights independently of a model, the evaluation approach described in Section 2.3 is more appropriate.

2.3. Selection performance on new models

Different machine learning predictors, as Decision Trees, SVMs, and Deep Neural Networks can be employed to assess the predictive power of different feature subsets. For each feature subset generated by selecting the most relevant features given by a feature weighting algorithm, we train a predictor and evaluate it’s predictive performance using the metrics as F1 score, precision, recall, and accuracy.

2.4. Weighted t-SNE

t-SNE (t-Distributed Stochastic Neighbor Embedding) is a popular technique for visualizing high-dimensional data in two dimensions, originally proposed by van der Maaten and Hinton [14] as a variation of SNE (Stochastic Neighbor Embedding), introduced by Hinton and Roweis [15], using the Student t-distribution with one degree of freedom. It calculates the distances between high-dimensional data points, typically using the Euclidean distance (Equation 9).

$$d(p, q) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2} \quad (9)$$

Weighted t-SNE, proposed by Grisci et al. [3], introduces a weighted Euclidean distance (Equation 10), adapted from Carroll and Chang [16], to enhance the visualization based on features weights given by a feature weighting algorithm. Those weights

range from 0 (less relevant) to 1 (more relevant). When scaling each dimension by its weight, dimensions with higher weights account for a greater portion of the distance between points and thus have more influence on their final positions in the visualization. Another significant advantage of weighted t-SNE is its ability to create visualizations based on label-specific feature weights, allowing us to use the weights from a specific label to determine if they can better distinguish it from the others.

$$d_w(p, q) = \sqrt{\sum_{i=1}^n (w_i(q_i - p_i))^2} \quad (10)$$

2.5. Stability

Feature weighting is not only crucial for improving model performance but also plays a significant role in research and interpretability. For instance, as an example, in Bioinformatics, it can help identify biomarkers associated with specific diseases. In such domains, the stability of a feature weighting algorithm is an important metric, as unstable algorithms can lead to non-replicable results. The issue of non-replicable gene selections has been a major concern for many researchers [17]. And several factors can contribute to unstable feature weighting, some of them are [18]:

- **Multiple informative features:** The presence of multiple highly correlated features can make it difficult for algorithms to consistently select the most relevant ones.
- **High-dimensional, low-sample data:** In such scenarios, feature weighting algorithms may be prone to overfitting and unstable results.

Stability metrics assess the consistency of feature selection and weighting results across different data samples. Data perturbation, such as adding, removing, or resampling instances, can be used to simulate variations in the training data [2]. K-Fold Cross-Validation is a widely used technique to evaluate model performance that can be used to measure feature weighting stability [17]. In this context it involves:

- **Splitting the dataset:** Divide the dataset into k multiple stratified folds.

- **Execute feature selection:** Execute the feature weighting method using $k - 1$ folds and persist the result which can be a simple feature subset, a rank or a weighted features set.
- **Repeating:** Repeat the process k times using all available folds $k - 1$ times each.

By comparing different subsets of features based on their weights from each execution, we can assess the stability of the method using different metrics, considering different perspectives over the output. For this work, we employ Jaccard, Spearman, and Pearson similarity measures:

- **Jaccard index:** Measures the similarity between two sets, in our case the top n weighted features. It ranges from 0 (no overlap) to 1 (identical sets), being defined by:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}, \quad (11)$$

where A and B are the two subsets being analyzed.

- **Spearman's rank correlation coefficient:** Measures the relationship between two ranked lists. It ranges from -1 (linearly inverse correlated) to 1 (linearly correlated), where 0 indicates no correlation. It is defined by:

$$r_s(p, q) = 1 - \frac{6 \sum_{i=1}^n (p_i - q_i)^2}{n(n^2 - 1)}, \quad (12)$$

where p and q are two rankings presented as permutation lists and n is the number of element in the rankings.

- **Pearson correlation coefficient:** Measures the linear correlation between two sets of numerical values, being used to measure the correlation between feature weights. It's boundaries are the same of Spearman's rank correlation coefficient, being defined by:

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}} \quad (13)$$

where x and y represent each subset, $\bar{x}$ and $\bar{y}$ represent the mean value of each subset and n represents the subset size.

3. Experiments

In this section, we introduce complementary experiments that further support our findings. All neural networks shared a common architecture of four hidden layers (100, 200, 100, 100), with 10% Dropout applied between layers. The models were optimized using AdamW with a learning rate of 0.001 and evaluated using Stratified K-Fold cross-validation ($K = 11$). The test split was set to 20% for synthetic datasets and 15% for microarray datasets. Regarding regularization, FSL and its extensions used $\lambda = 0.001$ for synthetic and $\lambda = 0.0025$ for microarray datasets, while Lasso used $\lambda = 0.01$. Batch sizes and training epochs varied by dataset: synthetic datasets used a batch size of 32, while Prostate and Breast used 16, and Colorectal and Liver used 8. Training duration was 2000 epochs for XOR, 800 for Prostate, 400 for Breast, and 300 for the remaining datasets. Finally, for interpretability baselines, DeepSHAP and LIME utilized 1000 background samples for synthetic data (100 for SAGE). However, due to data dimensionality constraints, all methods were limited to 50 background samples for the microarray datasets. More details are available in the provided source code.

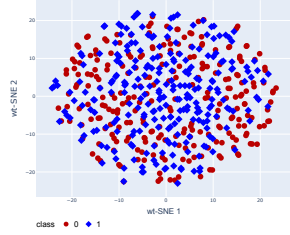
Figure 1 presents the t-SNE and a “perfect” weighted t-SNE for each synthetic dataset. The “perfect” weighted version assigns a weight of 1.0 to all informative features and 0.0 to all non-informative ones. Figure 2 shows the t-SNE visualizations for each microarray dataset. Since the informative features are unknown for these datasets, a “perfect” weighted t-SNE cannot be constructed.

Figure 3 reports the PIFS and PSFI evaluations on the XOR dataset, comparing the original FSL with the proposed extensions.

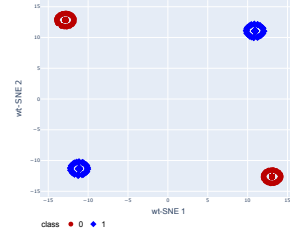
Figure 4 shows a heatmap of the 15 highest-weighted features for the XOR dataset for each extension and for the original FSL. Figure 5 presents the feature-ranking positions across multiple executions of FSL and the extensions. The goal is to determine whether the informative and non-informative features form clearly separated ranking groups. Since the models produce non-stable rankings, the results suggest a persistent separation between informative and non-informative features. Finally Figure 6 presents the weighted t-SNE results.

Figure 7 presents the PIFS (percentage of informative features selected) and PSFI

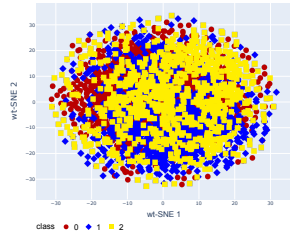
Figure 1: t-SNE and “perfect” weighted t-SNE of each synthetic datasets. The “perfect” weighted t-SNE is achieved by assigning a weight of 1 for all informative features, while irrelevant features were assigned a weight of 0.



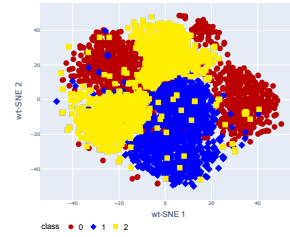
(a) XOR t-SNE



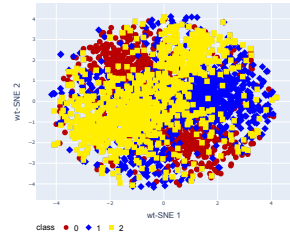
(b) XOR “perfect” weighted t-SNE



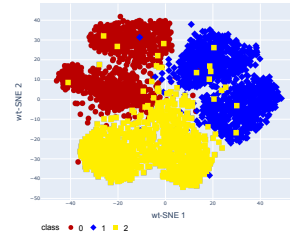
(c) SynthA t-SNE visualization



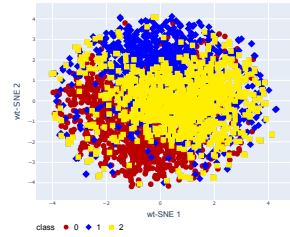
(d) SynthA “perfect” weighted t-SNE visualization



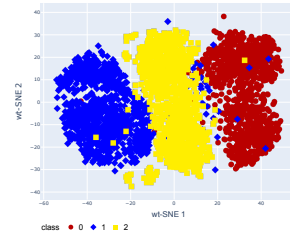
(e) SynthB t-SNE visualization



(f) SynthB “perfect” weighted t-SNE visualization

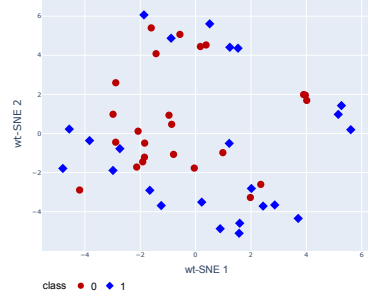


(g) SynthC t-SNE visualization

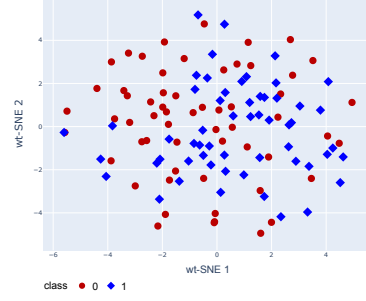


(h) SynthC “perfect” weighted t-SNE visualization

Figure 2: **t-SNE of CuMiDa datasets.** Analysis of each CuMiDa datasets used on this work. A weighted t-SNE approach was not implemented due to the lack of prior knowledge about the relevance of each feature.



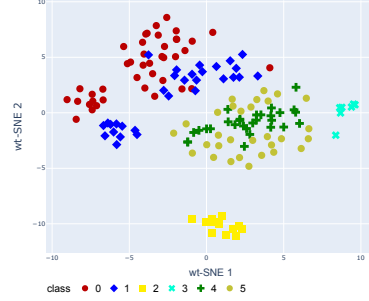
(a) Liver t-SNE visualization



(b) Prostate t-SNE visualization



(c) Colorectal t-SNE visualization



(d) Breast t-SNE visualization

(percentage of selected features that are informative) analyses for SynthA, SynthB, and SynthC across increasing selection subsets, comparing the original FSL with the proposed extensions. This analysis highlights how consistently informative features receive higher weights across multiple runs.

Table 1 reports the HD and 2D silhouette coefficients for the SynthA dataset based on the feature weights produced by the original FSL and the extensions, with corresponding visual results shown in Figure 8.

Figure 9 exhibits a heatmap with the normalized weights of the top-30 features ordered by the general features weights generated by each proposed extension over SynthB dataset. Relevant features are prefixed with *informative_for_labels_* followed by an identification character sequence and the directly related label. As example, *informa-*

Figure 3: **Evaluation of PIFS and PSFI on the XOR dataset between original FSL and proposed extensions.** All scores are the average over rankings generated by 11 independent execution. Vertical axis represents the score constrained from 0 to 1 as both measures are percentages. Horizontal axis represents the subset size being evaluated. All subsets are defined by the top- n features from a ranking based on the feature weights generated by each algorithm.

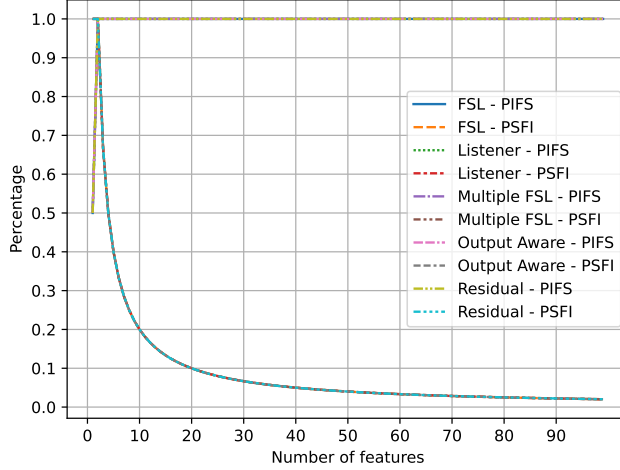


Table 1: **SynthA dataset HD and 2D silhouette coefficients based on the feature weights attributed by original FSL and the proposed extensions.** All feature weights were provided from the general weights, not being label-specific. High-dimensional (HD) silhouette coefficients are calculated before the dimension reduction performed by weighted t-SNE, while 2D silhouette coefficients are calculated after. “None” represents the silhouette coefficients without adding feature weights. In bold is the highest average. Best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis and Dunn’s test are highlighted in orange.

Weight source	HD silhouette	2D silhouette
None	.0030	-.013
FSL	.0105±.0005	.1755±.0560
Listener	.0109±.0003	.1291±.0433
Residual	.0109±.0004	.1755±.0560
Output Aware	.0128±.0005	.1696±.0693
Multiple FSL	.0123±.0004	.1773±.0922

tive_for_labels_S_0 is a relevant feature uniquely identified by the character sequence “S” and relevant only for label 0 (suffix “_0”). As shown in the heatmap, Multiple FSL and Output Aware assigned the highest weights to label-relevant features. Listener assigned highest weight values to all relevant features despite their relation with only a specific label, struggling to accurately prioritize features by their relative label. The *informative_for_labels_R_1* is an example of feature relevant only for label 1 assigned

Figure 4: **Heatmap visualization with the 15 highest weighted features for XOR dataset.** Normalized weights for each feature ranked from the highest general weight to the lowest. “General” represents the average label-specific weights for the proposed extensions Residual, Output Aware and Multiple FSL.

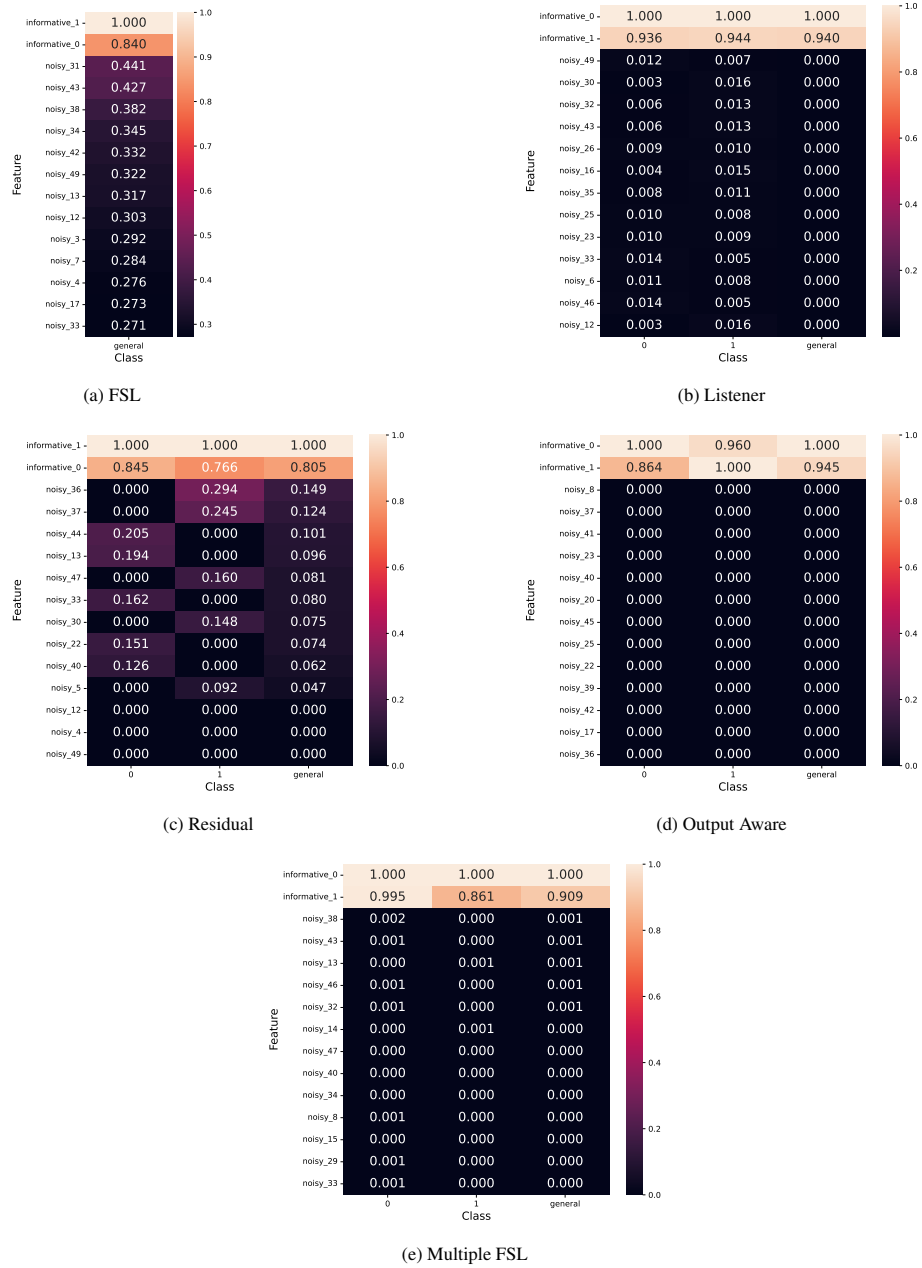
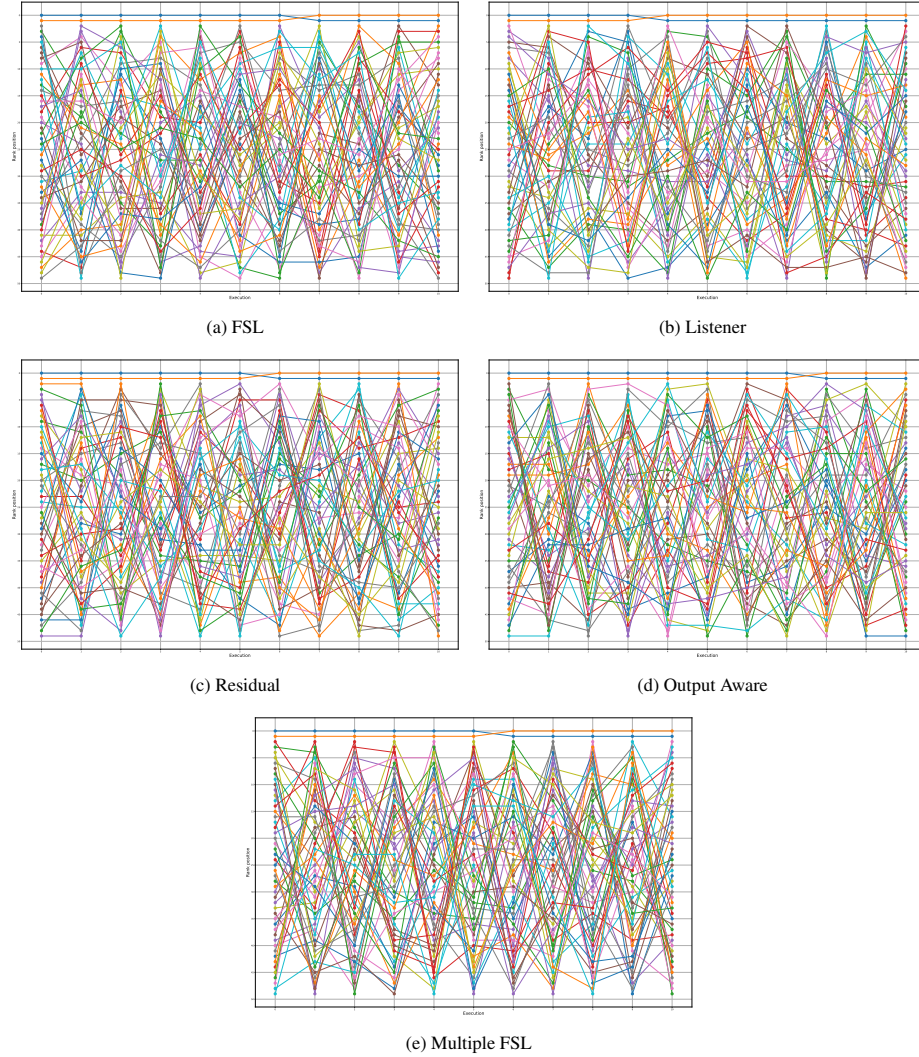
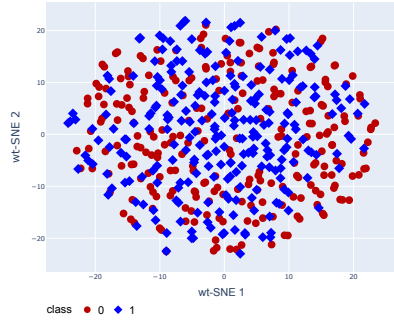


Figure 5: **Feature ranking positions across FSL and proposed extensions executions on XOR dataset.** All rankings were created by sorting the features in descending order of their weights. Each algorithm was executed 11 times (horizontal axis). The horizontal axis represents each individual execution, while the vertical axis shows the rank position of each feature, from 0 (top) to 49 (bottom), within that execution. Each color represent a different feature, with blue and orange lines at the top indicating the two relevant features.

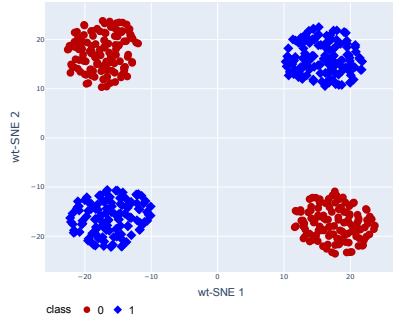


with similar normalized weights for all labels by Listener, being 0.822 for label 0, 0.9 for label 1 and 0.917 for label 2. Residual exhibited a more inconsistent behavior. In some cases, such as *informative_for_labels_I_0*, it correctly assigned the highest weight to the relevant label. However, in other cases, like *informative_for_labels_S_0* and *in-*

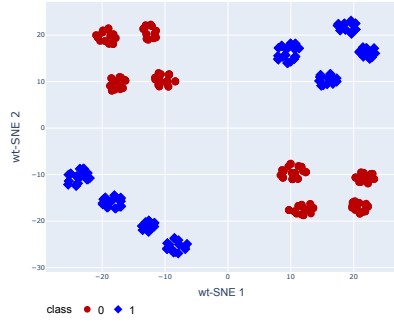
Figure 6: **Visualization of XOR dataset with weighted t-SNE using the feature weights attributed by original FSL and the proposed extensions.** Figure 6a presents a t-SNE visualization of the data without feature weights. In contrast, 6b, 6c, 6d, 6e and 6f employ weighted t-SNE, utilizing the generated weights from original FSL and each proposed extension to enhance the separation between clusters. False labels (0) are represented in red, while true labels (1) are represented in blue.



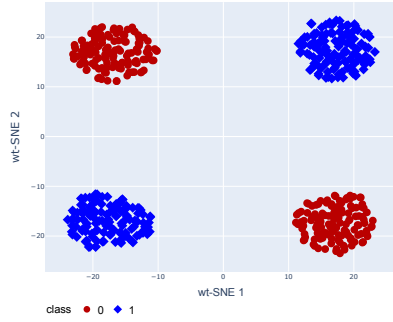
(a) Without weights



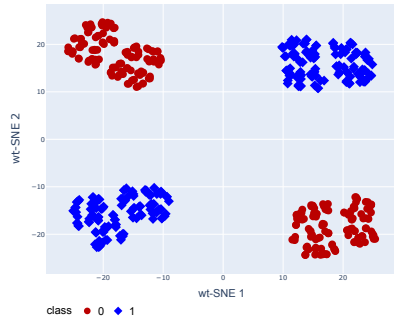
(b) FSL



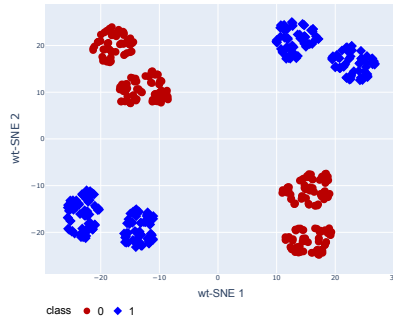
(c) Listener



(d) Residual

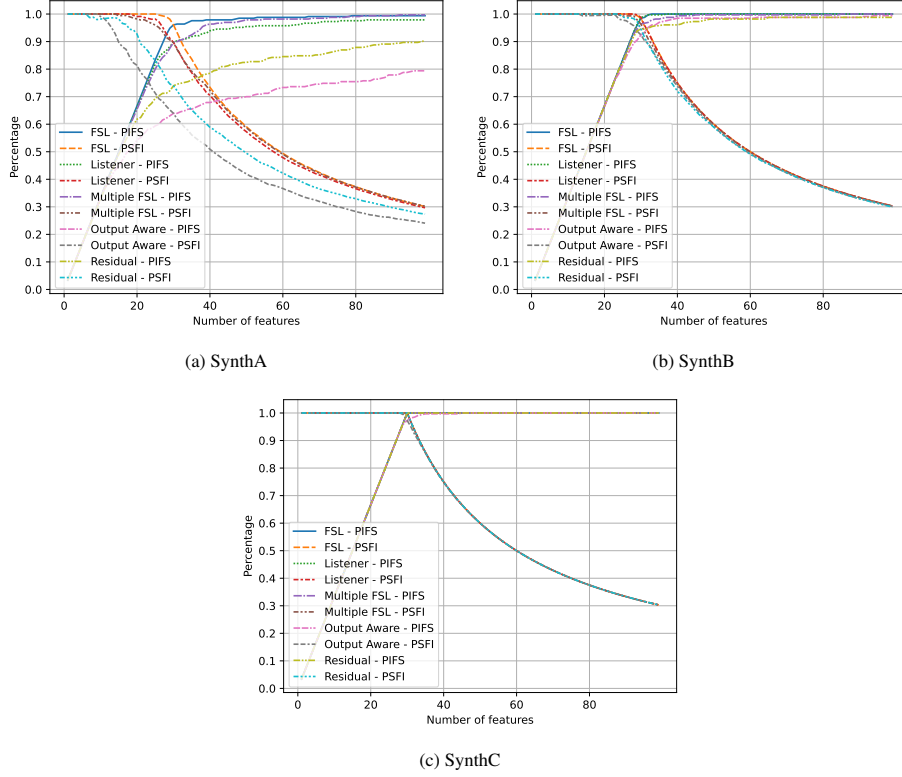


(e) Output Aware



(f) Multiple FSL

Figure 7: **Analysis of PIFS and PSFI on SynthA, SynthB and SynthC datasets between original FSL and proposed extensions for increasing selection subsets.** All values scores are the average over rankings generated by 11 independent models execution. Vertical axis represents the scores constrained from 0 to 1 as both measures are percentages. Horizontal axis represents the subsets size selected from the top-n features with the highest weights. Figure 7a represents the evaluation over SynthA dataset, Figure 7b over SynthB and Figure 7c over SynthC.

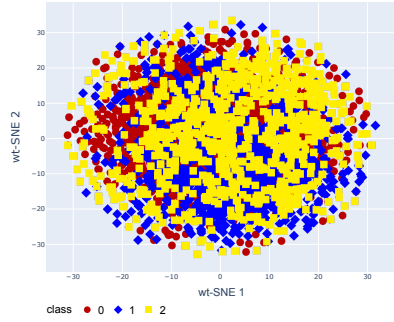


formative_for_labels_N_0, it assigned the lowest weights to the correct label.

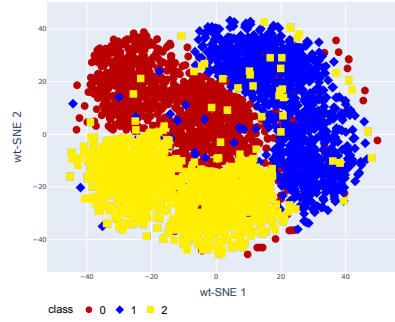
When comparing feature weights on SynthC, we observed instances where features related to two labels are sometimes considered more relevant to classify a third label, as exemplified in Table 2 for the feature *informative_per_label_L_0_1* (relevant for the labels 0 and 1) where the highest weight was associated with label 2 instead of 0 and 1. One possible hypothesis is that the model prioritizes those attributes for differentiating the third label instead of defining the first and second labels that have informative values.

Because FSL and its extensions affect model training, Table 3 reports their predic-

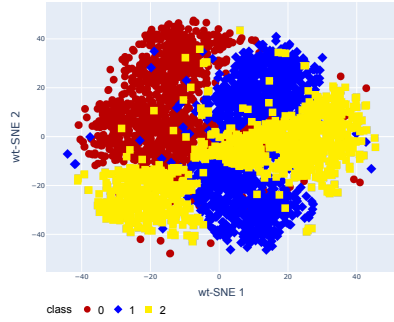
Figure 8: **SynthA weighted t-SNE visualization using the general feature weights attributed by original FSL and the proposed extensions.** Figure 8a presents a t-SNE visualization of the data without feature weights. In contrast, 8b, 8c, 8d, 8e and 8f employ weighted t-SNE, utilizing the generated weights from original FSL and all proposed extensions to enhance the separation between clusters. Label 0 is represented in red, label 1 in blue and label 3 in yellow.



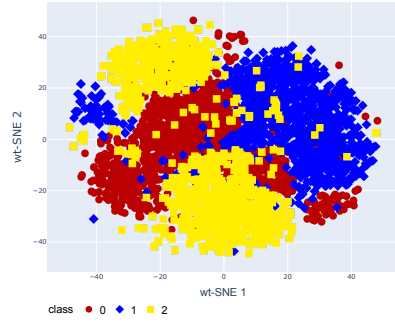
(a) Without weights



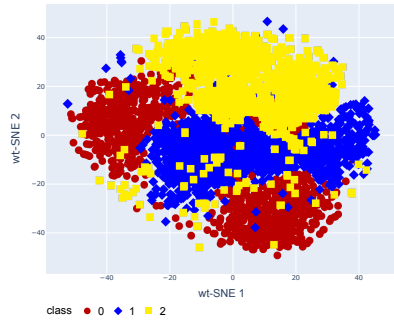
(b) FSL



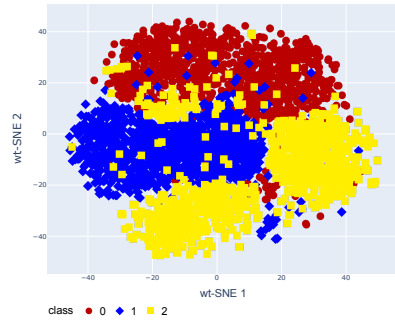
(c) Listener



(d) Residual



(e) Output Aware



(f) Multiple FSL

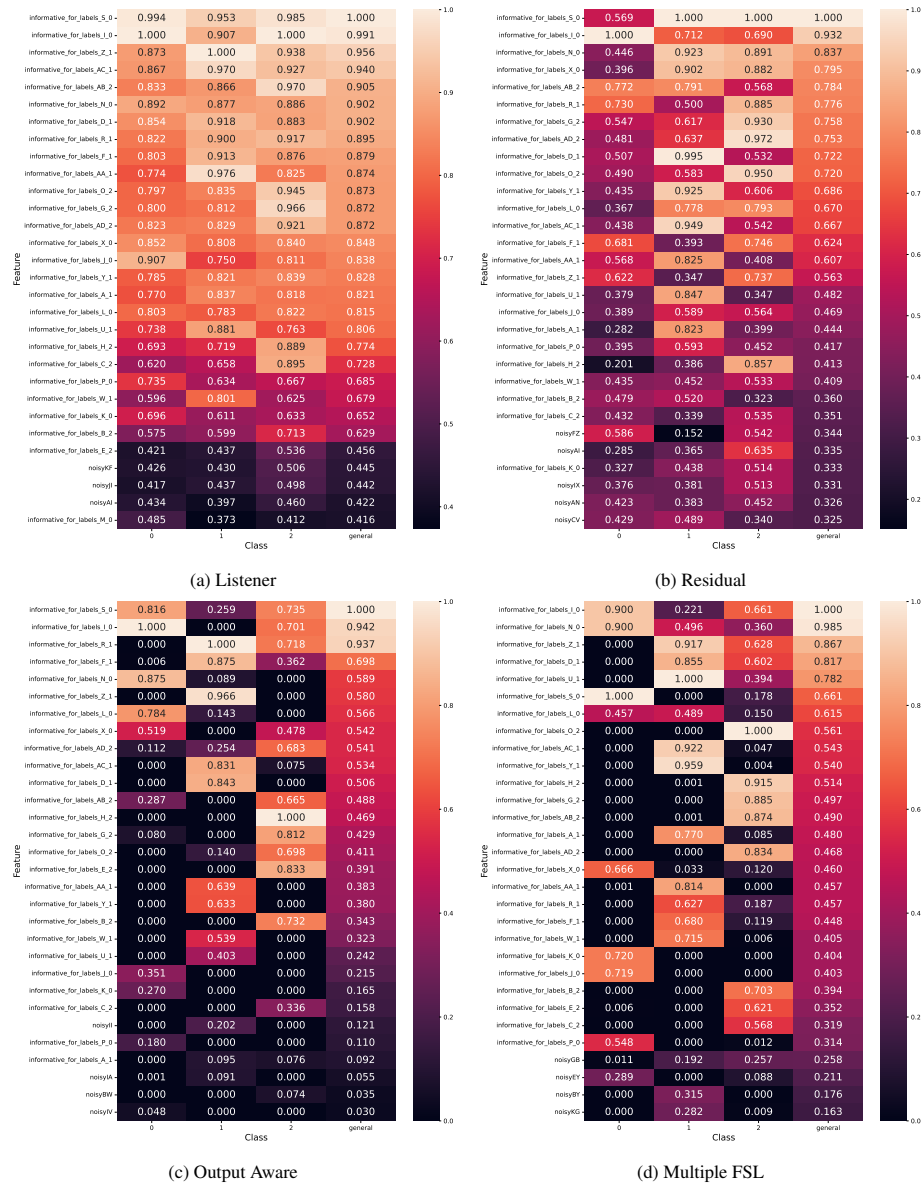


Figure 9: **Heatmap visualization of top-30 general features for each extension from SynthB dataset.** Normalized feature weights for each algorithm, ranked by their overall general weight. Feature names prefix indicate if they are informative or noisy, followed by an identification letter sequence, which ensures that all names are unique, and the label which they are related. For example, *informative_for_labels_A.1* is the informative feature identified by the letter 'A' that is related to label 1, while *noisyAI* is a noisy feature identified by the letter sequence "AI".

Table 2: **Analysis of feature weights for *informative.per.label.L.0.1* from SynthC dataset.** Each column represents the feature weight given by each algorithm to the feature *informative.per.label.L.0.1*. Although the feature is directly related to label 0 and label 1, all extensions gave a higher weight, highlighted in bold, related to label 2.

Algorithm	Label 0	Label 1	Label 2
Listener	.834	.727	.886
Residual	.398	.400	.862
Output Aware	.463	.513	.599
Multiple FSL	.341	.626	.811

tive performance on the Breast dataset.

?? compares the execution time of Multiple FSL with the other algorithms on the SynthA dataset. Figure 10 and Figure 11 present weighted t-SNE visualizations for Multiple FSL and the comparison algorithms, using general feature weights for the former and label-1-specific weights for the latter.

Table 4 presents a stability comparison on the SynthB dataset using the $m - 1$ approach. The results are consistent with the analysis of the SynthA dataset: Kruskal-Wallis demonstrated the highest stability, followed by Random Forest, ReliefF, and Lasso. Notably, despite relying on neural networks, both DeepSHAP and Multiple FSL exhibited high stability scores. Specifically, Multiple FSL achieved high Jaccard and Spearman scores, while DeepSHAP attained a slightly higher Pearson score.

Table 5 reports the HD and 2D silhouette coefficients for the Colorectal dataset using general feature weights generated by Multiple FSL, DeepSHAP, LIME, and Lasso, complementing the weighted t-SNE visualization in Figure 12.

Figure 13 shows the impact of feature erasure based on label 4 feature weights for the Breast dataset for Lasso, LIME, DeepSHAP, and Multiple FSL. And Figure 14 presents the weighted tSNE visualization using label 1 feature weights generated by Multiple FSL, LassoNet and TabNet.

Table 6 presents the label-specific F1 scores for the embedded feature weighting techniques capable of detecting label-specific weights. Multiple FSL achieved the highest scores, followed by LassoNet, and both successfully improved upon the baseline results. Table 7 extends this analysis to the Isolet dataset. Here, TabNet demonstrated superior performance across all labels, followed closely by Multiple FSL. Al-

Table 3: **Individual predictive performance of FSL and proposed extensions on Breast dataset per label.** The average scores were calculated across 11 independent executions using test dataset. For each reported metric we present the average and standard. In bold is the highest average of each metric and label. Best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis and Dunn’s test are highlighted in orange.

Algorithm	Label	Precision	Recall	F1 score
FSL	0	.990±.032	.977±.048	.983±.029
	1	.951±.084	.985±.048	.965±.049
	2	1.0±.0	1.0±.0	1.0±.0
	3	1.0±.0	1.0±.0	1.0±.0
	4	.974±.055	1.0±.0	.986±.030
	5	1.0±.0	.939±.080	.967±.044
Listener	0	.980±.043	1.0±.0	.989±.023
	1	.974±.055	.970±.064	.969±.041
	2	1.0±.0	1.0±.0	1.0±.0
	3	1.0±.0	1.0±.0	1.0±.0
	4	.964±.079	1.0±.0	.980±.045
	5	1.0±.0	.924±.109	.957±.064
Residual	0	.954±.079	.784±.193	.840±.104
	1	.581±.397	.403±.316	.403±.283
	2	1.0±.0	.727±.278	.809±.207
	3	.727±.445	.682±.440	.697±.437
	4	.639±.320	.818±.305	.818±.256
	5	.626±.06859	.682±.305	.536±.110
Output Aware	0	.941±.094	.898±.231	.906±.185
	1	.802±.282	.682±.297	.705±.246
	2	.868±.236	.970±.096	.890±.173
	3	.920±.254	1.0±.0	.928±.227
	4	.847±.280	.894±.287	.867±.278
	5	.864±.02307	.712±.285	.715±.155
Multiple FSL	0	.990±.032	1.0±.0	.995±.017
	1	.987±.041	.985±.048	.985±.033
	2	1.0±.0	1.0±.0	1.0±.0
	3	1.0±.0	1.0±.0	1.0±.0
	4	.961±.064	1.0±.0	.979±.034
	5	1.0±.0	.939±.107	.965±.063

though Multiple FSL yielded slightly lower scores for labels F and S compared to TabNet, it still outperformed the baseline and LassoNet. Conversely, Lasso yielded the poorest results on both the Isolet and Letter Recognition datasets.

Figure 10: **SynthA weighted t-SNE visualization using the feature weights attributed by Multiple FSL and compared algorithms.** Figure 10a presents a t-SNE visualization of the data without feature weights. In contrast other figures employ weighted t-SNE, utilizing the generated weights from each feature weighting algorithm.

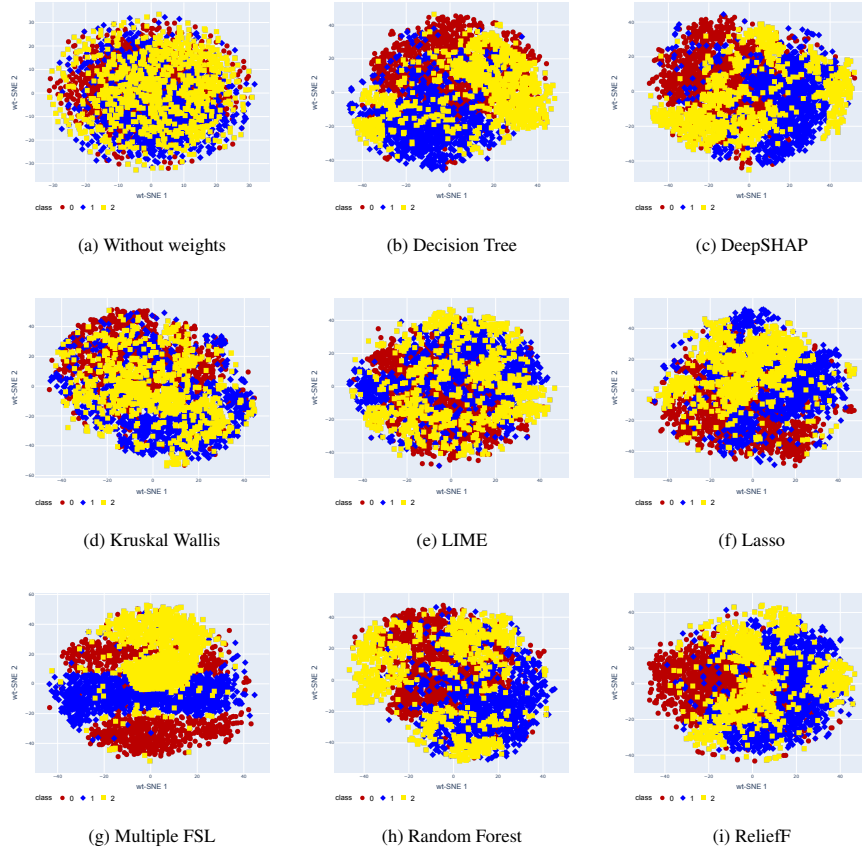
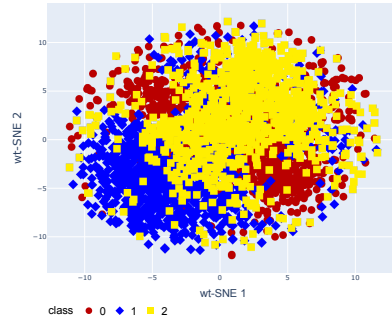


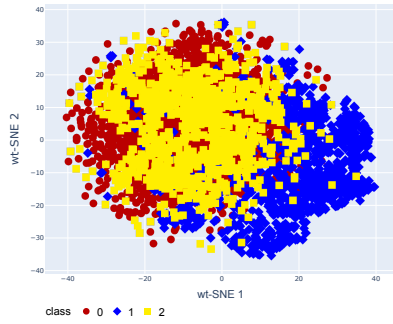
Figure 11: **SynthB weighted t-SNE visualization using the feature weights attributed to label 1.** Figure 11a presents a t-SNE visualization of the data without feature weights. In contrast other figures employ weighted t-SNE, utilizing the generated weights from each algorithm specifically related to label 1 (blue diamonds).



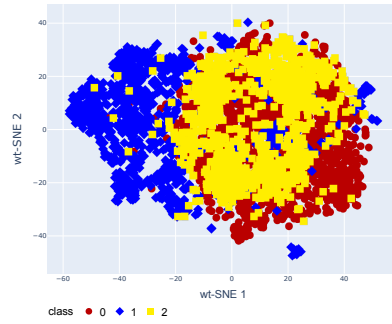
(a) Without weights



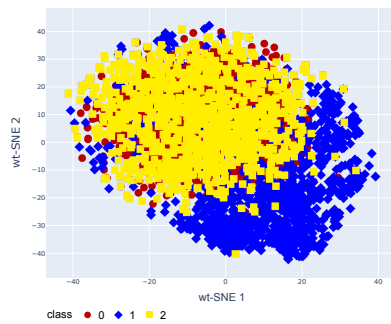
(b) Lasso



(c) LIME



(d) DeepSHAP



(e) Multiple FSL

Table 4: **Stability comparison between algorithms feature weights on SynthB dataset.** Jaccard similarity measures set overlap, ranging from 0 (no overlap) to 1 (identical sets), being used to assess the similarity between the top-n highest-weighted features. Spearman accesses ranking similarity, ranging from -1 (perfect inverse correlation) to 1 (perfect correlation). Pearson measures linear correlation between the feature weights, ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation) where 0 means no correlation. Pearson and Spearman was measured using all features (100), while Jaccard was measured using the top-30 features given by the feature weighting algorithm, which is the number of informative features. In bold is the highest average of each dataset and label. Best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis and Dunn’s test are highlighted in orange.

Algorithm	Pearson	Spearman	Jaccard
Decision Tree	0.728 $\pm$.087	0.330 $\pm$.040	0.536 $\pm$.044
Random Forest	0.989 $\pm$.003	0.342 $\pm$.039	0.947 $\pm$.026
Kruskal Wallis	0.999$\pm$.000	1.000$\pm$.000	1.000$\pm$.000
ReliefF	0.987 $\pm$.001	0.924 $\pm$.008	0.976 $\pm$.033
Lasso	0.989 $\pm$.001	0.840 $\pm$.035	0.753 $\pm$.061
LIME	0.874 $\pm$.026	0.162 $\pm$.052	0.388 $\pm$.050
DeepSHAP	0.920 $\pm$.010	0.420 $\pm$.040	0.752 $\pm$.046
Multiple FSL	0.906 $\pm$.011	0.630 $\pm$.032	0.876 $\pm$.051

Table 5: **Colorectal dataset HD and 2D silhouette coefficients with general feature weights.** “None” represents the t-SNE without using weighted features, being a fixed value. Averages were calculated with the weighted t-SNE results given the feature weights generated by each algorithm through 11 different executions. For the comparison was used the general feature weights without an specific related label. In bold is the highest average. Best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis and Dunn’s test are highlighted in orange.

Weight source	HD silhouette	2D silhouette
None	.024	.090
Lasso	.187$\pm$.011	.487 $\pm$.047
LIME	.072 $\pm$.001	.207 $\pm$.026
DeepSHAP	.096 $\pm$.005	.298 $\pm$.030
Multiple FSL	.159 $\pm$.015	.561$\pm$.021

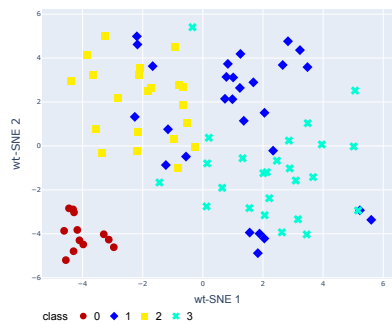
Figure 12: **Colorectal dataset weighted t-SNE visualization using general feature weights.** Figure 12a presents a t-SNE visualization without feature weights.



(a) Without weights



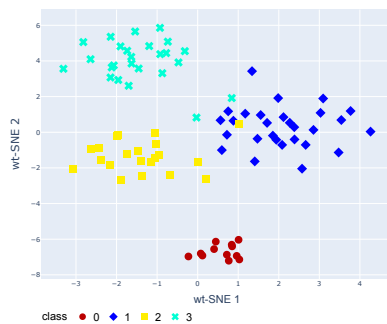
(b) Lasso



(c) LIME

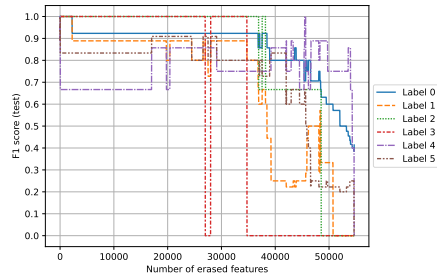


(d) DeepSHAP

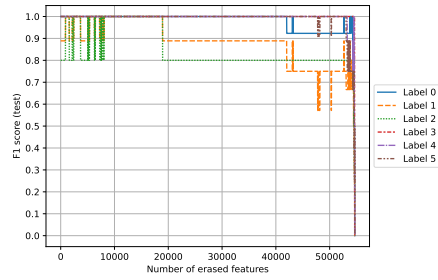


(e) Multiple FSL

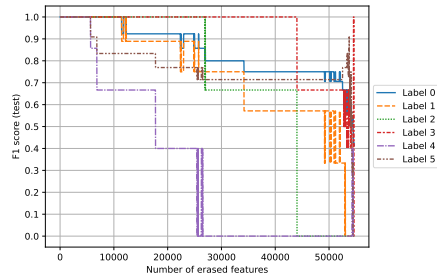
Figure 13: **Impact of feature erasure based on label 4 feature weights for Breast dataset.** The features were erased from the dataset following a ranking based on the feature weights given for label 4 by each algorithm. The horizontal axis shows the number of erased features, while the vertical axis displays the F1 score calculated from the test dataset.



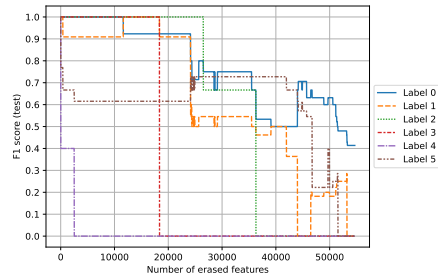
(a) Lasso



(b) LIME

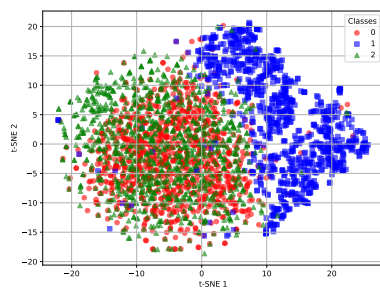


(c) DeepSHAP

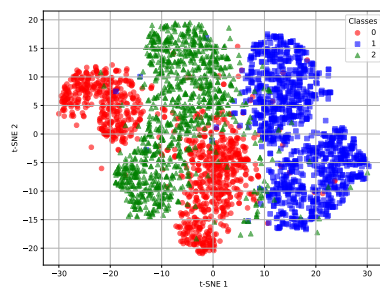


(d) Multiple FSL

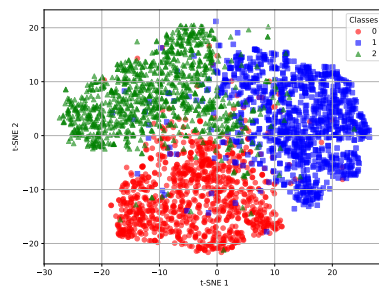
Figure 14: **Weighted tSNE applied to SynthB using label 1 weights from Multiple FSL, LassoNet and TabNet.** Label 1 samples are presented as blue squares.



(a) Multiple FSL



(b) LassoNet



(c) TabNet

Table 6: **Label-specific F1-score, mean $\pm$ standard deviation over 11 executions, for Letter Recognition test dataset.** In bold is the best average per label. Best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis and Dunn’s test are highlighted in orange.

Label	Baseline	Multiple FSL	TabNet	LassoNet	Lasso
A	0.989 $\pm$ 0.005	0.993 $\pm$ 0.003	0.941 $\pm$ 0.019	0.988 $\pm$ 0.005	0.603 $\pm$ 0.004
B	0.945 $\pm$ 0.006	0.940 $\pm$ 0.012	0.859 $\pm$ 0.025	0.928 $\pm$ 0.009	0.546 $\pm$ 0.011
C	0.970 $\pm$ 0.010	0.976 $\pm$ 0.005	0.908 $\pm$ 0.019	0.946 $\pm$ 0.010	0.492 $\pm$ 0.003
D	0.959 $\pm$ 0.011	0.957 $\pm$ 0.010	0.855 $\pm$ 0.027	0.935 $\pm$ 0.005	0.555 $\pm$ 0.007
E	0.953 $\pm$ 0.015	0.950 $\pm$ 0.009	0.885 $\pm$ 0.022	0.951 $\pm$ 0.006	0.085 $\pm$ 0.023
F	0.956 $\pm$ 0.008	0.952 $\pm$ 0.010	0.858 $\pm$ 0.024	0.920 $\pm$ 0.019	0.475 $\pm$ 0.015
G	0.955 $\pm$ 0.009	0.951 $\pm$ 0.012	0.849 $\pm$ 0.019	0.935 $\pm$ 0.011	0.020 $\pm$ 0.007
H	0.961 $\pm$ 0.007	0.952 $\pm$ 0.011	0.806 $\pm$ 0.035	0.916 $\pm$ 0.014	0.005 $\pm$ 0.008
I	0.958 $\pm$ 0.006	0.957 $\pm$ 0.004	0.923 $\pm$ 0.022	0.960 $\pm$ 0.011	0.748 $\pm$ 0.006
J	0.961 $\pm$ 0.007	0.959 $\pm$ 0.009	0.895 $\pm$ 0.025	0.972 $\pm$ 0.003	0.594 $\pm$ 0.006
K	0.972 $\pm$ 0.008	0.968 $\pm$ 0.013	0.889 $\pm$ 0.021	0.955 $\pm$ 0.015	0.092 $\pm$ 0.029
L	0.990 $\pm$ 0.006	0.994 $\pm$ 0.005	0.932 $\pm$ 0.014	0.970 $\pm$ 0.007	0.655 $\pm$ 0.005
M	0.984 $\pm$ 0.007	0.978 $\pm$ 0.006	0.934 $\pm$ 0.009	0.980 $\pm$ 0.006	0.498 $\pm$ 0.005
N	0.973 $\pm$ 0.009	0.978 $\pm$ 0.006	0.902 $\pm$ 0.022	0.940 $\pm$ 0.012	0.376 $\pm$ 0.010
O	0.965 $\pm$ 0.008	0.967 $\pm$ 0.011	0.871 $\pm$ 0.025	0.945 $\pm$ 0.008	0.053 $\pm$ 0.022
P	0.954 $\pm$ 0.009	0.955 $\pm$ 0.008	0.890 $\pm$ 0.020	0.947 $\pm$ 0.006	0.589 $\pm$ 0.010
Q	0.967 $\pm$ 0.008	0.967 $\pm$ 0.013	0.899 $\pm$ 0.021	0.957 $\pm$ 0.007	0.574 $\pm$ 0.012
R	0.950 $\pm$ 0.012	0.948 $\pm$ 0.013	0.857 $\pm$ 0.018	0.917 $\pm$ 0.015	0.658 $\pm$ 0.006
S	0.971 $\pm$ 0.006	0.979 $\pm$ 0.008	0.868 $\pm$ 0.032	0.963 $\pm$ 0.009	0.239 $\pm$ 0.017
T	0.980 $\pm$ 0.006	0.970 $\pm$ 0.007	0.903 $\pm$ 0.011	0.961 $\pm$ 0.007	0.667 $\pm$ 0.008
U	0.988 $\pm$ 0.006	0.979 $\pm$ 0.006	0.932 $\pm$ 0.018	0.980 $\pm$ 0.005	0.617 $\pm$ 0.011
V	0.965 $\pm$ 0.007	0.959 $\pm$ 0.012	0.928 $\pm$ 0.014	0.964 $\pm$ 0.008	0.613 $\pm$ 0.007
W	0.977 $\pm$ 0.010	0.976 $\pm$ 0.010	0.934 $\pm$ 0.014	0.968 $\pm$ 0.005	0.524 $\pm$ 0.004
X	0.977 $\pm$ 0.008	0.977 $\pm$ 0.006	0.913 $\pm$ 0.012	0.963 $\pm$ 0.005	0.071 $\pm$ 0.030
Y	0.975 $\pm$ 0.007	0.975 $\pm$ 0.010	0.922 $\pm$ 0.022	0.967 $\pm$ 0.007	0.445 $\pm$ 0.023
Z	0.982 $\pm$ 0.008	0.988 $\pm$ 0.006	0.961 $\pm$ 0.013	0.969 $\pm$ 0.004	0.556 $\pm$ 0.011

Table 7: **Label-specific F1-score, mean $\pm$ standard deviation over 11 executions, for Isolet test dataset.** In bold is the best average per label. Best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis and Dunn’s test are highlighted in orange.

Label	Baseline	Multiple FSL	TabNet	LassoNet	Lasso
A	0.953 $\pm$ 0.017	0.964 $\pm$ 0.017	0.974 $\pm$ 0.017	0.972 $\pm$ 0.012	0.752 $\pm$ 0.014
B	0.764 $\pm$ 0.046	0.869 $\pm$ 0.027	0.899 $\pm$ 0.030	0.865 $\pm$ 0.022	0.791 $\pm$ 0.017
C	0.982 $\pm$ 0.010	0.996 $\pm$ 0.006	0.982 $\pm$ 0.014	0.975 $\pm$ 0.008	0.882 $\pm$ 0.009
D	0.836 $\pm$ 0.056	0.905 $\pm$ 0.025	0.908 $\pm$ 0.023	0.899 $\pm$ 0.018	0.767 $\pm$ 0.012
E	0.871 $\pm$ 0.048	0.893 $\pm$ 0.033	0.918 $\pm$ 0.022	0.913 $\pm$ 0.020	0.862 $\pm$ 0.014
F	0.900 $\pm$ 0.037	0.916 $\pm$ 0.021	0.957 $\pm$ 0.035	0.969 $\pm$ 0.016	0.720 $\pm$ 0.020
G	0.901 $\pm$ 0.059	0.936 $\pm$ 0.038	0.959 $\pm$ 0.034	0.960 $\pm$ 0.012	0.720 $\pm$ 0.009
H	0.980 $\pm$ 0.021	0.989 $\pm$ 0.011	0.990 $\pm$ 0.017	0.996 $\pm$ 0.006	0.765 $\pm$ 0.021
I	0.961 $\pm$ 0.024	0.978 $\pm$ 0.014	0.990 $\pm$ 0.008	0.990 $\pm$ 0.008	0.917 $\pm$ 0.011
J	0.939 $\pm$ 0.033	0.979 $\pm$ 0.010	0.975 $\pm$ 0.015	0.973 $\pm$ 0.010	0.772 $\pm$ 0.016
K	0.945 $\pm$ 0.028	0.972 $\pm$ 0.012	0.983 $\pm$ 0.015	0.978 $\pm$ 0.017	0.686 $\pm$ 0.022
L	0.919 $\pm$ 0.011	0.955 $\pm$ 0.015	0.974 $\pm$ 0.011	0.951 $\pm$ 0.013	0.898 $\pm$ 0.000
M	0.793 $\pm$ 0.045	0.878 $\pm$ 0.032	0.917 $\pm$ 0.034	0.852 $\pm$ 0.027	0.697 $\pm$ 0.022
N	0.810 $\pm$ 0.039	0.895 $\pm$ 0.027	0.931 $\pm$ 0.030	0.876 $\pm$ 0.023	0.692 $\pm$ 0.021
O	0.909 $\pm$ 0.022	0.945 $\pm$ 0.019	0.967 $\pm$ 0.009	0.935 $\pm$ 0.015	0.874 $\pm$ 0.005
P	0.808 $\pm$ 0.044	0.870 $\pm$ 0.023	0.881 $\pm$ 0.036	0.872 $\pm$ 0.020	0.667 $\pm$ 0.023
Q	0.966 $\pm$ 0.010	0.969 $\pm$ 0.019	0.983 $\pm$ 0.014	0.959 $\pm$ 0.008	0.705 $\pm$ 0.012
R	0.967 $\pm$ 0.012	0.984 $\pm$ 0.008	0.986 $\pm$ 0.010	0.981 $\pm$ 0.007	0.928 $\pm$ 0.010
S	0.917 $\pm$ 0.050	0.938 $\pm$ 0.016	0.967 $\pm$ 0.030	0.979 $\pm$ 0.011	0.821 $\pm$ 0.005
T	0.809 $\pm$ 0.048	0.884 $\pm$ 0.023	0.908 $\pm$ 0.021	0.886 $\pm$ 0.024	0.620 $\pm$ 0.015
U	0.957 $\pm$ 0.022	0.969 $\pm$ 0.017	0.970 $\pm$ 0.016	0.960 $\pm$ 0.006	0.860 $\pm$ 0.008
V	0.800 $\pm$ 0.036	0.890 $\pm$ 0.035	0.896 $\pm$ 0.019	0.891 $\pm$ 0.025	0.718 $\pm$ 0.017
W	0.957 $\pm$ 0.018	0.966 $\pm$ 0.021	0.978 $\pm$ 0.016	0.965 $\pm$ 0.013	0.837 $\pm$ 0.008
X	0.952 $\pm$ 0.015	0.975 $\pm$ 0.007	0.974 $\pm$ 0.013	0.983 $\pm$ 0.008	0.794 $\pm$ 0.011
Y	0.986 $\pm$ 0.015	0.992 $\pm$ 0.007	0.997 $\pm$ 0.005	1.000 $\pm$ 0.000	0.976 $\pm$ 0.004
Z	0.945 $\pm$ 0.022	0.954 $\pm$ 0.013	0.950 $\pm$ 0.027	0.926 $\pm$ 0.014	0.766 $\pm$ 0.016

Selector	Granularity	Arch. Agnostic	Inference overhead	Training mechanism
<i>Embedded</i>				
Decision Tree	Global	No (is its own classifier)	N/A	Greedy recursive splitting
Random Forest	Global	No (is its own classifier)	N/A	Bagged greedy tree ensemble
Lasso	Label-specific	No (is its own linear classifier)	N/A	Coordinate descent (ℓ_1 -regularized)
LassoNet	Label-specific	No (fixed skip-connection MLP)	N/A	Backprop + proximal gradient (path-wise ℓ_1)
TabNet	Label-specific	No (own sequential-attention architecture)	N/A	Backprop (sequential attention masks)
<i>Filter</i>				
Kruskal-Wallis	Global	N/A (produces no classifier)	N/A	Closed-form statistical test per feature
ReliefF	Global	N/A (produces no classifier)	N/A	Instance-based nearest-neighbor heuristic
<i>Embedded — FSL family</i>				
FSL	Global	Partial [‡]	one extra elementwise multiply	Backprop, joint with classifier (ℓ_1 -regularized gate)
Listener	Label-specific	Partial [‡]	one extra elementwise multiply	Backprop; per-label signal from weight drift while training one label at a time
Residual	Label-specific	Partial [‡]	one extra linear residual term	Backprop, joint with classifier
Output Aware	Label-specific	Partial [‡]	one extra elementwise multiply [¶]	Backprop; gate conditioned on the one-hot label at train time
Multiple FSL	Label-specific	Partial [‡]	one batched matmul ($n_{\text{labels}} \times n_{\text{features}}$) ^{¶¶}	Backprop, joint with classifier
<i>Post-hoc</i>				
LIME	Label-specific	Yes (any black-box predict function)	None	Local surrogate linear regression on perturbed samples
DeepSHAP	Label-specific	Partial (restricted to differentiable/neural models)	None	Gradient-based SHAP approximation (DeepLIFT-style backprop)
SAGE	Label-specific	Yes (any predict(predict_proba model))	None	Marginal-imputation Shapley value estimation (KernelEstimator)

Table 8: **Comparison of feature selectors.** [‡]The FSL family’s gate is an input-masking layer applied before an arbitrary downstream differentiable classifier — the gating mechanism itself is architecture-agnostic in design. [§]Listener trains one shared weight vector but fits it one label at a time, tracking how much it moves during each label’s turn as that label’s attribution. [¶]Output Aware’s forward pass branches on train/eval mode: at training time the gate is selected via the batch’s one-hot labels; at evaluation/inference it always uses the sum of all label gates, applied as a single element wise multiply, unlike Multiple FSL which never needs a separate forward pass per label at inference time. ^{¶¶}This selector’s forward pass is mathematically identical to running the shared classifier once per label sequentially (same total FLOPs), but stacks every label’s input into one batched matrix multiplication instead, which GPUs parallelize far better. Post-hoc and filter methods incur no inference overhead since feature weighting is decoupled from the prediction model. Conversely, FSL and embedded methods integrate weighting directly into the inference process. While simple embedded models (e.g., Decision Trees, Random Forest, Lasso) have minimal architectural overhead, complex architectures like LassoNet and TabNet exhibit significantly higher processing times. For the FSL family of methods, we specify the computational impact of the feature weighting mechanism, as they are designed to be agnostic to their internal model.

Dataset	NeuralBaseClassifier	Decision Tree	Random Forest	Lasso	LassoNet	TabNet	FSL	Listener	Residual	Output Aware	Multiple FSL
XOR	0.078 ± 0.051	0.439 ± 0.071	0.081 ± 0.024	0.118 ± 0.028	0.003 ± 0.001	0.067 ± 0.028	0.003 ± 0.007	0.009 ± 0.007	0.069 ± 0.050	0.060 ± 0.042	0.003 ± 0.006
SynthA	0.086 ± 0.027	0.462 ± 0.023	0.276 ± 0.011	0.109 ± 0.006	0.044 ± 0.017	0.101 ± 0.025	0.073 ± 0.010	0.069 ± 0.010	0.062 ± 0.010	0.081 ± 0.037	0.062 ± 0.010
SynthB	0.090 ± 0.038	0.242 ± 0.016	0.310 ± 0.008	0.158 ± 0.005	0.087 ± 0.067	0.067 ± 0.018	0.037 ± 0.011	0.038 ± 0.005	0.025 ± 0.004	0.133 ± 0.031	0.029 ± 0.005
SynthC	0.076 ± 0.032	0.212 ± 0.016	0.297 ± 0.004	0.123 ± 0.007	0.041 ± 0.013	0.054 ± 0.020	0.086 ± 0.007	0.074 ± 0.011	0.084 ± 0.011	0.164 ± 0.029	0.066 ± 0.016
Liver	0.094 ± 0.142	0.045 ± 0.084	0.242 ± 0.016	0.117 ± 0.043	0.050 ± 0.074	0.111 ± 0.074	0.033 ± 0.080	0.072 ± 0.126	0.002 ± 0.007	0.070 ± 0.104	0.023 ± 0.053
Colorectal	0.172 ± 0.080	0.399 ± 0.108	0.252 ± 0.058	0.195 ± 0.059	0.190 ± 0.046	0.109 ± 0.091	0.129 ± 0.064	0.161 ± 0.060	0.218 ± 0.102	0.171 ± 0.070	0.136 ± 0.079
Breast	0.071 ± 0.032	0.173 ± 0.044	0.364 ± 0.018	0.227 ± 0.032	0.138 ± 0.044	0.117 ± 0.024	0.034 ± 0.022	0.027 ± 0.018	0.119 ± 0.053	0.054 ± 0.037	0.033 ± 0.030
Isotest	0.079 ± 0.019	0.199 ± 0.010	0.252 ± 0.003	0.417 ± 0.004	0.027 ± 0.007	0.023 ± 0.004	0.079 ± 0.014	0.130 ± 0.011	0.036 ± 0.008	0.308 ± 0.071	0.016 ± 0.005
Letter Recognition	0.010 ± 0.002	0.139 ± 0.004	0.149 ± 0.002	0.287 ± 0.003	0.011 ± 0.002	0.012 ± 0.003	0.011 ± 0.002	0.009 ± 0.002	0.009 ± 0.001	0.139 ± 0.095	0.011 ± 0.002

Table 9: **Expected Calibration Error (ECE) comparison across datasets.** Lower values indicate better calibration. The proposed Multiple FSL consistently achieves highly calibrated probabilities without the need for post-hoc adjustments. In bold is the lowest average. Best results with statistical significance (p ; 0.01) according to the Kruskal-Wallis and Dunn's test are highlighted in orange.

References

- [1] William H Kruskal and W Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952. URL <https://doi.org/10.1080/01621459.1952.10483441>.
- [2] Matheus Cezimbra Barbieri, Bruno Iochins Grisci, and Márcio Dorn. Analysis and comparison of feature selection methods towards performance and stability. *Expert Systems with Applications*, 249:123667, 2024. ISSN 0957-4174. URL <https://doi.org/10.1016/j.eswa.2024.123667>.
- [3] Bruno Iochins Grisci, Mathias J. Krause, and Marcio Dorn. Relevance aggregation for neural networks interpretability and knowledge discovery on tabular data. *Information Sciences*, 559:111–129, 2021. ISSN 0020-0255. URL <https://doi.org/10.1016/j.ins.2021.01.052>.
- [4] Suchi Vora and Hui Yang. A comprehensive study of eleven feature selection algorithms and their impact on text classification. In *2017 Computing Conference*, pages 440–449, 2017. URL <https://doi.org/10.1109/SAI.2017.8252136>.
- [5] Igor Kononenko. Estimating attributes: Analysis and extensions of relief. In Francesco Bergadano and Luc De Raedt, editors, *Machine Learning: ECML-94*, pages 171–182, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. ISBN 978-3-540-48365-6. URL https://doi.org/10.1007/3-540-57868-4_57.
- [6] Kenji Kira and Larry A. Rendell. A practical approach to feature selection. In Derek Sleeman and Peter Edwards, editors, *Machine Learning Proceedings 1992*, pages 249–256. Morgan Kaufmann, San Francisco (CA), 1992. ISBN 978-1-55860-247-2. URL <https://doi.org/10.1016/B978-1-55860-247-2.50037-1>.
- [7] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of relief and rrelief. *Machine learning*, 53:23–69, 2003. URL <https://doi.org/10.1023/A:1025667309714>.

- [8] Shan Suthaharan. Machine learning models and algorithms for big data classification. *Integr. Ser. Inf. Syst*, 36:1–12, 2016. URL <https://doi.org/10.1007/978-1-4899-7641-3>.
- [9] Anthony J Myles, Robert N Feudale, Yang Liu, Nathaniel A Woody, and Steven D Brown. An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 18(6):275–285, 2004. URL <https://doi.org/10.1002/cem.873>.
- [10] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001. URL <https://doi.org/10.1023/A:1010933404324>.
- [11] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 01 1996. ISSN 0035-9246. URL <https://doi.org/10.1111/j.2517-6161.1996.tb02080.x>.
- [12] Kai Sun, Shao-Hsuan Huang, David Shan-Hill Wong, and Shi-Shang Jang. Design and application of a variable selection method for multilayer perceptron neural network with lasso. *IEEE Transactions on Neural Networks and Learning Systems*, 28(6):1386–1396, 2017. URL <https://doi.org/10.1109/TNNLS.2016.2542866>.
- [13] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2660–2673, 2017. URL <https://doi.org/10.1109/TNNLS.2016.2599820>.
- [14] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL <https://jmlr.org/papers/v9/vandermaaten08a.html>.
- [15] Geoffrey E Hinton and Sam Roweis. Stochastic neighbor embedding. In S. Becker, S. Thrun, and K. Obermayer, editors, *Ad-*

- vances in Neural Information Processing Systems*, volume 15. MIT Press, 2002. URL https://proceedings.neurips.cc/paper_files/paper/2002/file/6150ccc6069bea6b5716254057a194ef-Paper.pdf.
- [16] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of “eckart-young” decomposition. *Psychometrika*, 35(3):283–319, 1970. URL <https://doi.org/10.1007/BF02310791>.
- [17] Wael Awada, Taghi M. Khoshgoftaar, David Dittman, Randall Wald, and Amri Napolitano. A review of the stability of feature selection techniques for bioinformatics data. In *2012 IEEE 13th International Conference on Information Reuse & Integration (IRI)*, pages 356–363, 2012. URL <https://doi.org/10.1109/IRI.2012.6303031>.
- [18] Zengyou He and Weichuan Yu. Stable feature selection for biomarker discovery. *Computational Biology and Chemistry*, 34(4):215–225, 2010. ISSN 1476-9271. URL <https://doi.org/10.1016/j.compbiolchem.2010.07.002>.