



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,*}

^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

HIGHLIGHTS

- We propose four label-specific feature weighting algorithms.
- We compiled evaluation methods for feature weighting in multi-class problems.
- We compare all extensions to identify the best performing one.
- We compare our proposal with state-of-the-art feature weighting algorithms.

ARTICLE INFO

Keywords:

Feature selection
Interpretable machine learning
Artificial neural networks
Dimensionality reduction
Gene expression data
Machine learning

ABSTRACT

In the age of data-driven decision-making, deep learning models have become powerful tools for uncovering patterns and making predictions that influence many aspects of society. However, their broader adoption in scientific and business applications is often limited by a lack of interpretability. Feature weighting methods, which assess the relevance of individual features, are among the most widely studied approaches for improving interpretability. While post hoc techniques have advanced significantly, especially in modeling nuanced relationships between features and labels in multiclass problems, embedded methods typically fall short in this regard. Embedded techniques integrate feature relevance learning directly into the model training process, but they are often unable to capture label-specific relationships. To address the limitation, we propose multiple extensions to this Feature Selection Layer, an embedded feature weighting mechanism for neural networks. Through extensive experimentation, we identify the most effective extension and benchmark it against established feature weighting methods. Our results show that the proposed method matches existing techniques in identifying globally relevant features and outperforms them in capturing label-specific relevance. By leveraging the internal dynamics of the trained neural network, the method reveals how individual features contribute to the prediction of specific labels, requiring no post-processing beyond the training process.

1. Introduction

Over the past decade, the exponential growth of data driven by the Internet has led to the successful application of deep neural networks across diverse fields, including computer vision, natural language processing, robotics, and healthcare [1]. However, state-of-the-art models often operate as “black boxes”, lacking interpretability. In critical domains like healthcare or human resources, high

* Corresponding author.

Email addresses: jpsilva@inf.ufrgs.br (J.P. Silveira e Silva), mdorn@inf.ufrgs.br (M. Dorn), bigrisci@inf.ufrgs.br (B. Iochins Grisci).

<https://doi.org/10.1016/j.ins.2026.124273>

Received 20 January 2026; Received in revised form 6 October 2026; Accepted 6 October 2026

Available online 10 October 2026

0020-0255/© 2026 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

The Article Processing Charge for the publication of this research was funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) (ROR identifier: 00x0ma614).

predictive accuracy alone is insufficient [2]. Hidden biases, such as a model inappropriately relying on demographic attributes like gender, can severely compromise a model's reliability and exacerbate social issues [3]. Furthermore, neural networks are prone to learning “shortcuts”, achieving high classification performance by exploiting data artifacts rather than capturing the true underlying relevance of the features [4]. Regulations like the EU General Data Protection Regulation (GDPR) increasingly mandate a “right to explanation” for automated decisions, making interpretability not just a scientific preference, but a legal necessity [5].

These challenges have catalyzed extensive research in Interpretable Machine Learning (IML). IML methods generally fall into two categories: local methods, which explain individual predictions, and global methods, which elucidate the overall behavior of the model across a dataset. Among global methods, feature weighting is a widely adopted approach that assigns importance scores to input features [6]. When applied to neural networks, feature weighting clarifies the decision-making process and can enhance generalization by mitigating the influence of noisy features. This is particularly relevant for tabular data. Unlike images or text, tabular data lacks spatial or sequential structure, which poses unique challenges for representation learning. Consequently, standard neural networks often underperform on these datasets due to their high sensitivity to uninformative attributes [7].

Problem: Formally, in a multi-class tabular dataset, a neural network learns a mapping from an input matrix $X \in \mathbb{R}^{N \times M}$ (N samples, M features) to a set of discrete labels $y \in \{1, \dots, L\}$ (L possible labels or classes). Most global feature weighting methods generate a single global feature relevance vector $w \in \mathbb{R}^M$, assigning one score per feature. However, in multi-class tasks, feature importance frequently varies across different labels; a feature highly discriminative for one label may act as noise for another [8]. An ideal approach should instead produce a label-specific relevance matrix $W \in \mathbb{R}^{M \times L}$, where $W_{m,l}$ indicates the relevance of a feature m specifically for a label l . A robust label-specific method must filter general noise, highlight unique label-defining features, maintain predictive performance, and provide granular interpretability without requiring multiple separate models.

Research gap: A prominent method to handle feature weighting within neural networks is the Feature Selection Layer (FSL) [9], an embedded technique that autonomously learns feature relevance from the model's internal dynamics during training. However, the current FSL architecture is fundamentally constrained by its design: it only learns a single general weight per feature (a global relevance vector). Consequently, it cannot discern whether a feature is universally important or only relevant to a specific subset of labels, limiting its utility in uncovering complex, label-dependent patterns.

Proposed method: To bridge this gap, we propose an architectural extension to the FSL that explicitly captures label-specific relevance. Instead of passing input features through a single one-dimensional weight vector, our extension embeds a two-dimensional matrix directly into the network. During training, the network dynamically adjusts these weights via gradient descent, learning to route and scale features differently depending on the specific label pathway. Consequently, the network simultaneously classifies the data and builds a transparent matrix mapping exactly which features drive the prediction of each specific label. Evaluations on both synthetic and real-world high-dimensional tabular datasets demonstrate that our approach performs on par with established techniques in identifying general features and significantly outperforms them in capturing label-specific relevance, all while maintaining or improving overall predictive performance.

This study addresses the research question: “Can an embedded FSL be capable of learning label-specific feature relevance?” To answer this, we address the following complementary questions:

1. “How to evaluate a label-specific feature weighting algorithm?” (Section 2)
2. “How to adapt FSL to learn label-specific feature relevance?” (Section 4)
3. “How does this approach compare to other algorithms?” (Section 5)

Guided by these research questions, and to provide a comprehensive roadmap of our proposed approach, the main contributions of this paper are:

- We introduce a novel label-specific extension to the Feature Selection Layer (FSL), expanding its learnable parameters to map features to specific labels dynamically during training.
- We consolidate and adapt evaluation methodologies to rigorously assess the quality of both general and label-specific feature weighting techniques.
- We validate the proposed approach on synthetic and real-world high-dimensional tabular data, demonstrating its efficacy against established state-of-the-art baselines.

The remainder of this work is structured as follows: Section 2 reviews interpretability and evaluation methodologies; Section 3 outlines related studies; Section 4 details multiple possible label-specific FSL extensions; Section 5 presents experiments and results; and Section 6 concludes the paper, discussing limitations and future work.

2. Methods and concepts

This section outlines the theoretical foundation of this work, covering interpretability (Section 2.1), feature weighting as a tool for interpretability (Section 2.2), and evaluation metrics for validating feature weighting techniques with an emphasis on multi-class problems (Section 2.3).

2.1. Interpretability

Interpretability is a broad and challenging concept to define. Murdoch et al. [10] describe interpretability in Machine Learning (ML) as the extraction of relevant knowledge from a model, relating to relationships that are either contained in the data or learned by the model. Interpretable Machine Learning (IML) methods can be used to discover knowledge, to debug or justify the model and its predictions, and to improve it [6]. IML methods can be local or global. Local methods explain individual predictions, such as Shapley values. Global methods, on the other hand, explain the overall model behavior and how it behaves on average for a given dataset. One popular type of global explanation is feature weighting, which assigns weights to features based on their relevance to the predictions [6]. In this context, Barbieri et al. [11] and Grisci et al. [8] highlight the importance of considering multi-class problems while ranking feature relevance. When dealing with different labels, the relevance of features can vary, and acknowledging these variations enhances model interpretability. Most state-of-the-art Machine Learning algorithms are “black-box” models, with complex structures that make their learned patterns difficult to explain. This lack of interpretability can lead to distrust and can conceal bias.

2.2. Feature weighting and selection

One effective approach to enhancing interpretability is feature weighting, which involves algorithms that assign continuous relevance scores to features based on their importance for predictions. Feature selection is a subset of this approach, using binary weights to retain or discard features [12]. Both are widely used in machine learning, pattern recognition, statistics, and data mining for identifying relevant features, improving accuracy, reducing computation, and enhancing interpretability [13]. Unlike feature extraction, these methods preserve the original input space, resulting in more interpretable models [11]. According to Miao and Niu [13], feature weighting algorithms are usually grouped into five categories: filter, wrapper, embedded, ensemble and hybrid. Another approach uses post-hoc techniques, which define feature weights using trained models by understanding the relevance that they give to each feature based on one or multiple instances. In this work, we choose five consolidated algorithms to evaluate our new approach: Kruskal–Wallis filter, ReliefF, Decision Trees, Random Forests and Least Absolute Shrinkage and Selection Operator (LASSO). Further details on these algorithms are provided in Section 1 of the **Supplementary Material**.

2.3. Feature weighting evaluation

To assess and compare feature weighting algorithms, we adapted the evaluation framework proposed by Barbieri et al. [11], originally designed for binary classification, to handle multi-class problems by including additional metrics. Our evaluation follows four complementary strategies.

(1) **Identification of informative features:** In real-world, high-dimensional datasets, the exact mapping of relevant features to specific labels is often unknown, making it difficult to objectively evaluate feature weighting algorithms. To overcome this, we utilize synthetic datasets where the ground-truth informative features for each label are explicitly defined. This precise knowledge allows us to explicitly track how effectively an algorithm recovers these relevant features while ignoring noise. To quantify this, we evaluate top- n feature subsets (ranked by their assigned weights) using two complementary metrics. **Percentage of Informative Features Selected (PIFS)** acts as a measure of recall (or sensitivity). It calculates the proportion of the total explicitly informative features that are successfully included in a given selected subset (see Eq. 7 in the **Supplementary Material**). A higher PIFS score indicates a better recovery of the underlying relevant features. A score of 1.0 signifies that all genuinely informative features for that label have been successfully identified and included in the subset. **Percentage of Selected Features that are Informative (PSFI)** acts as a measure of precision. It evaluates the purity of the selected subset by calculating the proportion of its features that are genuinely informative (see Equation 8 in the **Supplementary Material**). A higher PSFI score indicates that the algorithm is highly accurate in its weighting, assigning high ranks strictly to relevant features rather than noisy or irrelevant ones. These metrics are uniquely appropriate for evaluating label-specific feature selection because they can be calculated individually against the ground-truth feature set of each specific label. By calculating PIFS and PSFI across incrementally larger subset sizes, we can observe the exact threshold at which an algorithm stops selecting label-specific informative features and begins incorporating global noise.

(2) **Feature erasure and predictive power:** Following established perturbation analysis for tabular data [14], we evaluate feature relevance by masking specific inputs. This is achieved by setting feature values to zero after normalization, and measuring the impact on the model performance. The computational simplicity of this approach is critical, as feature erasure evaluation can be prohibitive for the high-dimensional datasets used in our experiments. Furthermore, we evaluate the predictive power of different feature subsets by training SVM models using the most relevant features identified.

(3) **Weighted t-SNE and Silhouette Coefficient [15]:** It is an extension of t-Distributed Stochastic Neighbor Embedding (t-SNE), a widely used technique for visualizing high-dimensional data in 2D. Weighted t-SNE modifies the standard approach by incorporating feature relevance scores (normalized between 0 and 1) into the Euclidean distance calculation. Consequently, features with higher relevance exert a greater influence on the spatial arrangement of points in the final visualization, while less relevant features contribute less to the distance between points. To quantify the cluster separation quality, the weighted t-SNE results are presented alongside the Silhouette Coefficient. This metric ranges from -1 to 1 . A value close to 1 indicates that a point is well-matched to its assigned cluster, while 0 implies the point is near a decision boundary and -1 suggests the point may be assigned to the wrong cluster. Compared to other visualization techniques, t-SNE has some important advantages that make it preferable for feature weighting, such as its ability to capture non-linear data relationships and its robust performance in cluster separation [16]. The primary goal of weighted t-SNE is to provide a global visual inspection tool for comparing the impact of different feature weighting techniques on data clustering, having already been applied in different works such as [8,17,18]. Therefore, the analysis is comparative. The visualization of the

weighted dataset should be contrasted with the original (unweighted) t-SNE projection. Regarding implementation, we adhere to the guidelines by Kobak and Berens [19] and Grisci et al. [15].

(4) **Algorithm stability:** Given the importance of reproducibility, particularly in fields such as Bioinformatics, we assess stability via two methods. First, we apply data perturbation using K-Fold Cross-Validation as proposed by Awada et al. [20]. We train each feature weighting algorithm with different fold combinations and compare the generated feature weights using the Jaccard index, Spearman's rank correlation, and Pearson correlation. Secondly, we employ the $m - 1$ approach, with m being the number of features. After initially training the algorithm, we modify the dataset by removing the top-ranked feature. We then execute the algorithm again and compare the newly generated weights against the original results using the same stability metrics. More details are provided in Section 2 of the **Supplementary Material**.

3. Related work

This section reviews key research on extracting feature weights, focusing on interpretability. The majority of existing methods capable of assigning label-specific feature weights rely on post-hoc analysis. LIME is a popular post-hoc, model-agnostic method for explaining individual predictions of black-box models [21]. It works by locally approximating the model around a specific input using a simple interpretable model. This involves perturbing the input features and observing the impact on the model's output, estimating feature relevance [22]. Though designed for local explanations, LIME can also support global feature weighting by aggregating feature weights from multiple representative inputs [23]. Another widely used post-hoc technique is SHAP, which encompasses multiple local approximation methods using Shapley values as a measure of feature relevance. Here, we focus on DeepSHAP. While the original SHAP is a model-agnostic method, DeepSHAP leverages the inherent structure of a neural network to enhance computational efficiency [24]. Relevance Aggregation is another post-hoc algorithm designed to improve the interpretability of neural networks trained on tabular data based on the Layer-wise Relevance Propagation (LRP) [25]. LRP was designed primarily for local interpretability on image datasets. Unlike images, where the order of pixels and their spatial relationships are crucial, tabular data features are often independent of their order. Additionally, the same information in an image, such as a pixel, can have varying meanings, a characteristic absent in tabular data. By leveraging this fixed representation, Relevance Aggregation combines the feature relevance weights computed from multiple samples, producing global label-specific feature weights [8]. Unlike other post-hoc techniques, FeatureX integrates model perturbation and feature correlation analysis to quantify feature importance [26]. Perturbation measures each feature's impact on performance, while correlation highlights associations and redundancies. Compared to LIME and SHAP, FeatureX achieved slight improvements in prediction and up to 9 times faster execution, mainly on complex models where SHAP's performance degrades. SAGE is a post-hoc, model-agnostic feature weighting algorithm designed to estimate global feature importance. It quantifies predictive power while accounting for complex feature interactions. Like SHAP, SAGE derives from game-theoretic Shapley values; however, it is specific to global rather than local feature importance. This approach makes SAGE orders of magnitude faster than naive SHAP. It also demonstrates superior performance compared to methods such as Feature Ablation, Mean Importance, and Permutation Tests [27]. Unlike post-hoc methods, embedding methods integrate feature weighting during training, improving computational efficiency. However, they typically lack the ability to define label-specific feature relevance. Being designed for high-dimensional biological data, FsNet is a nonlinear general feature selection network. It comprises a selection layer composed of concrete random variables, which are continuous relaxations of one-hot vectors. A reconstruction layer is also included to stabilize the training process. During training, FsNet selects features using the selection layer while simultaneously maximizing classification accuracy and minimizing reconstruction error [28]. By also incorporating a selection layer, Feature Selection Layer (FSL) [9], Algocf 1, is another embedded approach for general feature weighting which adds a layer without bias after the input layer where each neuron has a one-to-one correspondence with a feature. Unlike FsNet, FSL does not employ a reconstruction or any other additional layer, resulting in a simplified implementation with lower impact on model training. During prediction, before entering the network, each feature is multiplied by its corresponding weight which is trained jointly with the rest of the network's parameters and represents the feature relevance. FSL demonstrated superior performance when compared to other methods like Random Forest, ReliefF, Mutual Information and the Attention-based Feature Selection (AFS) [9]. AFS uses neural networks with attention modules to assign general feature weights by formulating feature-label correlations as binary classification problems supported by a shallow attention network for each feature. For small datasets, weight assignment may be limited, so a hybrid initialization using normalized weights from other methods was proposed by the authors [29]. LassoNet represents a distinct approach among embedded methods, capable of inferring label-specific feature importance. Unlike the standard Lasso, which is limited to linear models, LassoNet extends feature weighting to neural networks with multiple hidden layers. It achieves this by incorporating skip (residual) layers and enforcing a hierarchical constraint where a feature may only participate in any non-linear hidden layer if its skip-layer representative is active. Furthermore, the method employs a modified objective function to integrate feature selection directly into the training process [30]. Unlike a generic embedded method, TabNet is an interpretable deep learning architecture designed for tabular data that employs a sequential attention mechanism to select salient features at each decision step. By jointly performing feature selection and output mapping, the model provides both local and global interpretability regarding feature importance. Empirical evaluations demonstrate that TabNet yields superior predictive performance compared to established tree-based ensembles like XGBoost, LightGBM and CatBoost, while offering more robust feature weighting than post-hoc methods such as LIME and Integrated Gradients [31]. DOvE [32], an extension of One-versus-Each (OvE) methods, addresses the limitations of label-independent selection by employing a relevance matrix, named "three-layer label-specific classification", where classification models can be used to discriminate between different combinations of labels given the selected features to generate the final output. LSFSFI [33] addresses the limitations of global feature selection in multi-label datasets, where each sample can have multiple labels simultaneously. According to the authors most feature selection

Algorithm 1: Feature Selection Layer (FSL) training. A single feature weighting vector W is learned jointly with the downstream classifier f_ϕ : at each step, W is passed through a ReLU to produce a non-negative gate W^+ , which element-wise multiplies the input features before they reach the classifier. An ℓ_1 penalty on W , scaled by λ , drives the gate toward sparsity, while the classifier is trained to minimize a class-frequency-weighted cross-entropy loss. After training for E epochs, the final activated gate $W^+ = \text{ReLU}(W)$ is returned as the learned per-feature importance scores.

Input: Training set $D_{\text{train}} = \{(x_i, y_i)\}_{i=1}^n$, regularization strength λ , learning rate η , epochs E

Output: Learned feature weights $W^+ \in \mathbb{R}^d$

Initialize feature weighting vector parameters $W \leftarrow \mathbf{1}_d$;

Initialize classifier parameters ϕ randomly;

for $epoch = 1$ **to** E **do**

foreach $mini\text{-}batch (X, y) \in D_{\text{train}}$ **do**

$W^+ \leftarrow \text{ReLU}(W)$;

// non-negative feature gate

$X' \leftarrow X \odot W^+$;

// gate the input features

$\hat{y} \leftarrow f_\phi(X')$;

// shared classifier

$\mathcal{L} \leftarrow \text{CE}_{\text{balanced}}(\hat{y}, y) + \lambda \|W\|_1$;

// loss + regularization

 Update (W, ϕ) via backpropagation on $\mathcal{L}$ (AdamW, learning rate η);

return $W^+ = \text{ReLU}(W)$

methods used for multi-label datasets consider a single feature space for all labels, leading to suboptimal solutions since different labels can depend on different features. LSFSFI addresses this issue by using a partial fuzzy mutual implication granularity information to model feature-label dependencies, integrating this as prior knowledge into an embedded multi-label feature selection model.

4. Proposed extensions

As discussed in Section 2.2, feature weighting is essential for interpreting models and handling high-dimensional, low-sample datasets. While global feature weighting suits single-class tasks, it often fails in multi-class problems, especially under label imbalance, harming minority class performance. We propose four extensions to FSL that enable label-specific feature weighting. These enhance both multi-class accuracy and interpretability while preserving FSL's core properties by leveraging internal model dynamics. Although these are essentially feature weighting models, we chose to retain the name "Feature Selection Layer" to explicitly reference the adapted method.

4.1. Listener extension

We begin by introducing the Listener, with the objective of gaining deeper insights into the feature weight learning process within the original FSL implementation. Unlike the other extensions, which intervene directly in the forward pass to compute label-specific weights during training, Listener leaves the forward pass unchanged from that of the base FSL: a single shared weight vector w gates the input for every batch, regardless of label. Its label-specific signal is instead obtained passively, by observing how that shared vector moves while training is restricted to one label at a time. Each training step processes a batch containing samples from a single class. Between training forward passes, Listener records the weight vector's current value, and after the optimizer updates it, it accumulates the resulting shift into a per-label accumulator dedicated to that batch's class composed of a matrix $W_{L \times M}$, where L is the number of labels and M is the number of features (Fig. 1). Over the course of training, each label's accumulator stores the cumulative updates derived specifically from its own data. Rather than acting as an FSL extension, it functions as an observer, inferring a label's influence on a single, shared FSL by measuring how much it perturbs the model. While it remains relevant to understand how individual labels influence the FSL weights, it is crucial to distinguish this tool's specific role. It acts strictly as an analyzer of feature weight updates to the original FSL, rather than a true extension that modifies the underlying architecture, unlike the methods detailed in the following sections.

4.2. Residual extension

Residual learning addresses the vanishing gradient problem commonly encountered in deep neural networks, by incorporating "shortcut connections" between layers, which allow gradients to flow more easily through the network [34]. We extend the FSL by introducing a weight matrix $W_{L \times M}$, where L is the number of labels and M is the number of features (Fig. 2). Each row should learn the feature relevance for a single label. To enable the model to learn different feature weights for each label, we incorporate a shortcut connection that computes the sum of the FSL and the last hidden layer outputs to generate the model's output for each label. To ensure compatibility between the dimensions of the outputs, we employ a linear projection as specified by [34]. The residual connection adds, for each label, the label-specific weight vector W_l multiplied by the input X to the output for label l of the last hidden layer, where l is one of the L labels. This process influences the output values for each specific label and the backpropagation process as the gradients can flow directly back through the residual connection to the Feature Selection Layer. Since Residual exclusively records label-specific feature weights, general feature weights are determined by averaging the corresponding label-specific weights; the same applies to Output Aware (Section 4.3) and Multiple Feature Selection Layer (Section 4.4).

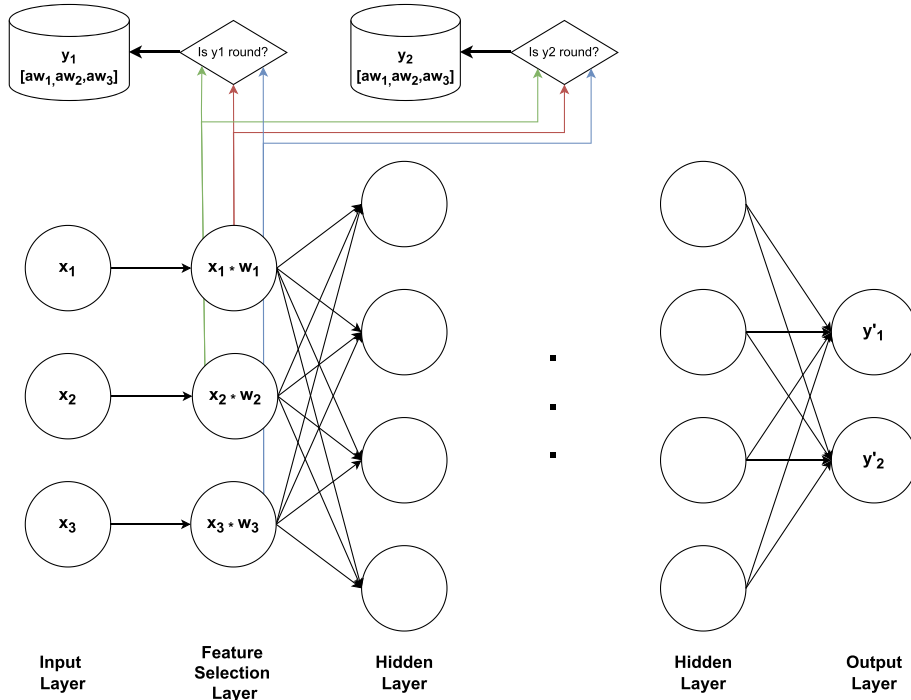


Fig. 1. Representation of Listener extension. In this example, two listeners, one per label, accumulate weight updates for each of the three features ($aw_1 \dots aw_3$) based on the label in the current batch. The FSL remains unchanged. For M features and L labels, $y_L[aw_1, \dots, aw_M]$ stores the accumulated weights per feature-label pair.

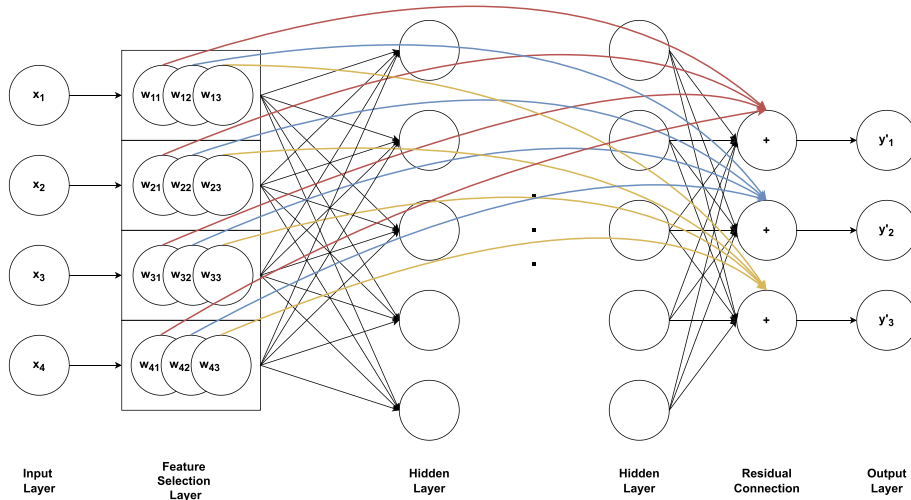


Fig. 2. Representation of Residual extension. In this example, three weights were added for each feature on FSL, one per label. The colored lines represent the connections from the feature weights, already multiplied by the feature value, to their specific label outputs.

4.3. Output aware extension

The proposed extension also leverages a feature weight matrix, $W_{L \times M}$ (Fig. 3), where each row represents the feature weights associated with a specific label. During training, the expected output y is utilized to dynamically select the row that corresponds to the respective label from $W_{L \times M}$ to be multiplied by the input before it is fed to the hidden layers. The weights are updated alongside all other network parameters during backpropagation. Given a batch of N training samples the process has the following steps: (1) The expected output vector $\vec{y}_N$ for the batch of N samples is converted into a one-hot encoded matrix $Y_{N \times L}$. Each row in $Y_{N \times L}$ represents a sample, with the value one in the column that corresponds to the expected label and the value zero elsewhere. (2) The one-hot encoded matrix $Y_{N \times L}$ is multiplied by the feature weight matrix $W_{L \times M}$ to produce a new matrix $V_{N \times M}$ (Eq. 1) which will be composed of N rows, one for each sample, and M columns with a single weight for each feature that corresponds to sample's

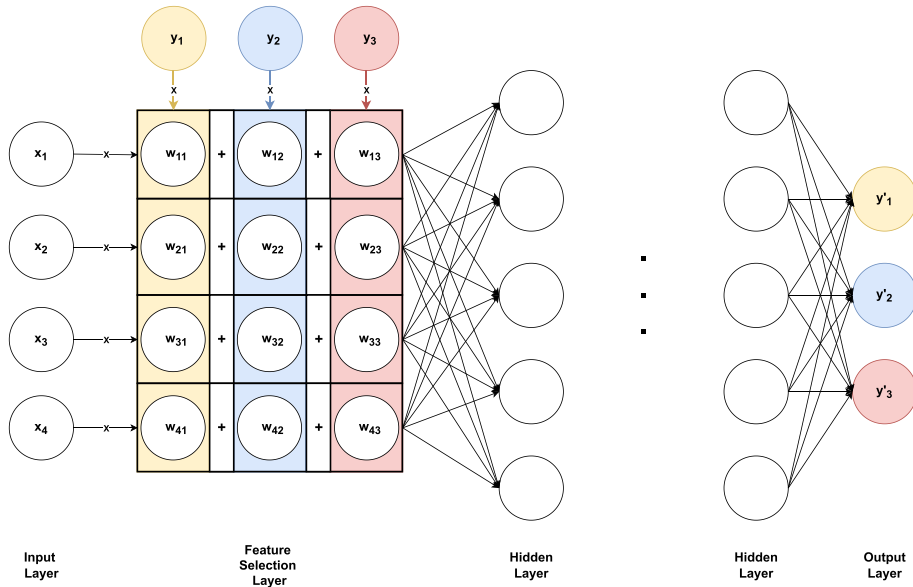


Fig. 3. Representation of Output Aware extension. FSL is implemented with multiple weight columns, each corresponding to a specific label (yellow, blue, and red). During training, a one-hot encoded vector (y_1, y_2, y_3) indicates the expected label, which will have value 1, while the others will be 0. Each one-hot encoded vector multiplies the weight matrix, effectively zeroing out weights not related to the current input's expected label. After being trained, the input features are multiplied by the sum of their feature weights.

label. (3) The matrix $V_{N \times M}$ is then multiplied element-wise with the input matrix $X_{N \times M}$, weighting the input features based on their relevance to the expected label. (4) After training, to keep using the feature relevance information for future predictions, all columns of the weight matrix $W_{L \times M}$ are summed together to form a single, general weight vector which will be used for predictions.

$$\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1L} \\ y_{21} & y_{22} & \cdots & y_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{NL} \end{bmatrix} \times \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1M} \\ w_{21} & w_{22} & \cdots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ w_{L1} & w_{L2} & \cdots & w_{LM} \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1M} \\ v_{21} & v_{22} & \cdots & v_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ v_{N1} & v_{N2} & \cdots & v_{NM} \end{bmatrix} \quad (1)$$

4.4. Multiple feature selection layer extension

The Multiple Feature Selection Layer (Multiple FSL) extends the base model with one dedicated FSL gate per label, while keeping the underlying hidden and output layers shared across every label (Fig. 4). Concretely, each label $l \in \{1, \dots, L\}$ owns a learnable weight vector $\theta_l \in \mathbb{R}^M$, collected into a matrix $\Theta = [\theta_1; \dots; \theta_L] \in \mathbb{R}^{L \times M}$, where M is the number of input features and L the number of labels. As defined in Section 4.5, a ReLU activation clips each weight at zero,

$$w_l = \text{ReLU}(\theta_l), \quad l = 1, \dots, L, \quad (2)$$

so that w_l acts as a non-negative, per-label feature gate: a feature k with $w_{l,k} = 0$ is entirely excluded from label l 's prediction, regardless of its relevance to any other label. Given an input $\mathbf{x} \in \mathbb{R}^M$, every gate is applied independently and propagated through the shared network $f: \mathbb{R}^M \rightarrow \mathbb{R}^L$ (the hidden and output layers common to every label):

$$\mathbf{z}_l = f(\mathbf{x} \odot w_l), \quad l = 1, \dots, L, \quad (3)$$

where $\odot$ denotes the element wise (Hadamard) product. Each $\mathbf{z}_l \in \mathbb{R}^L$ is a full output vector, but only its l -th entry (the one computed from label l 's own gated view of $\mathbf{x}$) is kept:

$$\hat{y}_l = [\mathbf{z}_l]_l = [f(\mathbf{x} \odot w_l)]_l, \quad l = 1, \dots, L. \quad (4)$$

The model is trained end-to-end with the classification loss (cross-entropy) plus an ℓ_1 penalty (Section 4.5) over Θ ,

$$\mathcal{L} = \mathcal{L}_{\text{CE}}(\hat{\mathbf{y}}, \mathbf{y}) + \lambda \|\Theta\|_1, \quad (5)$$

which drives each label's gate toward using only the features that remain informative for that label.

Because Eq. (3) evaluates f once per label, a single prediction requires L full propagations through the shared network instead of one, increasing both training and inference cost linearly in L . This sequential evaluation, Multiple FSL as defined above, is the less memory consuming implementation, since only one label's propagation is ever held in memory at a time, and is therefore useful when GPU memory is limited. However, all experiments reported in this article instead use a mathematically identical but substantially

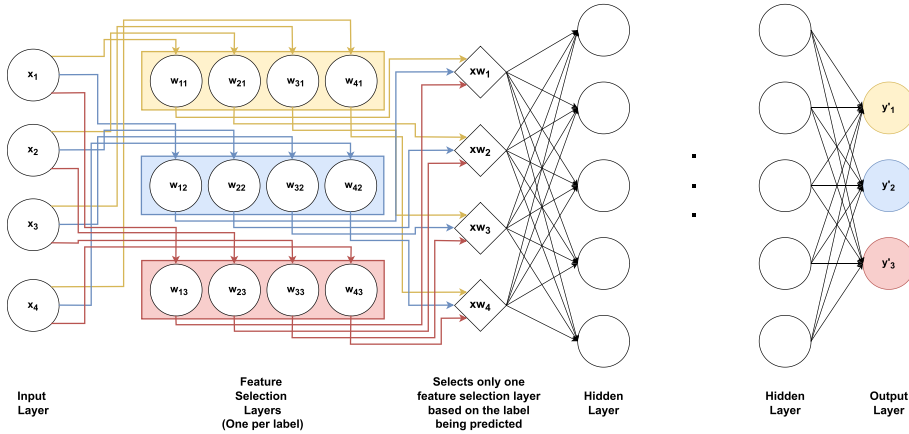


Fig. 4. Representation of Multiple feature selection Layer extension. Multiple selection layers are created, one for each label described by colors yellow and blue and red. The internal network is executed three times before backpropagation, one for each possible label. Only the feature weights related to the label being estimated are selected at each execution.

faster implementation which issues the L propagations as a single batched matrix operation rather than L sequential ones. Concretely, it stacks every label's gated view of $\mathbf{x}$ into one matrix,

$$\tilde{\mathbf{X}} = (\mathbf{1}_L \mathbf{x}^T) \odot \mathbf{W} \in \mathbb{R}^{L \times M}, \quad \mathbf{W} = \begin{bmatrix} w_1^T \\ \vdots \\ w_L^T \end{bmatrix} \in \mathbb{R}^{L \times M}, \quad (6)$$

where $\mathbf{1}_L \in \mathbb{R}^L$ is the all-ones vector, so that row l of $\tilde{\mathbf{X}}$ is exactly $\mathbf{x} \odot w_l$. The shared network is then evaluated once, applied row-wise to $\tilde{\mathbf{X}}$,

$$\mathbf{Z} = f(\tilde{\mathbf{X}}) \in \mathbb{R}^{L \times L} \quad (7)$$

and the final prediction is recovered by reading off the diagonal of $\mathbf{Z}$, which is identical to Eq. (4). Eq. (7) performs the same L forward passes as the sequential formulation, but as one matrix multiplication over L rows instead of L separate calls to f , which GPUs parallelize far more efficiently, at the cost of holding all L rows of $\tilde{\mathbf{X}}$ (and their intermediate activations) in memory simultaneously.

Unlike Output Aware, once training converges the weight gates $w_1, \dots, w_L$ are fixed and reused unchanged for every subsequent prediction. The predicted class is therefore never computed from a single, generic representation of $\mathbf{x}$, but from L label-specific ones, each shaped by the feature weighting that training assigned to that label. This holds regardless of which class ends up being predicted: every candidate label's score is produced using its own dedicated, already-learned gate, so the label-specific feature weighting learned during training carries over intact into inference, for every class, on every sample.

4.5. Regularization, activation functions and initial weights

The original FSL incorporates an adapted l_1 regularization term exclusively for its weights. In this work, we employ the default l_1 regularization without adaptations. This encourages sparsity and causes feature weights to become zero or near-zero, effectively eliminating irrelevant features [35]. Table 1 details a study evaluating Multiple FSL on datasets with a wide range of scales (500–7,797 samples, 50–54,677 features, 2–26 classes) across various regularization strengths. The range $\lambda \in \{10^{-3}, 10^{-2}\}$ was the only one statistically tied for the best test F1-score on every dataset; all other values suffered significant performance drops on at least one dataset. The extremes proved uniformly detrimental for different reasons. Without regularization ($\lambda = 0$), the absence of selection pressure caused sharp F1-score drops on XOR (0.53 vs. 1.00) and SynthA (0.70 vs. 0.93). Conversely, excessive regularization ($\lambda = 1$) collapsed the weight vector toward a constant, degrading performance on SynthA (0.93 to 0.75) and Isolet (0.95 to 0.80). Additionally, Fig. 5 highlights a tradeoff between predictive accuracy and feature-space separability. On SynthA, the highest silhouette score is achieved at $\lambda = 10^{-3}$, while the highest F1-score is achieved at $\lambda = 10^{-1}$. Isolet, on the other hand, exhibits greater agreement between the test F1 and 2D silhouette scores, which both reach their peak values between $\lambda = 10^{-4}$ and $\lambda = 10^{-2}$. On XOR, both curves rise together until $\lambda = 10^{-2}$, where the F1-score saturates at 1.00, whereas the silhouette score continues to increase at $\lambda = 10^{-1}$ before decreasing at $\lambda = 1$. For the Breast dataset, the metrics track each other closely, with the exception of a significant divergence at $\lambda = 1$. Ultimately, while there is no universally optimal value, these findings support $\lambda \in \{10^{-3}, 10^{-2}\}$ as the recommended default. For the activation function we adopted ReLU, which sets negative values to zero, thereby setting all activated feature weights to zero or positive. In our experiments, this configuration yielded superior performance. However, as can be noticed in Table 2 other activations such as Tanh can achieve similar results; for this reason the activation function remains modular allowing different configurations. For instance, allowing negative weights can be valuable for analyzing negative contributions. In our experiments, we observed that an initial weight of 1 consistently achieved an equal or better prediction score and feature weighting than the original $1/m$ initialization

Table 1

Test F1-score, mean $\pm$ standard deviation over 5 executions, for Multiple FSL across varying regularization strengths λ ; in bold is the best average per dataset. Best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis test and Dunn's test are highlighted in Orange.

λ	XOR	SynthA	Isolet	Breast
0.0000	0.526 $\pm$ 0.160	0.704 $\pm$ 0.019	0.943 $\pm$ 0.007	0.961 $\pm$ 0.022
0.0001	0.588 $\pm$ 0.194	0.732 $\pm$ 0.053	0.947 $\pm$ 0.004	0.981 $\pm$ 0.026
0.0010	0.741 $\pm$ 0.268	0.894 $\pm$ 0.010	0.946 $\pm$ 0.004	0.961 $\pm$ 0.022
0.0100	1.000 $\pm$ 0.000	0.918 $\pm$ 0.008	0.940 $\pm$ 0.005	0.942 $\pm$ 0.021
0.1000	1.000 $\pm$ 0.000	0.926 $\pm$ 0.007	0.907 $\pm$ 0.007	0.902 $\pm$ 0.070
1.0000	1.000 $\pm$ 0.000	0.746 $\pm$ 0.032	0.800 $\pm$ 0.017	0.971 $\pm$ 0.026

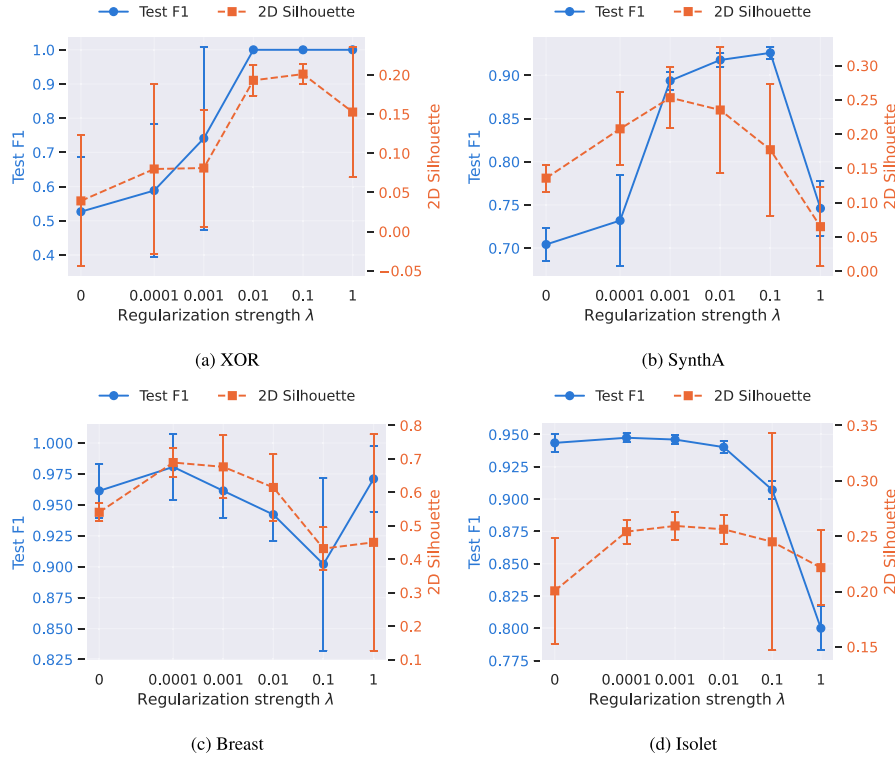


Fig. 5. Effect of regularization strength. test F1 (left axis, solid) and high-dimensional silhouette score (right axis, dashed) as a function of regularization strength λ , per dataset (mean $\pm$ std over 5 executions). Where the two curves diverge, the λ that maximizes classification accuracy is not the λ that maximizes feature-space separability.

used by FSL, where m is the number of features (as shown in Table 3). Consequently, we adopt a fixed initial weight of 1 for our experiments.

5. Experiments and results

This section presents experiments to evaluate the proposed FSL extensions compared with existing feature weighting algorithms. We start by introducing the utilized datasets (Section 5.1). Then we compare the extensions with the original FSL implementation, evaluate their capacity to select label-specific feature weights (Section 5.2), and select the best-performing one (Section 5.3). Next, we compare it with consolidated and established feature weighting techniques (Section 5.4). Finally, we discuss the obtained results (Section 5.5). All experiments were conducted using Python on a workstation equipped with an Intel Core i5-14600 K CPU, 128 GB of RAM, and an NVIDIA GeForce RTX 5070 Ti GPU with 16 GB of VRAM. Decision Tree, Random Forest and Lasso were implemented and optimized using the Scikit-Learn library. Kruskal Wallis and ReliefF implementations were based on Barbieri et al. [11]'s framework. LIME, DeepSHAP, LassoNet, TabNet and SAGE were implemented with their respective authors' libraries and recommendations, being optimized for each dataset. Further details on the classification and mechanics of each feature weighting method are available in Table 8 of the **Supplementary Material**.

Weighted t-SNE evaluations were performed following the author's recommended parameters. Neural network models were trained employing AdamW optimizer and label-weighted cross-entropy loss. For all neural networks, the same model architecture was used, differing only in the inclusion of the feature weighting algorithm. Input data were normalized when needed. For all experiments,

Table 2

Effect of the activation function on Multiple FSL across datasets calculated over 5 executions. In bold is the best average per dataset and metric, among activation functions. Best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis and Dunn's tests are highlighted in Orange.

Dataset	Activation	Test F1	Test Loss	2D Silhouette
XOR	ReLU	1.0000 $\pm$ 0.0000	0.0004 $\pm$ 0.0007	0.1931 $\pm$ 0.0195
	Sigmoid	0.4235 $\pm$ 0.0746	0.7011 $\pm$ 0.0083	0.0028 $\pm$ 0.0024
	Tanh	0.9980 $\pm$ 0.0045	0.0029 $\pm$ 0.0061	0.1974 $\pm$ 0.0177
SynthA	ReLU	0.9168 $\pm$ 0.0111	0.4945 $\pm$ 0.0733	0.1656 $\pm$ 0.0240
	Sigmoid	0.7090 $\pm$ 0.0119	0.7701 $\pm$ 0.0168	0.1472 $\pm$ 0.0178
	Tanh	0.9187 $\pm$ 0.0097	0.5199 $\pm$ 0.0428	0.1971 $\pm$ 0.0693
Isolet	ReLU	0.9461 $\pm$ 0.0038	0.1956 $\pm$ 0.0124	0.2592 $\pm$ 0.0127
	Sigmoid	0.9197 $\pm$ 0.0039	0.3021 $\pm$ 0.0154	-0.1497 $\pm$ 0.0275
	Tanh	0.9494 $\pm$ 0.0061	0.1885 $\pm$ 0.0153	0.2534 $\pm$ 0.0140
Breast	ReLU	0.9807 $\pm$ 0.0265	0.0948 $\pm$ 0.1350	0.6937 $\pm$ 0.0494
	Sigmoid	0.9613 $\pm$ 0.0216	0.3730 $\pm$ 0.3584	0.5106 $\pm$ 0.0614
	Tanh	0.9710 $\pm$ 0.0265	0.1326 $\pm$ 0.1241	0.7137 $\pm$ 0.0423

Table 3

Comparison of Multiple FSL with weight initialization of 1 and $1/m$ (m is the number of input features) across four synthetic datasets and the real world Breast dataset. In bold is the best average for each number of features. Best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis and Dunn's tests are highlighted in Orange.

Features	Init. Weight	Test F1	Test Loss	2D Silhouette
50	1	1.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.648 $\pm$ 0.021
	$1/m$	0.667 $\pm$ 0.456	0.440 $\pm$ 0.602	0.682 $\pm$ 0.027
500	1	1.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.725 $\pm$ 0.008
	$1/m$	0.833 $\pm$ 0.373	0.220 $\pm$ 0.493	0.726 $\pm$ 0.007
5000	1	0.998 $\pm$ 0.001	0.003 $\pm$ 0.002	0.600 $\pm$ 0.038
	$1/m$	0.997 $\pm$ 0.002	0.016 $\pm$ 0.013	0.601 $\pm$ 0.072
50,000	1	0.976 $\pm$ 0.018	0.086 $\pm$ 0.038	0.350 $\pm$ 0.216
	$1/m$	0.946 $\pm$ 0.013	0.162 $\pm$ 0.057	0.099 $\pm$ 0.076
54,675 (Breast)	1	0.981 $\pm$ 0.026	0.095 $\pm$ 0.135	0.694 $\pm$ 0.049
	$1/m$	0.942 $\pm$ 0.021	0.187 $\pm$ 0.063	0.648 $\pm$ 0.041

we ensured a separation between learning and evaluation. Feature weights were computed using only the training subsets, while the test subsets, comprising 20% of synthetic data and 15% of real-world data, remained unseen. This procedure was applied across all methods ensuring the use of the same data partitions. More details about the experiments parameters are provided in Section 3 of the **Supplementary Material**.

5.1. Datasets

To evaluate and compare the proposed FSL extensions against existing feature weighting algorithms, we utilized both synthetic and real-world datasets. While the final objective of any feature weighting method is to demonstrate effectiveness on real-world datasets, validation on synthetic data is a common practice in this field [36] and serves as a critical first step. It enables controlled experiments where experimental conditions, such as the proportion of irrelevant features, can be systematically varied, and provides ground truth knowledge of the optimal feature set, allowing for a rigorous and confident assessment of the method's accuracy [37]. In this work we propose four synthetic datasets. The first is a binary XOR dataset, with only two relevant features. It tests the algorithm's ability to capture strong non-linear feature interactions. To evaluate performance on multi-class problems, we generate three synthetic datasets (Table 4) using the algorithm by Guyon [38] adapted by Scikit-Learn. Each of the three datasets comprises 30 relevant features. SynthA provides a common set of relevant features for all labels. SynthB utilizes a label-specific feature approach, assigning 10 different features for each label. SynthC incorporates both methodologies, assigning 15 relevant features per label. In addition to these synthetic datasets, we also generated simpler datasets using the algorithm proposed by Guyon [38], varying only the number of features and labels for ablation studies (Tables 1, 3, 7 and 8). For real-world evaluation, we use three microarray datasets from CuMiDa [39], one binary and two multi-class, characterized by high dimensionality and limited sample sizes, and two multi-class benchmark datasets from UCI Machine Learning Repository, Letter Recognition [40] and Isolet [41], with larger sample sizes and low-to-moderate dimensionality (Table 5). For simplicity, labels were replaced with numerical identifiers. The default t-SNE results for each dataset are provided in Figs. 1 and 2 of the **Supplementary Material**.

Table 4

Synthetic datasets with defined number of relevant features (Relevant) and relevant features per label (Per label).

Name	Feature Type	Labels	Features	Relevant	Per label	Samples
XOR	Binary	2	50	2	2	500
SynthA	Numerical	3	100	30	30	3000
SynthB	Numerical	3	300	30	10	3000
SynthC	Numerical	3	300	30	15	3000

Table 5

Real-world datasets selected from CuMiDa and the UCI Machine learning Repository. Liver, Isolet (298–300 samples per class), and Letter Recognition (734–813 samples per class) are class-balanced, whereas Colorectal and Breast exhibit greater imbalance across classes.

Dataset	Features	Labels	Identifier	Samples
Liver 22,405	22,284	normal	0	24
		heptocarcinoma	1	24
Colorectal 41,657	33,468	normal	0	12
		high_grade_dysplasia	1	28
		low_grade_dysplasia	2	21
		adenocarcinoma	3	25
Breast 45,827	54,676	Basal	0	41
		HER	1	30
		cell_line	2	14
		normal	3	7
		luminal_A	4	29
		luminal_B	5	30
Isolet	617	Letters from A to Z	0–25	7797
Letter Recognition	16	Letters from A to Z	0–25	20,000

5.2. Comparative analysis between proposed extensions

This section presents a series of experiments to evaluate the proposed extensions and the original FSL implementation. To validate an ideal FSL extension, we aim to answer two key questions: (1) Do the general feature weights contribute meaningfully to model predictions as in the original FSL implementation? (2) Do the label-specific feature weights mostly impact individual label predictions?

We start by evaluating the feature weighting capabilities of each extension on the XOR dataset. Since the FSL is integrated and trained alongside the neural network, its capacity to learn relevant features is expected to be directly correlated with the predictive performance. In our tests all algorithms consistently achieved the highest performance on the XOR dataset, achieving precision, recall, and F1 scores of 1.0 on test data. When analyzing the Percentage of Informative Features Selected (PIFS) and the Percentage of Selected Features that are Informative (PSFI), both metrics reached a perfect score of 1 across all algorithms and executions when considering only the two highest-weighted features. This means that in the XOR dataset, all methods consistently prioritized the two key features. Table 6 presents the Pearson and Spearman scores computed over all features. It also reports the Jaccard score for the top–2 features, a threshold chosen to match the number of informative features in the dataset. All algorithms achieved the highest correlation value with Jaccard, confirming that all of them were able to assign the highest weights to the two relevant features in all 11 executions consistently. The observed values near 1 for Pearson mean a high correlation of the feature weights given to all features, while the values near 0 for Spearman mean a low correlation between the rankings generated from the feature weights. Apart from the relevant features, almost all the other features had their weights close to zero, which explains the high correlation values given by Pearson. Additional experiments involving the XOR datasets are provided in Figs. 3–6 of the **Supplementary Material**.

All extensions introduce additional processing time and memory overhead during training to obtain deeper insights into feature relevance. As shown in Table 7, which details an experiment using synthetic datasets with varying numbers of features and labels, the Listener method generally exhibited the highest average execution time, followed by Output Aware. FSL, Residual, and Multiple FSL achieved the best results, with a slight advantage for FSL when considering the overall averages. Regarding memory usage on the same datasets (Table 8), all models showed an expected increase in peak GPU memory as the number of features grew, since the size of the feature weight vector scales accordingly. Multiple FSL also demonstrated a linear increase in memory usage relative to the number of labels. This occurs because the method processes all labels' feature weight vectors simultaneously in a single combined batch. While this memory footprint could be reduced by processing each label sequentially, doing so would negatively impact execution time. Notably, we generated specific synthetic datasets with varying numbers of features and labels for this experiment and deliberately employed a lightweight classifier architecture, since maximizing prediction scores was not the primary objective. This setup ensures that the observed memory growth predominantly reflects the overhead of the feature selection layer itself.

In addition to preserving the FSL property of generating non-label specific feature weights, the proposed extensions should be capable of quantifying label-specific relevance. We utilize the synthetic multi-class datasets SynthA, SynthB, and SynthC in this

Table 6

Stability comparison between algorithms' feature weights on XOR dataset. Jaccard similarity measures set overlap, ranging from 0 (no overlap) to 1 (identical sets), and is used to assess the similarity between the top-n highest-weighted features. Spearman assesses 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. In bold are the best values for each metric.

Algorithm	Stability metric	Features	Score
FSL	Pearson	50	.990
	Spearman	50	.098
	Jaccard	2	1
Listener	Pearson	50	.987
	Spearman	50	.150
	Jaccard	2	1
Residual	Pearson	50	.994
	Spearman	50	.106
	Jaccard	2	1
Output Aware	Pearson	50	.981
	Spearman	50	.145
	Jaccard	2	1
Multiple FSL	Pearson	50	.974
	Spearman	50	.103
	Jaccard	2	1

Table 7

Execution time (s) of each proposed extension across varying numbers of features and labels. Values represent the mean $\pm$ standard deviation over 5 executions. In bold is the best average per dataset. Best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis and Dunn's tests are highlighted in Orange.

Features	Labels	FSL	Listener	Output Aware	Residual	Multiple FSL
50	2	15.90 $\pm$ 1.50	21.32 $\pm$ 1.09	19.95 $\pm$ 1.43	16.54 $\pm$ 1.29	16.44 $\pm$ 1.47
	5	14.60 $\pm$ 0.42	19.37 $\pm$ 0.80	17.60 $\pm$ 0.51	15.88 $\pm$ 0.38	15.64 $\pm$ 0.44
	10	17.03 $\pm$ 1.57	27.15 $\pm$ 1.73	24.62 $\pm$ 1.95	18.25 $\pm$ 1.79	17.95 $\pm$ 1.76
	20	15.30 $\pm$ 1.20	26.43 $\pm$ 2.71	24.37 $\pm$ 2.84	17.08 $\pm$ 1.60	16.62 $\pm$ 1.44
500	2	17.83 $\pm$ 1.44	39.24 $\pm$ 2.33	35.86 $\pm$ 2.13	18.73 $\pm$ 1.29	18.38 $\pm$ 1.15
	5	15.17 $\pm$ 2.08	33.12 $\pm$ 3.42	29.87 $\pm$ 3.78	14.94 $\pm$ 2.04	14.66 $\pm$ 2.42
	10	12.14 $\pm$ 0.19	31.47 $\pm$ 0.67	27.17 $\pm$ 0.43	12.82 $\pm$ 0.29	12.62 $\pm$ 0.10
	20	12.09 $\pm$ 0.51	32.56 $\pm$ 0.45	27.78 $\pm$ 0.60	12.56 $\pm$ 0.35	12.11 $\pm$ 0.48
5000	2	12.91 $\pm$ 2.00	126.38 $\pm$ 7.54	121.74 $\pm$ 4.47	13.26 $\pm$ 2.43	13.06 $\pm$ 2.06
	5	12.40 $\pm$ 0.11	121.33 $\pm$ 3.78	118.62 $\pm$ 3.39	13.14 $\pm$ 0.52	13.24 $\pm$ 0.48
	10	11.69 $\pm$ 1.19	124.43 $\pm$ 5.29	120.03 $\pm$ 7.36	12.20 $\pm$ 1.80	12.02 $\pm$ 1.66
	20	9.89 $\pm$ 0.30	118.86 $\pm$ 1.10	118.73 $\pm$ 0.99	10.61 $\pm$ 0.42	10.84 $\pm$ 0.42

analysis. As previously discussed, the effectiveness of feature weighting is closely tied to the model's predictive capacity. Models that fail to generalize their classification to new samples often exploit patterns found in noisy features, leading the FSL to assign higher feature weights to them. The proposed extensions should therefore have a neutral or positive impact on the model's predictive capacity. Table 9 compares the prediction performance using F1 score on SynthA, SynthB and SynthC datasets. In contrast to the strong performance of Multiple FSL, the Output Aware extension suffered a substantial performance degradation on SynthA and SynthB datasets. On SynthB, Multiple FSL, FSL and Listener consistently achieved the highest and most balanced scores across all labels, while Residual demonstrated a high degree of degradation. For SynthC, Multiple FSL achieved the highest scores, followed by Output Aware, while Residual performed worst. In general, Output Aware exhibited inconsistent performance, with better results on SynthC. Its reliance on output knowledge during training negatively affected its generalization ability. Residual consistently underperformed for all datasets. Multiple FSL, Listener, and the original FSL consistently performed well.

Given their strong prediction performance, it's expected that these models were able to identify relevant features. Fig. 6 shows the PIFS and PSFI average scores for different feature subsets based on feature weights generated through 11 executions. As previously discussed, all methods were able to consistently detect the two relevant features for the XOR dataset. For SynthA, Output Aware was the only algorithm that did not assign the highest weights for all the 30 relevant features in all executions. For SynthB and SynthC, no method consistently achieved the 30 relevant features in the top -30 highest weights.

Table 8

Peak GPU memory usage (MB) of each proposed extension across varying numbers of features and labels. The standard deviation is omitted because the variation in peak GPU allocation is minimal; given the same architecture and dataset, the memory footprint remains nearly identical across all executions. All peak GPU memory allocations were measured using an NVIDIA GeForce RTX 5070 Ti (16 GB VRAM). In bold is the best average per dataset and metric.

Features	Labels	FSL	Listener	Output Aware	Residual	Multiple FSL
50	2	18.1	18.1	18.1	18.1	18.4
	5	18.1	18.2	18.2	18.1	19.0
	10	18.1	18.2	18.2	18.2	20.2
	20	18.1	18.2	18.2	18.2	22.4
500	2	24.8	25.7	25.7	24.8	26.1
	5	24.8	25.9	25.9	24.9	29.9
	10	24.8	25.8	25.9	24.9	36.2
	20	24.8	25.9	26.1	25.1	48.9
5000	2	89.3	113.6	113.6	89.5	101.0
	5	89.3	113.0	113.4	89.9	136.9
	10	89.3	112.5	113.3	90.5	194.5
	20	89.3	113.4	115.2	91.9	311.6

Table 9

Per label F1 scores of FSL and proposed extensions on the SynthA, SynthB, and SynthC datasets. The scores were calculated across 11 independent executions using test dataset. For each reported metric we present the average and standard deviation. In bold is the highest average of each dataset and label. The best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis and Dunn's test are highlighted in Orange.

Algorithm	Label	SynthA	SynthB	SynthC
FSL	0	.887±.015	.934±.013	.878±.012
	1	.885±.013	.942±.011	.863±.016
	2	.888±.014	.924±.013	.865±.012
Listener	0	.888±.007	.945±.011	.890±.016
	1	.883±.009	.946±.021	.867±.018
	2	.889±.009	.934±.013	.880±.016
Residual	0	.845±.017	.811±.028	.796±.012
	1	.846±.014	.833±.065	.765±.012
	2	.864±.017	.783±.049	.767±.009
Output Aware	0	.742±.103	.774±.054	.919±.015
	1	.689±.142	.762±.066	.905±.015
	2	.740±.070	.657±.259	.911±.014
Multiple FSL	0	.915±.011	.943±.007	.950±.010
	1	.911±.007	.946±.012	.944±.011
	2	.916±.011	.933±.011	.947±.005

The feature weights produced by each extension for SynthB (Fig. 9 of **Supplementary Material**) show that Multiple FSL and Output Aware correctly assign the highest weights to label-relevant features. In contrast, Listener assigns high weights to all relevant features regardless of their specific label association, failing to accurately prioritize features based on label specificity. Residual exhibits an inconsistent behavior, while it assigns the highest weights to the relevant label in some cases; it conversely assigns the lowest weights to the correct label in others.

Fig. 7 illustrates the effect of applying feature erasure based on SynthC using label 0 feature weights to evaluate the effectiveness of the extensions in detecting label-specific relevant features that are mostly impacting individual labels' predictions. Since the original FSL implementation lacks the ability to identify label-specific feature weights it is not included in this analysis. The observed higher drop in F1 scores for label 0 suggests that the removed features play a crucial role in correctly classifying samples belonging to this label on the trained model for all proposed extensions. However, Output Aware and Multiple FSL demonstrate a higher impact on their label 0 predictions when compared to the other labels.

Fig. 8 displays the positions of each feature, represented by colored lines, across 11 different executions of each method on SynthC dataset specifically related to label 0. The horizontal axis indicates the rank of each feature, ranging from 0 (top) to 299 (bottom), based on the score given by each FSL extension. The objective is not to trace the trajectory of individual features across executions, but rather to distinguish the patterns characterizing informative versus noisy features, thereby expanding our understanding of algorithm stability.

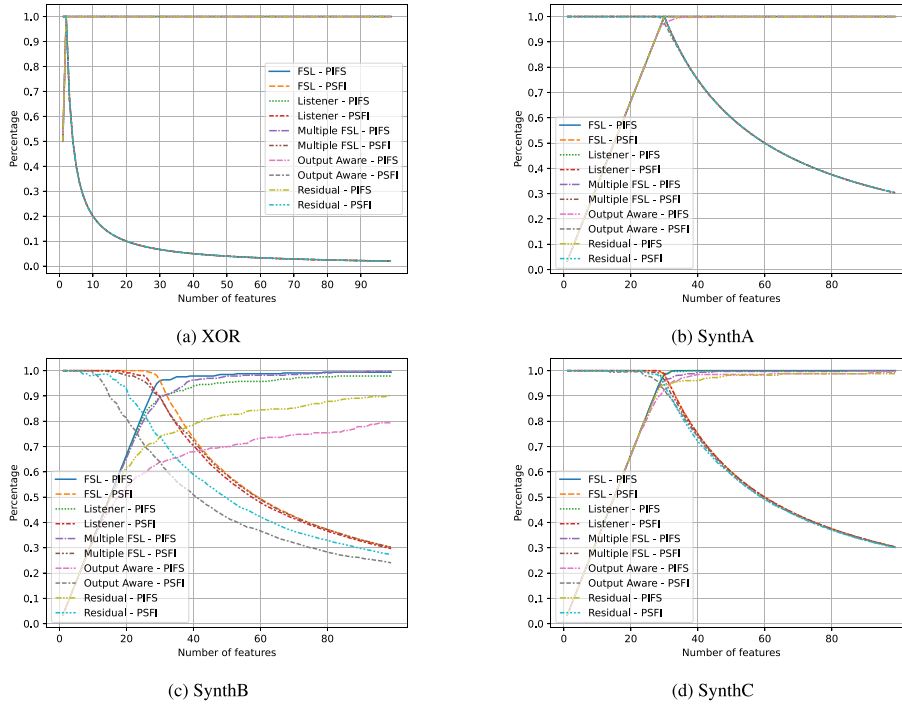


Fig. 6. Analysis of PIFS and PSFI on XOR, SynthA, SynthB and SynthC datasets between original FSL and proposed extensions for increasing selection subsets. All score values are the average over rankings generated by 11 independent model executions. Vertical axis represents the scores constrained from 0 to 1 as both measures are percentages. Horizontal axis represents the subset size selected from the top- n features with the highest weights.

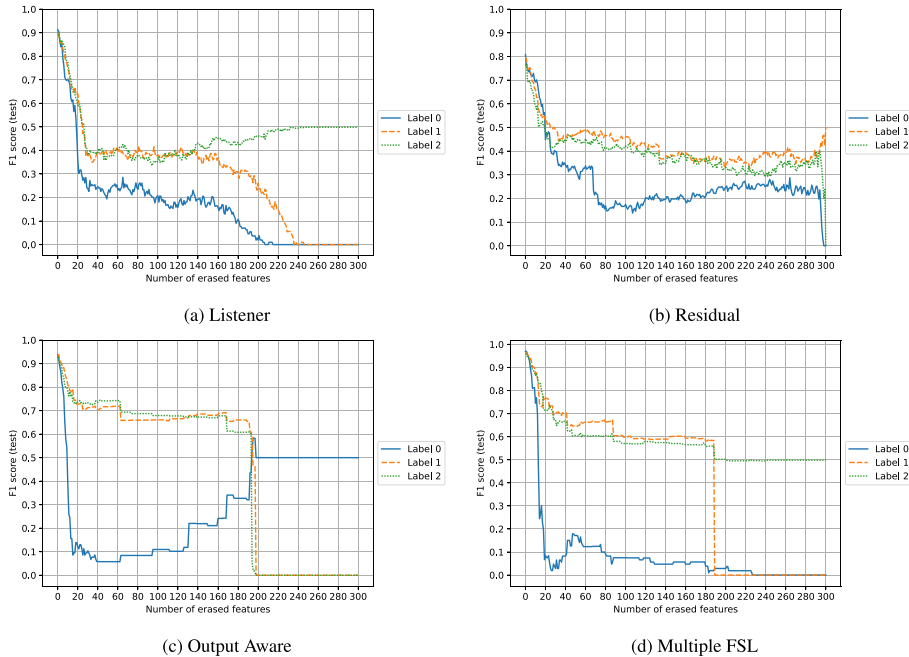


Fig. 7. Feature erasure (occlusion) based on SynthC using feature weights related to label 0 (blue line). Each line represents the F1 score trend related to a single label as we erase the top- n features based on their weights in descending order. All values were calculated using the test dataset.

Consistent with the stability evaluation on XOR dataset (Table 6), a distinct pattern emerges: informative features are consistently positioned at the top of the graph, while noisy features remain at the bottom. However, the precise positioning of relevant and irrelevant features varies between executions. Notably, the Listener method places a higher number of features in the upper ranks. This aligns with the observations in Fig. 7. Listener is not able to assign label-specific feature weights; it consistently maintains all informative features at the top, whereas Output Aware and Multiple FSL only assign higher scores to informative features related

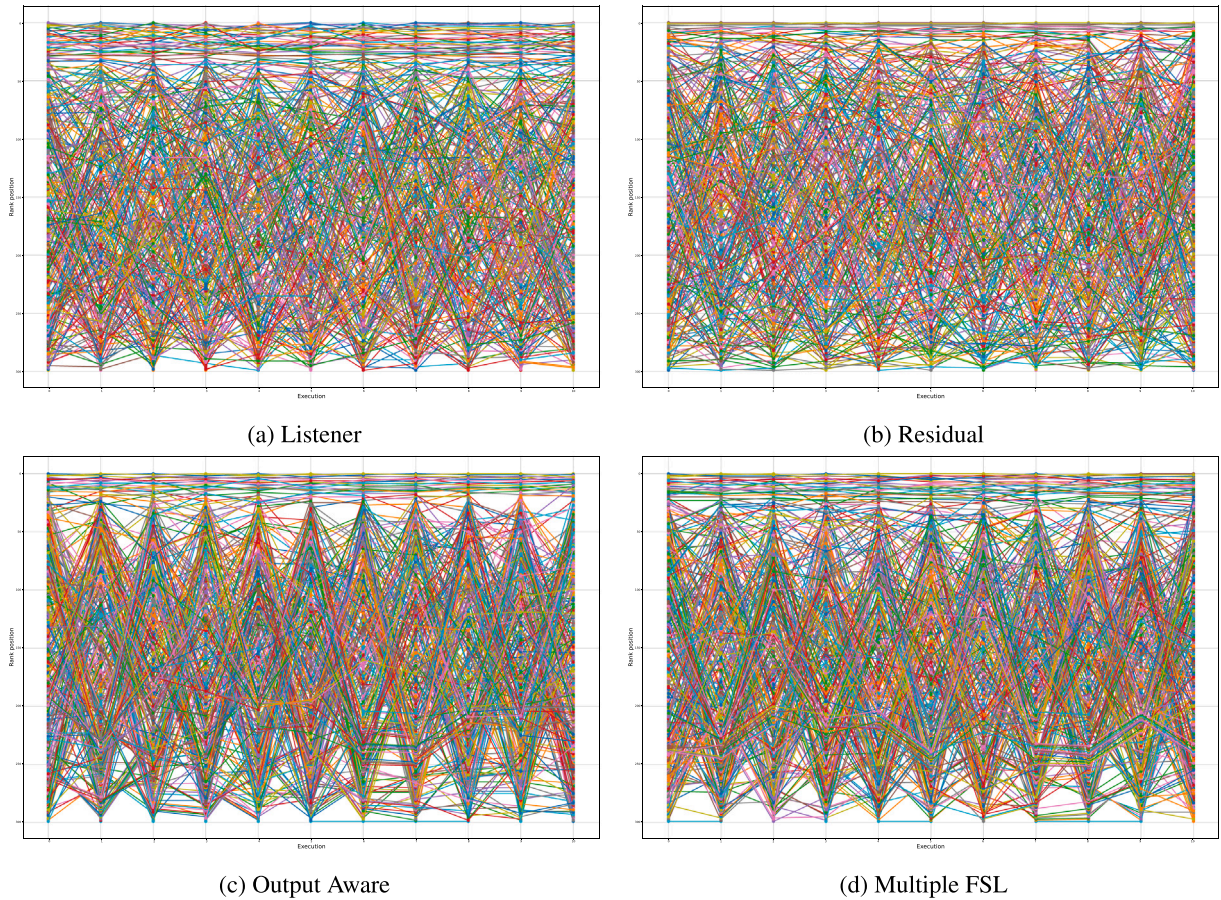
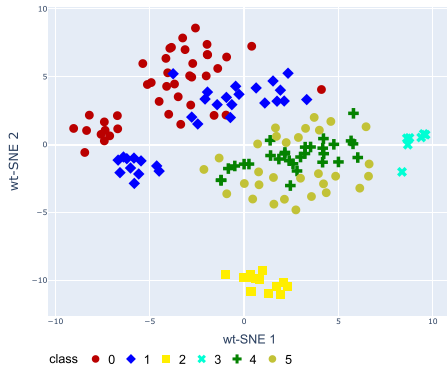


Fig. 8. Feature ranking position related to label 0 across proposed extensions' executions on SynthC. All rankings were created from the feature weights related to label 0, in descending order. Each algorithm was executed 11 times. The vertical axis represents each individual execution, while the horizontal axis shows the ranking position of each feature, from 0 (top) to 299 (bottom), within that run. Different colors represent different features. Given the high density of lines, this visualization is not intended for tracking individual features. Instead, the objective is to highlight a macro-level pattern: the clear distinction between relevant features, which form stable, straight bands consistently ranked at the top, and noisy features, which exhibit highly chaotic and fluctuating trajectories at the bottom of the graph. Additionally, the results demonstrate how the positions of both informative and noisy features vary across executions within each group.

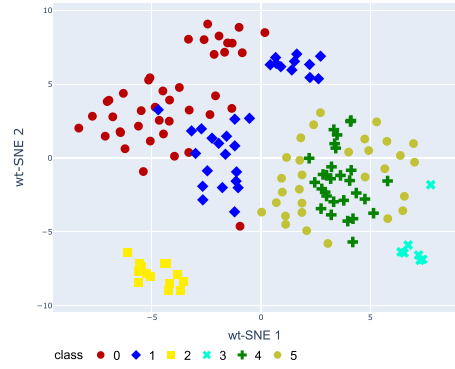
to label 0. Conversely, Residual exhibits the highest instability, making it difficult to distinguish between the informative features at the top and the noisy ones at the bottom. Additional experiments involving the synthetic datasets are provided in Figs. 7–8 and Tables 1–2 of the **Supplementary Material**.

To assess the performance of the algorithms on real-world data, we employed the Breast dataset. When comparing the prediction scores, FSL, Listener, and Multiple FSL achieved statistically significantly better performance for all labels considering precision, recall and F1 score metrics. Residual demonstrated the lowest prediction performance, with inconsistent results for most of the labels. The complete results with statistical significance are provided in Table 3 of the **Supplementary Material**. All algorithms, except Residual, achieved high prediction scores, therefore we expect they are able to assign higher weights to relevant features, similar to the observations made for synthetic datasets. Fig. 10 presents an analysis of the impact of subsets with the top 1000 features, ranked by weight, on the predictive performance of a Support Vector Classifier (SVC). The analysis was limited to the top 1000 features as most algorithms needed less than 100 features to achieve the best score of 1.0. Residual extension had the lowest performance when compared to the other feature weighting algorithms, confirming that it was not able to assign higher weights to the most relevant features compared to the other algorithms. In contrast, FSL, Multiple FSL, and Listener achieved high performance with feature subsets of less than 100 features, while Output Aware required fewer than 200 features to achieve comparable results, which suggests that they were able to detect more relevant features by assigning higher weights to them.

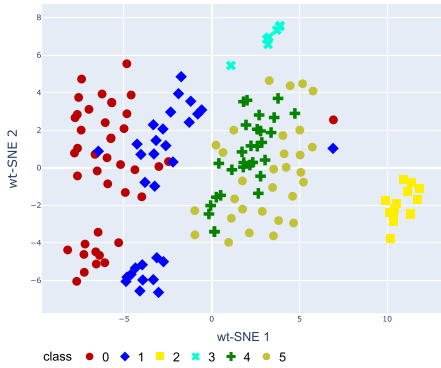
In addition to assigning higher weights to the most relevant features, the proposed method must be able to identify which features are relevant to specific labels in real-world datasets. Fig. 9 presents the weighted t-SNE visualizations for each proposed extension, using the feature weights specifically assigned to label 0 (red dots) of Breast dataset. When comparing the weighted t-SNE visualizations of Listener (Fig. 9b) and Residual (Fig. 9c) to the default t-SNE (Fig. 9a), no significant improvement in label 0 clustering is observed. While Output Aware (Fig. 9d) had a more noticeable impact on the overall visualization, it mainly failed to improve the clustering of label 0 samples. Among all algorithms, only Multiple FSL successfully enhanced the separation of label 0 samples specifically. While



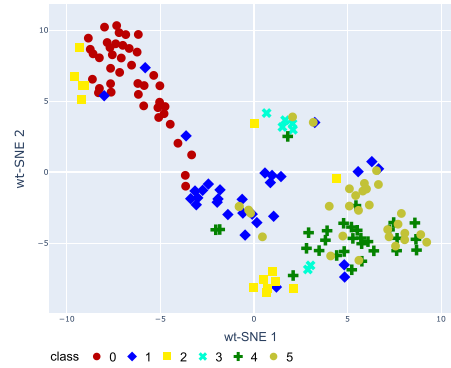
(a) Without weights



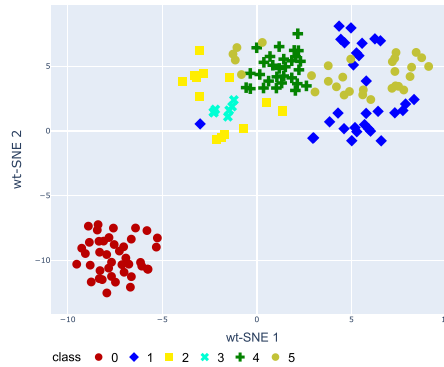
(b) Listener



(c) Residual



(d) Output Aware



(e) Multiple FSL

Fig. 9. Visualization of weighted t-SNE using label 0 (red dots) feature weights from Breast dataset. Each label is represented by a distinct color, with red assigned to label 0. Fig. 9(a) presents the original t-SNE without feature weights. Fig. 9(b)–(e) represents the weighted t-SNE for all proposed extensions using the assigned weights for each feature related to label 0.

the results for label 0 improved, the performance for other labels declined. This indicates that by prioritizing features specifically relevant to a label, the algorithm omitted features necessary for distinguishing the others. This behavior is consistent with an algorithm capable of identifying label-specific importance. Fig. 11 confirms that Multiple FSL final model relies heavily on the features that it classified as more relevant as shown by applying feature erasure based on label 0 feature relevance, represented by the blue line. Listener and Multiple FSL were significantly more affected by feature erasure compared to the other algorithms, suggesting that these algorithms gave highest weights to features essential for label 0 predictions. Although in Listener the performance on other labels was more affected than in Multiple FSL, this aligns with its lower ability to select label-specific relevant features, as observed in the analysis of synthetic datasets and in weighted t-SNE visualization. Output Aware, despite its ability to identify label-specific

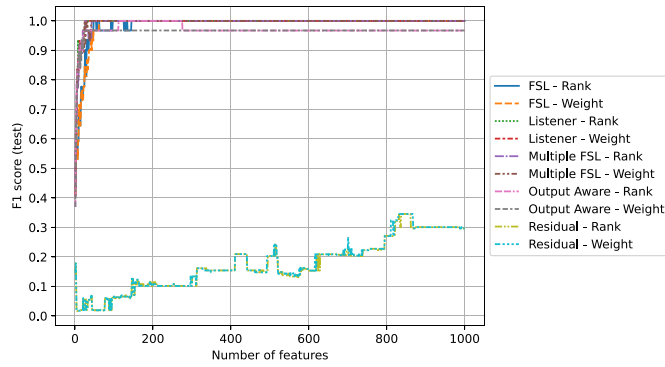


Fig. 10. Comparing SVC prediction performance on top-1000 features ranked by weight from FSL and proposed extensions for Breast dataset. The vertical axis shows the F1 score on test dataset. The horizontal axis shows the number of features used, from 1 to 1000. Feature subsets were defined by the general feature weights given for each specific method. For each method, two comparisons were conducted: one using only feature subsets, and another using weighted features.

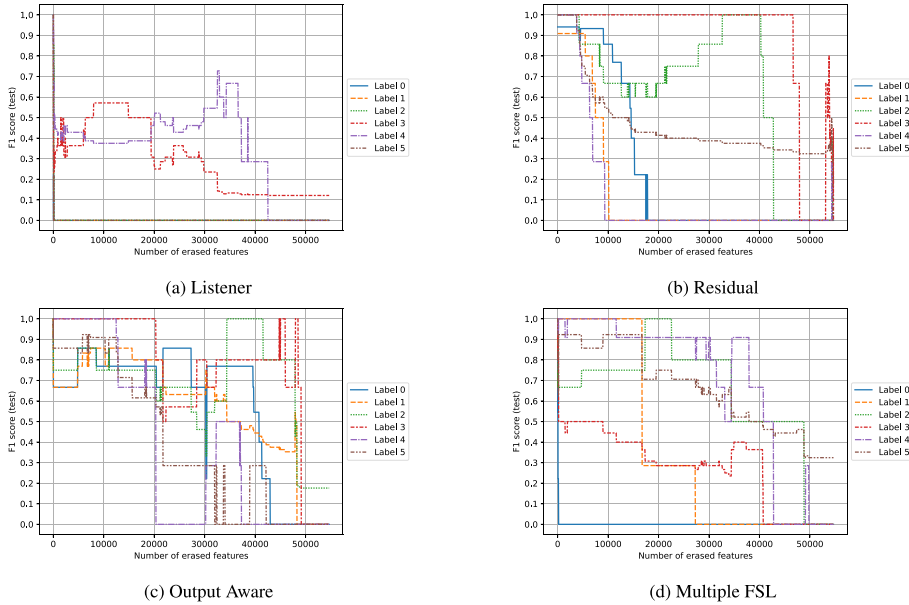


Fig. 11. Impact of feature erasure based on label 0 feature weights for Breast dataset. The features were erased from the dataset following a ranking based on the feature weights given for label 0. The horizontal axis shows the number of erased features, while the vertical axis displays the F1 score calculated on the test dataset.

feature weights demonstrated with synthetic datasets, exhibited decreased performance on microarray datasets. In such cases, the model may prioritize noisy features over truly relevant ones, effectively “memorizing” the dataset using irrelevant patterns.

5.3. Proposed extensions results

When analyzing XOR and SynthA datasets, we could confirm that all extensions are able to assign highest weights to relevant features. However, when considering label-specific feature relationships within the SynthB and SynthC datasets, only Output Aware and Multiple FSL exhibited the capability to identify such distinctions. All proposed extensions increase memory usage by storing label-specific feature weights, unlike in the original FSL implementation. With M features and L labels, the original FSL adds M new weight parameters to the model, while the proposed extensions add $M \times L$ new weights. For Listener, these label-specific weights are accumulated as detached statistics after each batch rather than optimized as trainable parameters, so its GPU peak memory usage remains close to that of the original FSL implementation. Output Aware adds a new calculation during the forward pass that causes a small increase in memory usage based on the number of features, but this increase remains independent of the number of labels. Residual adds a small residual connection computed separately from the main forward path, which causes no measurable impact on memory consumption compared to the original FSL. Multiple FSL, on the other hand, increases the size of the weighted input based on both the number of features and labels, making it the only extension whose GPU memory usage scales with the number of labels. In our experiments, this increase in memory did not translate into longer execution times, as the hardware had sufficient memory to accommodate all network parameters. However, it is important to note that Multiple FSL can instead be implemented to run a separate forward pass per class rather than a single forward pass with an enlarged input, reducing memory consumption to the level

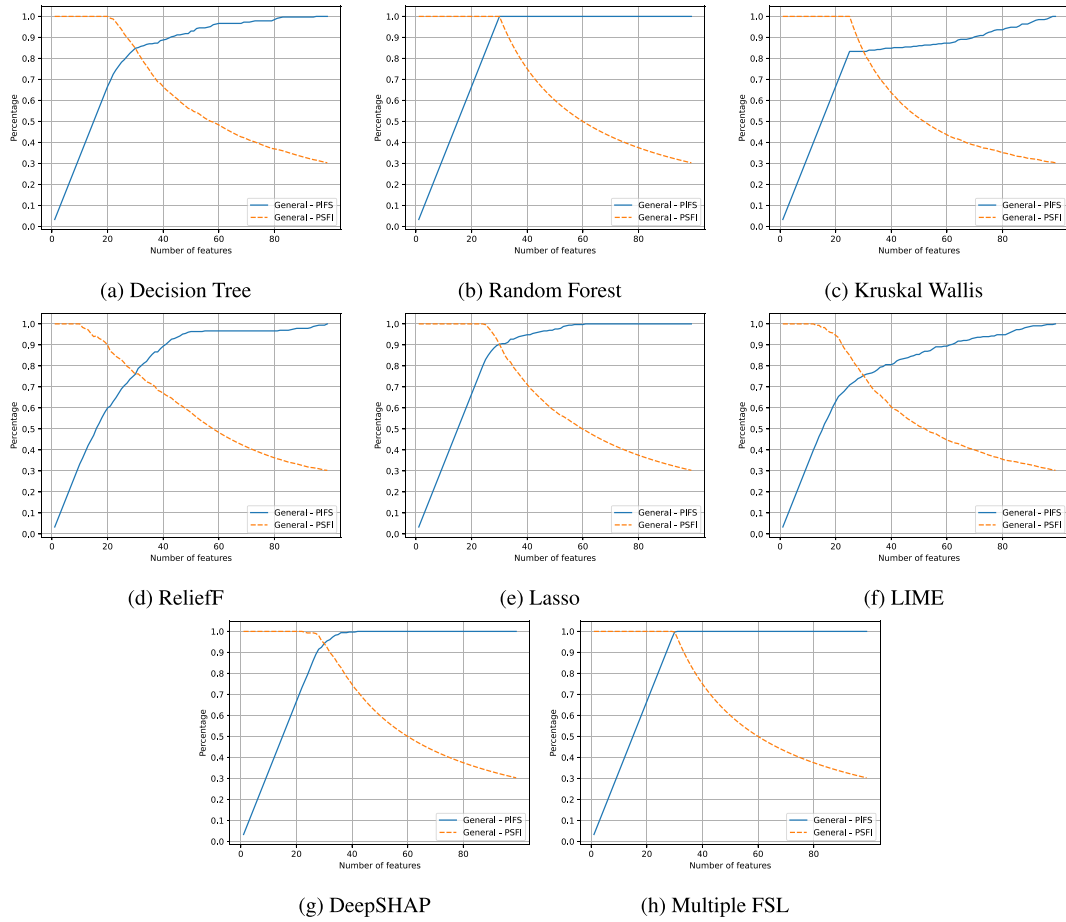


Fig. 12. Evaluation of PIFS and PSFI on the SynthA dataset for Multiple FSL and comparison of feature weighting algorithms. all scores are averages over 11 independent runs per algorithm. Vertical axis represents scores constrained from 0 to 1, and the horizontal axis shows subset sizes based on the top- n features by descending feature weight. PSFI is represented by Orange lines, PIFS by blue lines. With 30 features, a perfect result has PSFI stable at 1.0 up to the top 30 features and PIFS reaching 1.0 at the 30th feature.

of the other extensions while increasing execution time. On the Breast dataset (6 labels), for example, running a forward pass per class caused an average $3.8\times$ increase in execution time when compared with the original FSL. Despite the higher memory usage, Multiple FSL consistently achieved the best results across all datasets. Output Aware, while effective on synthetic datasets, caused a degradation in the model prediction metrics and underperformed on the Breast dataset, which has more features and fewer samples than the other datasets, yielding lower accuracy and lower capacity to identify relevant features. Unlike Output Aware, given the multiple feed-forward executions, Multiple FSL is able to train each Feature Selection Layer with all available samples, reducing the bias and achieving the most consistent performance. Similar to the original FSL implementation, the selection layers directly influence the feature values used for prediction. However, each layer affects only the label to which it is explicitly associated. In the following section, we will compare the proposed Multiple FSL with other feature weighting techniques.

5.4. Comparative analysis with existing feature weighting algorithms

We start by comparing the overall performance and ability of each method to identify general relevant features by testing them on SynthA. Given the constraints of time and the scope of the research involving multiple datasets and evaluation methods, the presented experiments use a subset of the available feature weighting algorithms: Decision Tree, Random Forest, Lasso, Kruskal Wallis, ReliefF, LIME, and DeepSHAP. DeepSHAP was executed using a representative background of 1000 instances, as recommended by its library documentation to obtain the best performance within a suitable time. LIME weights were based on the results of 1000 samples. DeepSHAP, LIME, and Multiple FSL shared the same neural network architecture, except for the Feature Selection Layers in the last one. Fig. 12 illustrates the results on the SynthA dataset using rank-based feature subsets. The blue line represents the average PIFS (Percentage of Informative Features Selected), while the orange line represents the PSFI (Percentage of Selected Features that are Informative). The vertical axis denotes the metric score (ranging from 0 to 1), and the horizontal axis represents the subset size (top- n features) ranging from 0 to 100. For example, in the Decision Tree results (Fig. 12a), subsets containing the top-1 to top-20 features achieve a PSFI score of 1.0. This indicates that all features within these ranks are informative across all executions. However,

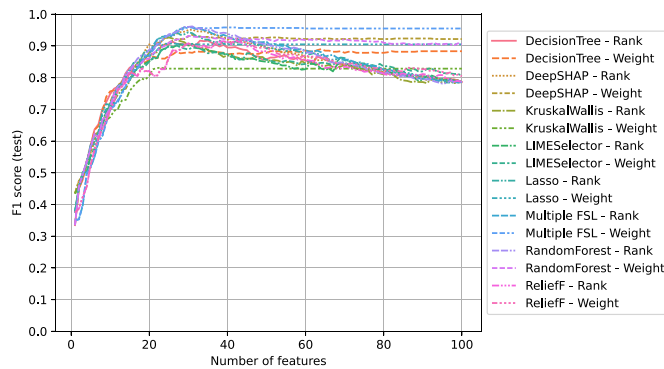


Fig. 13. Comparing SVC prediction performance when trained with top- n features ranked by weight from FSL and proposed extensions for SynthA dataset. The vertical axis shows the F1 score on test dataset. The horizontal axis shows the number of features being used. Feature subsets were defined by the general feature weight given for each specific algorithm. For each algorithm, two comparisons were conducted: one using only feature subsets, and another using weighted features.

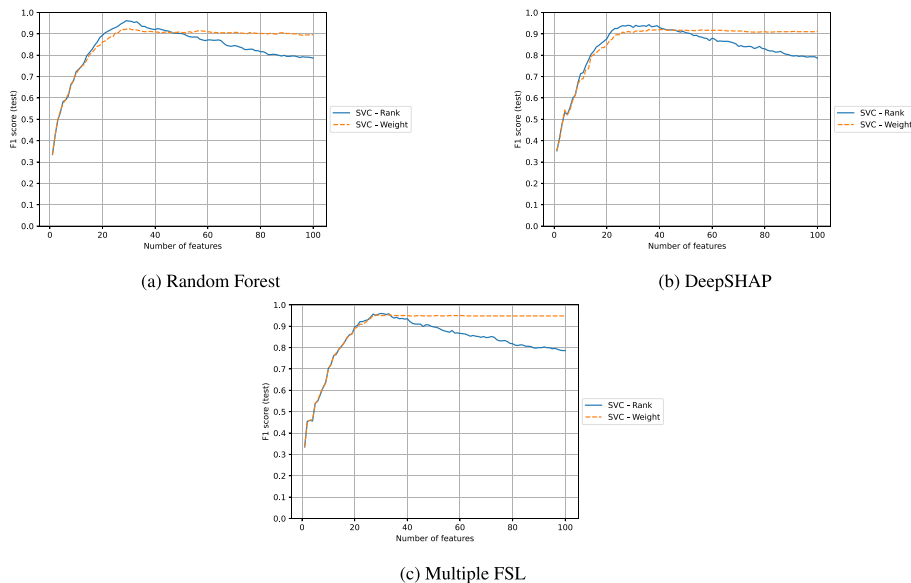


Fig. 14. Comparing SVC prediction performance when trained with top- n features ranked by weight from FSL and proposed extensions for SynthA dataset. Multiple feature subsets of increasing size are generated based on the feature weights assigned by each method. These subsets are then used to train an SVC model, and its prediction score is reported. Each algorithm's scores are presented separately for a better individual analysis.

starting at the top-21 subset, noisy features are introduced. The blue line (PIFS) only reaches 1.0 at the top-80 mark, meaning all informative features are only fully captured in subsets of size 80 or larger for all executions. This suggests that the Decision Tree consistently ranks 20 informative features highly but assigns lower scores to the remaining 10. Most algorithms consistently assigned high weights to at least 20 of the 30 informative features. Notably, Random Forest (Fig. 12b) achieved the best results for the SynthA dataset. It consistently identified the 30 informative features as the most relevant, evidenced by the intersection of the PIFS and PSFI curves exactly at the top-30 subset. Multiple FSL and DeepSHAP followed closely, achieving average PIFS scores of 0.997 and 0.942, respectively, when considering the top-30 features. As all algorithms successfully assigned higher weights to relevant features, we expect them to be able to improve the performance of a separate model given only their features with highest weights.

Fig. 13 illustrates the impact of different feature subsets for training a Support Vector Classifier (SVC) model to obtain a high accuracy. These subsets are defined based on the general feature weights assigned by each algorithm. All algorithms demonstrated comparable performance, with Random Forest, Multiple FSL, and DeepSHAP achieving the highest F1 scores of 0.96, 0.96, and 0.94, respectively. This aligns with the observation that these models assigned higher weights to a larger number of relevant features. Moreover, when utilizing weighted features (indicated by the top dashed lines), Multiple FSL, DeepSHAP, Random Forest, Lasso, and Decision Tree (in descending order of performance) exhibited greater robustness. Fig. 14 shows the effect of the feature erasure individually for Random Forest, DeepSHAP and Multiple FSL. They maintained higher scores when considering weighted features even with the inclusion of additional noisy features in the dataset. This resilience can be attributed to the assignment of near-zero weights to these less relevant features.

Table 10 presents stability metrics for Multiple FSL and comparison algorithms using the SynthA dataset. We employed the $m - 1$ approach, where the algorithm is executed twice: first with the complete feature set, with m equal to 100 features (Table 4), and

Table 10
Stability comparison between algorithms' feature weights on SynthA dataset. Pearson and Spearman were 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 for 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	0.833±.044	0.607±.097	0.633±.079
Random Forest	0.987±.002	0.688±.042	0.935±.000
Kruskal Wallis	0.999±.000	0.999±.000	1.000±.000
Relieff	0.997±.001	0.977±.007	0.748±.061
Lasso	0.992±.002	0.964±.012	0.925±.047
LIME	0.755±.043	0.475±.101	0.489±.067
DeepSHAP	0.944±.011	0.744±.034	0.947±.026
Multiple FSL	0.966±.005	0.854±.036	0.953±.030

subsequently with the top-ranked feature removed, totaling 99 features. Kruskal-Wallis proves to be the most stable algorithm across all metrics. Relieff and Lasso also demonstrate stability regarding feature weights (Pearson) and rankings (Spearman). However, their stability decreases, particularly for Relieff, when evaluating the Jaccard score for the top-30 features. Finally, while neural network-based algorithms generally exhibit lower stability, DeepSHAP and Multiple FSL achieve strong results across all metrics, performing particularly well on the Jaccard index. Complementary results using SynthB dataset are provided in Table 4 of the **Supplementary Material**.

Lasso, LIME, DeepSHAP and Multiple FSL are able to assign label-specific feature weights. To assess their capacity, we used the SynthB dataset, which has distinct features for each label. Fig. 15 presents a heatmap with all label-specific and general weights for the top-30 features following SynthB label 2 specific weights. It confirms that all algorithms were able to assign different weights to features based on the label. Additional experiments involving the synthetic datasets are provided in Figs. 10, 11 and 14 of the **Supplementary Material**. To further evaluate the capabilities of feature weighting algorithms we compare them on three microarray datasets, and we aim to assess their performance on real world high-dimensional datasets with limited samples. One of these datasets, Liver, is binary. For Liver, we evaluate the algorithms' ability to assign higher weights to the most relevant features without considering label-specific relations. Due to the limited sample size of microarray datasets and the high number of features, the background size for DeepSHAP was reduced to 50. Similarly, LIME was configured to evaluate 50 stratified samples to alleviate the processing time necessary associated with the high-dimensional feature space.

Fig. 16 shows the best weighted t-SNE result obtained on Liver dataset among 11 executions. Multiple FSL and Lasso were able to achieve better separation of the samples by their labels. Decision Tree, Random Forest, Kruskal Wallis, and DeepSHAP also had good results. Table 11 presents the average HD and 2D silhouettes after applying the feature weights compared with the silhouettes without them ("None"). As observed in the visual representations, Multiple FSL and Lasso had superior performance in 2D silhouette.

Fig. 18 presents the weighted t-SNE results from DeepSHAP, LIME, Lasso and Multiple FSL on Colorectal dataset using label 1-specific weights (blue diamonds). Only Multiple FSL demonstrated a significant improvement specifically for label 1 samples. The findings presented in Table 16 corroborate our visual observations with HD and 2D silhouette coefficients resulting from an one-versus-all (OVA) approach centered on label 1. Multiple FSL demonstrated the highest performance across all scenarios while DeepSHAP presented a slight improvement.

Fig. 19 presents the outcomes of feature erasure, conducted using the feature weights assigned to label 4 in the Breast dataset by Lasso, LIME, DeepSHAP, and Multiple FSL. In this validation process, a rapid drop in predictive performance for a specific label indicates that an algorithm successfully identified the most critical label-specific features. As desired, Lasso and Multiple FSL exhibited a substantial decline in performance when only a small number of their top-ranked features for label 4 were removed, confirming their accuracy. DeepSHAP demonstrated a similar sensitivity, albeit requiring a larger number of removed features to achieve the same drop. LIME, likely constrained by the limited sample size of the Breast dataset, was unable to accurately identify label-specific features. Fig. 20 presents the weighted t-SNE visualization using general and label-specific feature weights generated by Multiple FSL for each label available in Breast dataset. Features 2 (yellow) and 3 (cyan) were already distinguishable in the original t-SNE visualization. Note that labels 2 (yellow) and 3 (cyan) are already well defined in the original t-SNE visualization, but in the weighted t-SNE we can notice a slight improvement for them and a greater decrease for the other labels, which means that features relevant to other labels were "erased". However, a huge improvement can be noticed in all other labels, confirming Multiple FSL's advantage in identifying label-specific feature weights.

Table 17 compares Multiple FSL, LassoNet, TabNet, and SAGE using the SynthB dataset. Regarding general feature weights, SAGE performed better on average, with values slightly better than those of Multiple FSL and LassoNet. However, Multiple FSL outperformed the other methods when considering label-specific feature weights. Fig. 11 of the **Supplementary Material** presents a weighted t-SNE visualization using the label 1 specific weights generated by the methods. This visualization visually demonstrates Multiple FSL's superior capability in defining label-specific feature weights.

Table 12 presents the execution times (in seconds) for the evaluated feature weighting methods across five distinct datasets. The computational efficiency of the algorithms can be broadly categorized into traditional baseline models and more advanced neural or

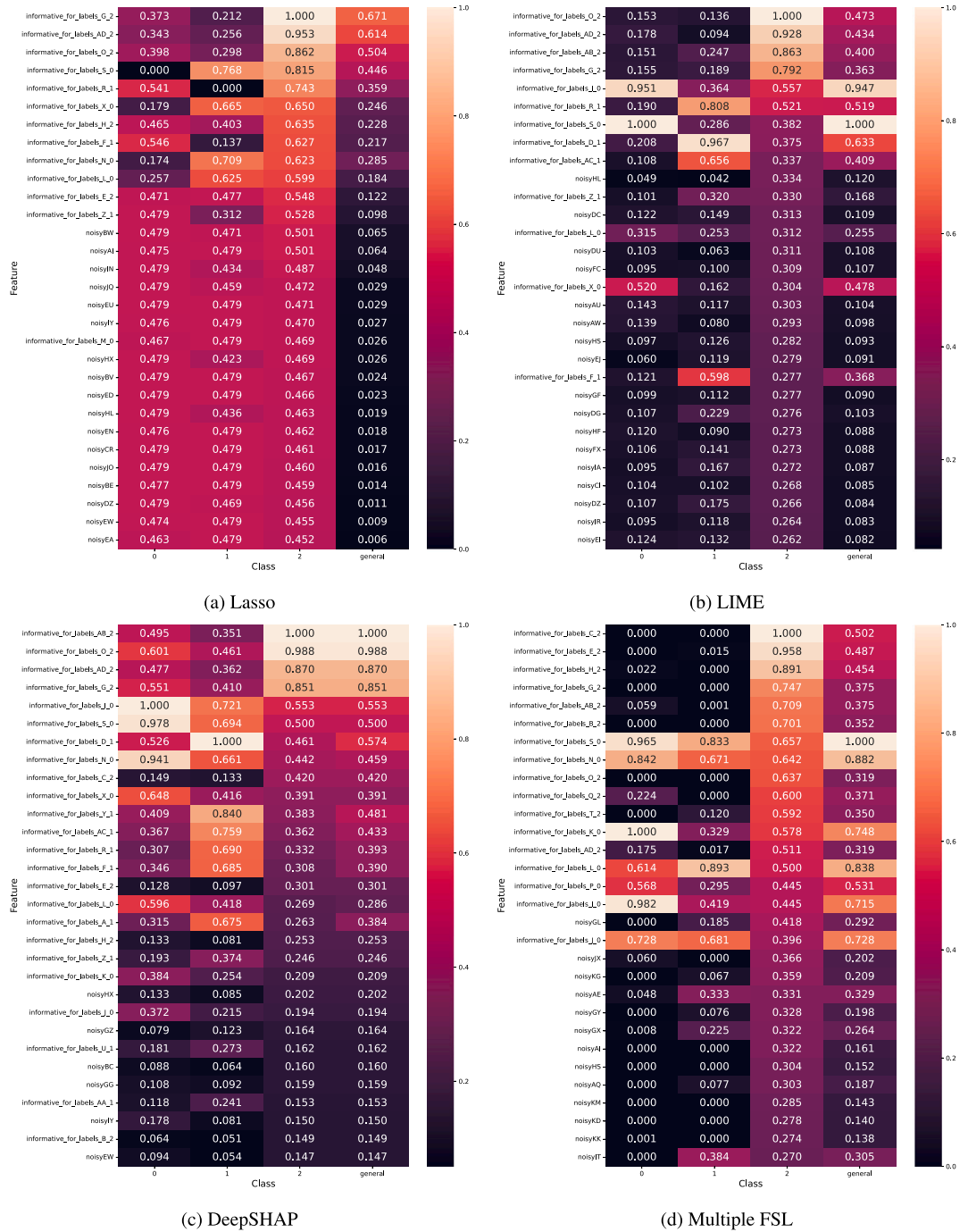


Fig. 15. Heatmap visualization of top-30 features for Lasso, LIME, DeepSHAP, and Multiple FSL, based on their label-specific weights for label 2 of the SynthB dataset. Normalized feature weights for each algorithm, are ranked by their label 2 (third column) feature weights. Feature name prefixes indicate whether they are informative or noisy, followed by an identification letter sequence, which ensures that all names are unique, and the label to which they are related. For example, *informative_for_labels_A_1* is the informative feature A that is related to label 1, while *noisyAI* is a noisy feature identified by the letters “AI”.

interpretation-based frameworks. As expected, traditional machine learning and statistical methods (Decision Tree, Random Forest, Kruskal-Wallis, ReliefF, and Lasso) achieve the lowest execution times, frequently running in under two seconds. These methods consistently achieve statistically significant speed advantages due to their lower algorithmic complexity and the absence of iterative deep learning training cycles. However, when focusing on deep learning-based feature selection and model explanation methods, the proposed Multiple FSL approach exhibits strong scalability and highly competitive computational efficiency. Specifically, it introduces a small computational overhead compared to the Base Neural Network and significantly outperforms complex interpretation-based

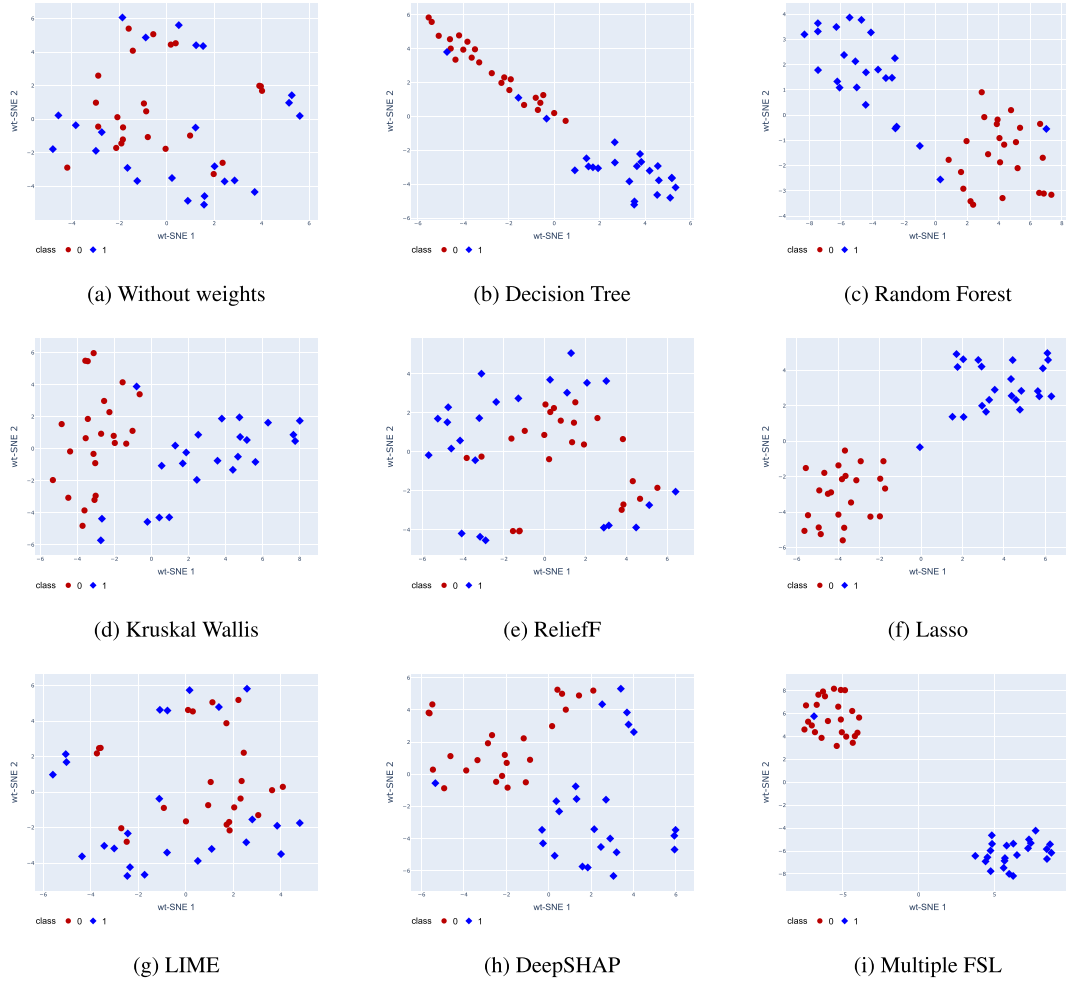


Fig. 16. Liver dataset weighted t-SNE visualization using general feature weights. Fig. 16(a) presents the t-SNE visualization. The other figures present the weighted t-SNE using the general feature weights extracted from each algorithm. The two classes are displayed in red circles (0) and blue diamonds (1).

Table 11

Average HD and 2D silhouette coefficients for Liver dataset. all feature weights were obtained from the general feature weights, not label-specific. Averages were calculated using the weighted t-SNE results given the feature weights generated by each algorithm across 11 different executions. In bold is the highest average for each column. The best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis and Dunn's tests, are highlighted in Orange.

Weight source	HD silhouette	2D silhouette
None	.038	.049
Decision Tree	.409±.073	.498±.056
Random Forest	.220±.019	.517±.047
Kruskal Wallis	.148±.013	.305±.061
ReliefF	.054±.003	.066±.016
Lasso	.382±.038	.661±.033
LIME	.039±.001	.051±.010
DeepSHAP	.130±.007	.232±.057
Multiple FSL	.302±.046	.668±.061

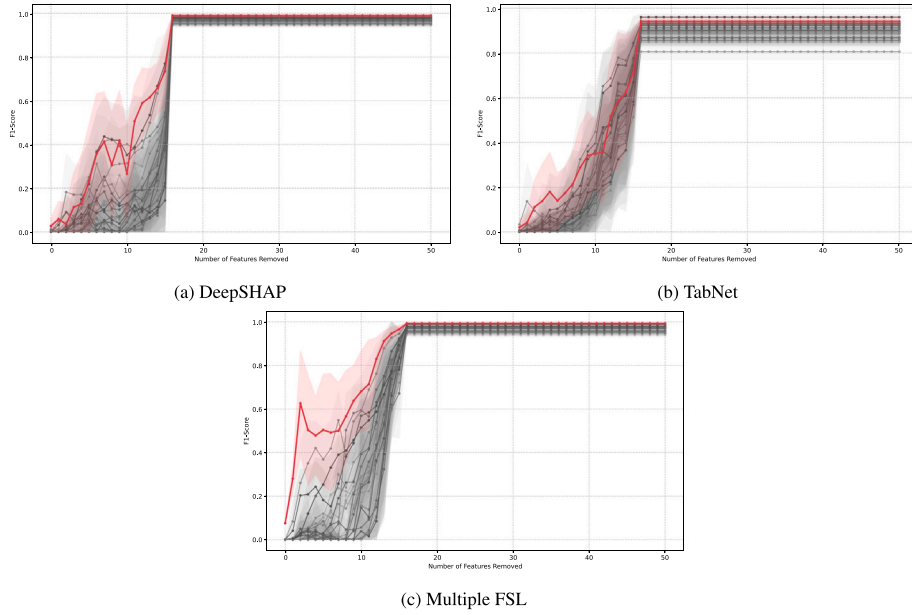


Fig. 17. Impact of the most relevant label-specific features on predictive performance in the Letter Recognition dataset. the graph illustrates the effect of initially erasing all features and then sequentially re-enabling them, one by one, according to the label-specific feature weight ranking for label 0. The red line denotes the F1-score of the target label, while the gray lines represent the F1-scores of all other labels.

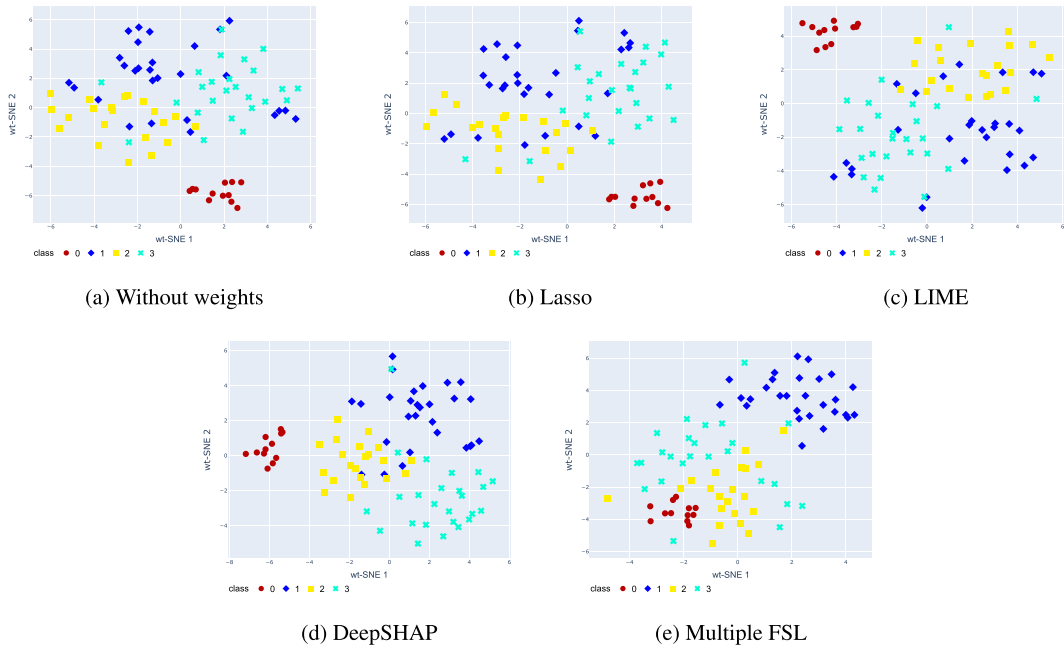


Fig. 18. Colorectal dataset weighted t-SNE visualization using feature weights related to label 1. Fig. 18(a) presents a t-SNE visualization without feature weights. Label 1 is represented by blue diamonds.

and embedded deep learning methods like LassoNet, TabNet, LIME, and DeepSHAP. Furthermore, Multiple FSL demonstrates notable robustness on high-dimensional data. For instance, on the Breast dataset, the SAGE method resulted in an out of memory error. In our experiments, we utilized SAGE's default Kernel estimator, which is highly sample-efficient but tracks a full $(M \times M)$ covariance matrix for its confidence intervals, leading to an $\mathcal{O}(M^2)$ memory complexity (where M is the number of features). For the Breast dataset (54,675 features), this accumulator requires tens of gigabytes of RAM, quickly exceeding the system's capacity. While SAGE offers an alternative Permutation estimator with an $\mathcal{O}(M)$ memory footprint, it computes marginal contributions by adding features sequentially, requiring roughly M model calls per permutation. Without the Kernel's variance-reduction techniques, it needs more permutations to converge, rendering it computationally prohibitive in terms of execution time for this dataset. In contrast, Multiple

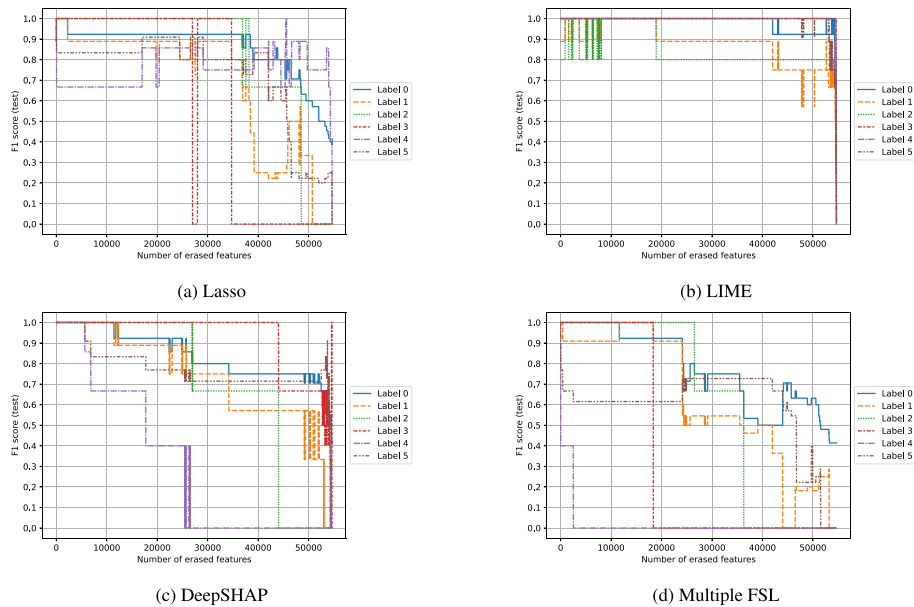


Fig. 19. Impact of feature erasure based on label 4 feature weights for Breast dataset. The features were erased from the dataset following a ranking based on the feature weights given for label 4 by each algorithm. The horizontal axis shows the number of erased features, while the vertical axis displays the F1 score calculated from the test dataset.

FSL handles the data efficiently in 39.89 ± 5.54 seconds. This execution time vastly outperforms those of other complex alternatives like LIME (487.27 ± 21.86 s) and LassoNet (386.92 ± 48.85 s). Crucially, the transition from the standard FSL framework to the proposed Multiple FSL achieves these advantages without incurring a notable performance penalty.

Peak GPU memory usage (VRAM) per feature weighting method across the five datasets is reported in Table 13. Although the Base Neural Network is not itself a feature weighting method, it is included for comparison: DeepSHAP directly reuses this same trained network as its underlying classifier, while FSL, Multiple FSL, and LassoNet each matches its hidden-layer architecture but trains an independent instance as part of their own optimization. LassoNet, in contrast, relies on a dedicated architecture unrelated to the base model. On XOR, SynthB, Isolet, and Letter Recognition, memory consumption remains modest and comparable across methods, ranging from roughly 18 MB to 145 MB. TabNet is the most memory-efficient method on three of these four datasets (17.7 MB on XOR, 21.1 MB on SynthB, 26.2 MB on Isolet), while the FSL achieves the best result on Letter Recognition (23.8 MB). The Breast dataset, however, exposes an exception to this pattern. While every other method's memory usage on Breast remains within the same order of magnitude (83.1–127.7 MB), TabNet's peak GPU allocation increases to over 12 GB. This behavior is consistent with TabNet's internal feature-grouping mechanism, which allocates a dense matrix scaling quadratically with the number of input features when no explicit feature grouping is provided. Breast, with 54,675 features, is by far the highest-dimensional dataset evaluated, and this quadratic growth accounts for the disproportionate memory demand observed. This result highlights an important practical limitation of TabNet's default configuration when applied to high-dimensional, ungrouped feature spaces such as gene-expression data. Multiple FSL, on the other hand, is the only method whose memory usage grows considerably with the number of labels, as can be seen on Isolet (114.3 MB) and Letter Recognition (144.8 MB). This demonstrates that Multiple FSL's memory usage is primarily driven by the number of output labels rather than the number of input features, consistent with its architecture maintaining a dedicated per-label weighting structure. However, unlike TabNet's quadratic blowup, this growth is linear rather than quadratic.

The Isolet and Letter Recognition datasets represent a distinct challenge for label-specific feature weighting methods, as both combine a small number of features with a large number of classes. All methods evaluated in the present work produced similar results, consistently identifying the same features as relevant for multiple labels. Among the methods capable of producing label-specific feature weights, DeepSHAP, TabNet, and Multiple FSL achieved the best scores. Table 14 shows the 2D silhouette coefficients per label for the Isolet dataset, computed using the label-specific feature weights produced by these three methods. As can be observed, most coefficients are negative or near zero, indicating that the weighted t-SNE embedding was unable to separate the samples belonging to the target label from the remaining samples. Although some improvement over the baseline can be observed, it remains minimal. Fig. 17 shows the impact of initially erasing all features from the Letter Recognition dataset and sequentially re-enabling them, one by one, based on the label-specific feature rankings for label 0. The red line represents the F1-score for label 0, while the gray lines represent the performance of the remaining labels. For all three methods, label 0 is, on average, the most heavily impacted class during the inclusion of the first few features. By the time 15 features are enabled, the F1-scores for all labels have already plateaued. This demonstrates that while all methods successfully select appropriate features for each label, the majority of features affect multiple classes simultaneously, which complicates the task of identifying strictly label-specific features.

In deep learning architectures that rely on raw predictions to infer feature importance, it is crucial to ensure that the label-wise logits are well-calibrated. A well-calibrated model outputs confidence scores that accurately reflect its true probability of correctness,

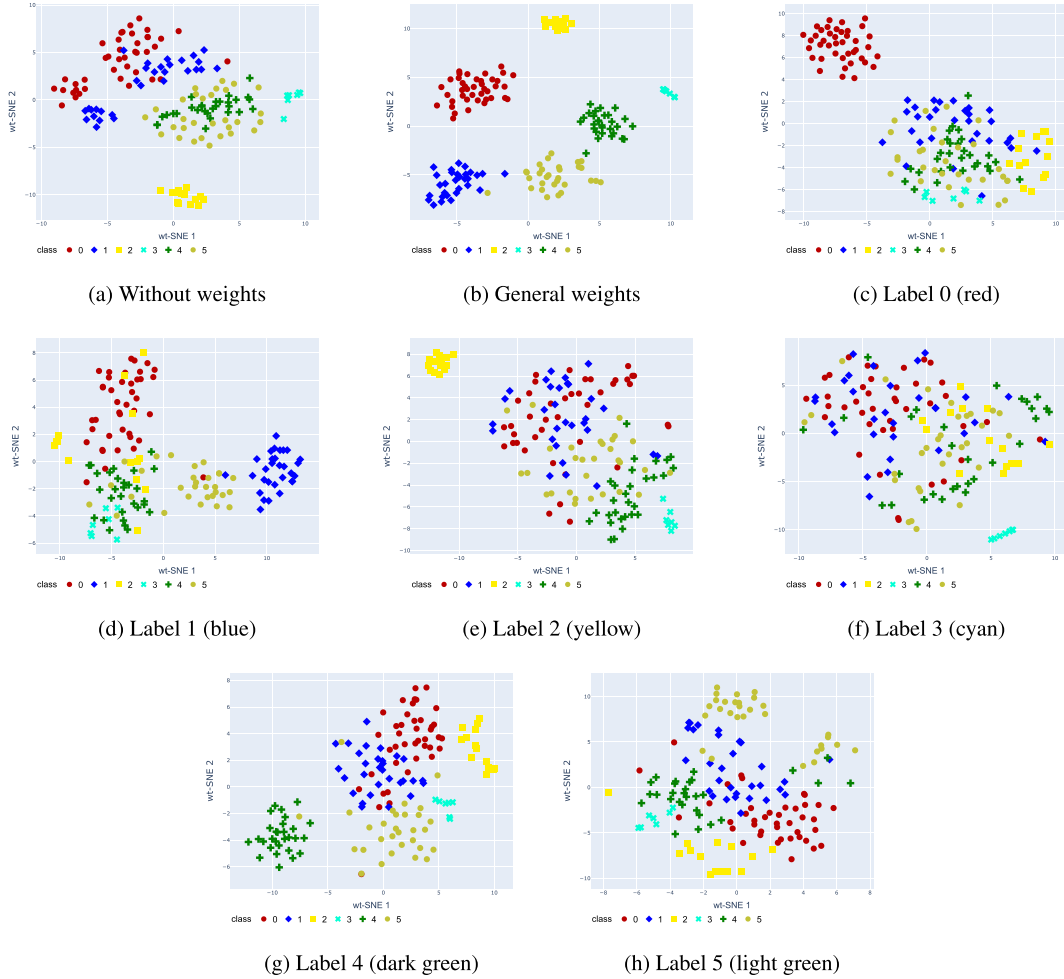


Fig. 20. Breast dataset weighted t-SNE visualization using feature weights from Multiple FSL. Fig. 20(a) presents the t-SNE visualization. Fig. 20(b) presents a weighted t-SNE visualization with general weights. Fig. 20(c) to (h) presents the weighted t-SNE visualization with label-specific weights.

Table 12

Execution times (s) for each feature weighting method across the XOR, SynthB, Breast, Isolet, and Letter Recognition datasets. Values represent the mean $\pm$ standard deviation over 11 executions. To ensure a fair comparison with embedded methods, the times reported for post-hoc methods represent the sum of the base model training time and the method's execution time. Note that SAGE's computational cost is not reported for the Breast dataset because it resulted in an out of memory error. The method's default kernel estimator requires tracking an $\mathcal{O}(M^2)$ covariance matrix, which demands tens of gigabytes of RAM for the 54,675 features of the Breast dataset, exceeding standard hardware limits. In bold is the best average per dataset. Best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis and Dunn's test are highlighted in Orange.

Model	XOR	SynthB	Breast	Isolet	Letter Recognition
Base Neural Network	4.59 $\pm$ 0.60	39.40 $\pm$ 4.22	14.83 $\pm$ 0.31	13.88 $\pm$ 0.53	98.19 $\pm$ 1.65
Decision Tree	0.00 $\pm$ 0.00	0.76 $\pm$ 0.21	1.24 $\pm$ 0.11	2.13 $\pm$ 0.05	0.04 $\pm$ 0.00
Random Forest	0.06 $\pm$ 0.00	1.94 $\pm$ 0.07	0.66 $\pm$ 0.01	5.73 $\pm$ 0.05	0.85 $\pm$ 0.01
Kruskal Wallis	0.02 $\pm$ 0.00	0.16 $\pm$ 0.03	18.09 $\pm$ 0.56	0.83 $\pm$ 0.00	0.02 $\pm$ 0.00
ReliefF	0.02 $\pm$ 0.01	0.23 $\pm$ 0.04	0.47 $\pm$ 0.01	1.84 $\pm$ 0.02	4.06 $\pm$ 0.06
Lasso	0.00 $\pm$ 0.00	0.06 $\pm$ 0.01	7.00 $\pm$ 0.74	2.57 $\pm$ 0.51	0.15 $\pm$ 0.27
LIME	8.96 $\pm$ 0.62	81.10 $\pm$ 4.86	487.27 $\pm$ 21.86	22.20 $\pm$ 0.58	128.10 $\pm$ 1.83
DeepSHAP	6.75 $\pm$ 0.66	94.54 $\pm$ 4.76	17.63 $\pm$ 0.56	18.79 $\pm$ 0.56	888.77 $\pm$ 64.91
LassoNet	190.77 $\pm$ 3.57	299.53 $\pm$ 19.94	386.92 $\pm$ 48.85	446.22 $\pm$ 45.68	367.71 $\pm$ 2.80
TabNet	11.68 $\pm$ 1.24	65.12 $\pm$ 15.62	178.61 $\pm$ 28.27	256.43 $\pm$ 39.44	196.93 $\pm$ 7.47
SAGE	3.00 $\pm$ 0.15	36.37 $\pm$ 2.05	–	228.40 $\pm$ 58.02	142.65 $\pm$ 21.44
FSL	9.36 $\pm$ 2.16	41.36 $\pm$ 2.37	19.21 $\pm$ 0.44	15.33 $\pm$ 0.65	114.09 $\pm$ 3.64
Multiple FSL	6.99 $\pm$ 0.94	44.32 $\pm$ 1.90	39.89 $\pm$ 5.54	16.55 $\pm$ 0.78	122.56 $\pm$ 2.49

Table 13

Peak GPU memory usage (MB) for each feature weighting method across the XOR, SynthB, Breast, Isolet, and Letter Recognition datasets. Standard deviation is omitted because the variation in peak GPU allocation is minimal; given the same architecture and dataset, the memory footprint remains nearly identical across all executions. All peak GPU memory allocations were measured using an NVIDIA GeForce RTX 5070 Ti (16 GB VRAM). In bold is the best average per dataset and metric.

Model	XOR	SynthB	Breast	Isolet	Letter Recognition
Base Neural Network	19.8	23.3	83.1	35.8	23.7
DeepSHAP	21.7	39.7	97.0	35.8	34.3
LassoNet	20.2	29.3	102.6	47.9	62.1
TabNet	17.7	21.1	12063.8	26.2	28.7
FSL	19.8	24.0	83.7	38.2	23.8
Multiple FSL	21.0	27.1	127.7	114.3	144.8

Table 14

Isolet dataset 2D silhouette coefficients per label. Averages were calculated over 11 executions. A one-vs-all (OVA) strategy was adopted, where all labels other than the target label were collapsed into a single category. In bold is the highest average. Best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis test and Dunn's test are highlighted in Orange.

Label	Baseline	DeepSHAP	TabNet	Multiple FSL
A	-0.182 ± 0.000	-0.173 ± 0.027	-0.128 ± 0.039	-0.181 ± 0.031
B	-0.103 ± 0.000	-0.039 ± 0.069	-0.002 ± 0.025	-0.041 ± 0.063
C	0.144 ± 0.000	0.135 ± 0.052	0.146 ± 0.100	0.179 ± 0.034
D	-0.052 ± 0.000	-0.041 ± 0.038	-0.017 ± 0.033	0.003 ± 0.093
E	-0.048 ± 0.000	-0.034 ± 0.043	0.038 ± 0.036	-0.131 ± 0.050
F	0.020 ± 0.000	0.097 ± 0.038	0.116 ± 0.054	0.104 ± 0.050
G	-0.029 ± 0.000	-0.061 ± 0.043	0.055 ± 0.026	0.062 ± 0.049
H	0.102 ± 0.000	0.172 ± 0.021	0.078 ± 0.125	0.089 ± 0.037
I	-0.038 ± 0.000	0.083 ± 0.069	0.091 ± 0.107	0.157 ± 0.043
J	-0.159 ± 0.000	-0.065 ± 0.055	0.024 ± 0.071	-0.060 ± 0.135
K	-0.161 ± 0.000	-0.027 ± 0.074	-0.039 ± 0.068	0.046 ± 0.040
L	0.016 ± 0.000	-0.020 ± 0.069	0.052 ± 0.080	0.070 ± 0.051
M	-0.139 ± 0.000	-0.121 ± 0.066	-0.021 ± 0.084	0.052 ± 0.057
N	-0.173 ± 0.000	-0.150 ± 0.033	-0.097 ± 0.071	0.008 ± 0.079
O	0.155 ± 0.000	0.007 ± 0.046	0.124 ± 0.044	0.154 ± 0.046
P	-0.019 ± 0.000	-0.122 ± 0.025	-0.010 ± 0.041	-0.073 ± 0.081
Q	0.105 ± 0.000	0.162 ± 0.024	0.065 ± 0.082	0.087 ± 0.077
R	0.100 ± 0.000	0.141 ± 0.049	0.158 ± 0.051	0.155 ± 0.027
S	0.067 ± 0.000	0.159 ± 0.028	0.193 ± 0.073	0.202 ± 0.015
T	0.014 ± 0.000	-0.029 ± 0.048	0.077 ± 0.037	0.100 ± 0.028
U	0.162 ± 0.000	0.094 ± 0.120	0.075 ± 0.073	0.136 ± 0.035
V	-0.063 ± 0.000	-0.063 ± 0.061	-0.018 ± 0.022	-0.073 ± 0.055
W	0.021 ± 0.000	0.097 ± 0.076	-0.037 ± 0.081	0.127 ± 0.073
X	0.110 ± 0.000	0.109 ± 0.054	0.146 ± 0.055	0.140 ± 0.025
Y	0.156 ± 0.000	0.192 ± 0.028	0.128 ± 0.055	0.177 ± 0.023
Z	0.090 ± 0.000	0.080 ± 0.034	0.104 ± 0.040	-0.004 ± 0.055

mitigating the overconfidence issue commonly found in modern neural networks. Unlike the original FSL implementation, our proposed Multiple FSL generates each label's output logit based on a distinct feature weighting vector. To empirically assess whether these label-wise logits remain well-calibrated despite originating from independent propagations (Eq. 7), we measured the top-1 Expected Calibration Error (ECE) [42] of the predicted-class confidence against actual accuracy for five models: the baseline neural network, LassoNet, TabNet, FSL, and Multiple FSL. The baseline neural network is identical to the one used internally by FSL and Multiple FSL, but without any feature gating. The original FSL, which relies on a single shared feature weighting vector to generate predictions for all classes, is theoretically less prone to producing miscalibrated logits. All five models were evaluated across the same nine benchmark datasets used throughout this work (Table 15). The baseline neural network yielded an average ECE of 0.084 across the nine datasets, noticeably higher than every gated model we evaluated: FSL (0.054), LassoNet (0.066), TabNet (0.073), and Multiple FSL (0.042). Notably, Multiple FSL achieved the lowest overall mean ECE and secured the best individual calibration on four of the nine datasets (SynthB, Liver, Breast, and Isolet), while tying for the lowest score on XOR. Even in datasets where it did not achieve the absolute lowest ECE, the performance gap was minimal. On Letter Recognition, it trailed the baseline neural network by a negligible margin of 0.001. On Colorectal, the smallest dataset in this comparison with only 86 samples and 4 classes, it trailed TabNet by 0.027, a difference well within the variance observed among the other models on that same dataset (0.129–0.190). Table 9 in the **Supplementary Material** presents the complete results, including comparisons with other embedded feature weighting methods. While Multiple FSL does not always achieve the absolute lowest average across all datasets, statistical analysis over the 11 executions reveals no significant difference between its performance and that of the top-performing models. If assembling label-wise

Table 15

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. For a complete comparison with additional baselines, please refer to the Table 9 of **Supplementary Material**. The lowest average is the bold. Best results with statistical significance ($p < 0.01$) according to in Kruskal-Wallis and Dunn's tests are highlighted in Orange.

Dataset	Baseline	LassoNet	TabNet	FSL	Multiple FSL
XOR	0.078 ± 0.051	0.003 ± 0.001	0.067 ± 0.028	0.003 ± 0.007	0.003 ± 0.006
SynthA	0.086 ± 0.027	0.044 ± 0.017	0.101 ± 0.025	0.073 ± 0.010	0.062 ± 0.010
SynthB	0.090 ± 0.038	0.087 ± 0.067	0.067 ± 0.018	0.037 ± 0.011	0.029 ± 0.005
SynthC	0.076 ± 0.032	0.041 ± 0.013	0.054 ± 0.020	0.086 ± 0.007	0.066 ± 0.016
Liver	0.094 ± 0.142	0.050 ± 0.074	0.111 ± 0.074	0.033 ± 0.080	0.023 ± 0.053
Colorectal	0.172 ± 0.080	0.190 ± 0.046	0.109 ± 0.091	0.129 ± 0.064	0.136 ± 0.079
Breast	0.071 ± 0.032	0.138 ± 0.044	0.117 ± 0.024	0.034 ± 0.022	0.033 ± 0.030
Isolet	0.079 ± 0.019	0.027 ± 0.007	0.023 ± 0.004	0.079 ± 0.014	0.016 ± 0.005
Letter R.	0.010 ± 0.002	0.011 ± 0.002	0.012 ± 0.003	0.011 ± 0.002	0.011 ± 0.002

Table 16

Colorectal dataset HD and 2D silhouette coefficients with feature weights related to label 1.

Averages were calculated over 11 executions. "None" represents the t-SNE without using weighted features, which is a fixed value. Averages were calculated from the weighted t-SNE results given the feature weights generated by each algorithm across 11 different executions. A one-vs-all (OVA) strategy was adopted, where all labels other than label 1 were converted into a single label. In bold is the highest average. Best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis and Dunn's tests are highlighted in Orange.

Weight source	HD silhouette	2D silhouette
None	.024	.090
Lasso	.024±.000	.060±.032
LIME	.024±.000	.036±.030
DeepSHAP	.048±.008	.141±.042
Multiple FSL	.143±.025	.446±.027

Table 17

PIFS comparison between Multiple FSL, LassoNet, TabNet and SAGE with SynthB dataset. The measure considers a subset of 30 features, which is the number of informative features in SynthB dataset. SAGE is the only method unable to define label-specific weights. In bold is the highest average for each dataset and label. The best results with statistical significance ($p < 0.01$) according to the Kruskal-Wallis and Dunn's tests are highlighted in Orange.

Algorithm	General	Label 0	Label 1	Label 2
LassoNet	0.948±.031	0.936±.067	1.0±.000	0.882±.098
TabNet	0.530±.071	0.518±.147	0.591±.130	0.509±.138
SAGE	0.951±.035	–	–	–
Multiple FSL	0.921±.040	0.982±.040	1.0±.000	1.0±.000

logits from independent feature-weighting layers introduced a systematic scale mismatch, Multiple FSL would be expected to exhibit poorer calibration than both the non-gated baseline and models that derive all logits from a single shared layer, such as FSL. Instead, it matches or outperforms them on average. Its rare instances of underperformance are minor and more plausibly attributed to the affected dataset's sample size rather than the multi-propagation mechanism itself. Consequently, we did not apply temperature scaling or a shared cross-label normalization to Multiple FSL, as the empirical evidence strongly indicates that its per-label logits are natively well-calibrated relative to comparable single-propagation alternatives.

Fig. 20 presents the weighted t-SNE visualization using general and label-specific feature weights generated by Multiple FSL for each label available on Breast dataset. A substantial improvement is observed in the visualization of each target label, accompanied by a deterioration in the others, indicating that not only were the label-specific features effectively highlighted, but features relevant to other labels were also suppressed, confirming the ability of Multiple FSL to identify label-specific feature weights. More experiments are presented in Figs. 12–13 and Tables 5–7 of **Supplementary Material**.

5.5. Results

To improve both the interpretability and performance of neural networks, we propose an extension to FSL that introduces label-specific feature weighting. The extension was evaluated on synthetic and real-world datasets to assess its ability to: (1) assign general feature weights, as in the original FSL, to identify the most relevant features; (2) assign label-specific weights to prioritize features for individual labels; and (3) examine the relationship between the feature relevance assigned by the algorithm and the resulting model prediction performance. Initial experiments on the XOR and SynthA datasets validated Multiple FSL's ability to assess general feature relevance. Subsequently, the SynthB and SynthC datasets were employed to assess the algorithm's capacity to discriminate between features relevant to specific labels. Through the application of feature erasure, weighted t-SNE, and an SVC model trained with subsets of the highest weighted features on three microarray datasets, we validate the effectiveness of Multiple FSL in handling high-dimensional real-world datasets with limited sample sizes. Compared to existing algorithms, Multiple FSL performs similarly for general relevant features while demonstrating a superior ability to capture label-specific feature relationships, mainly on high-dimensional datasets with a small sample size.

6. Conclusions

The present work has the objective of improving Deep Neural Network interpretability for multi-class data by introducing a new embedded feature weighting algorithm. While post-hoc methods can assign label-specific weights, embedded approaches use the internal mechanisms of neural networks during the training phase. However, most existing embedded methods lack label-specific relevance, limiting interpretability in multi-class scenarios. To address this, we posed the research question: "Can an embedded FSL be capable of learning label-specific feature relevance?" To answer it, we proposed four FSL extensions and identified Multiple FSL as the most promising. This novel embedded feature weighting algorithm builds upon FSL and adds label-specific selection layers, enabling identification of relevant features per label. To address the complementary question, "How to evaluate a label-specific feature weighting algorithm?", we applied different methodologies to evaluate the identification of relevant features and the assignment of higher weights to them. Synthetic datasets demonstrated that the algorithm assigns high weights to relevant features and low weights to noisy features, distinguishing between general and label-specific relevance. In real-world data, weighted t-SNE and SVC models highlighted the algorithm's ability to define label-specific feature weighting and feature erasure confirmed that the predictor relies most on top-weighted features. Our results demonstrated that Multiple FSL is capable of improving the model interpretability by indicating which features impact each label.

Finally, to address the question, "How does this approach compare to other algorithms?", we conducted comparative analyses with multiple established methods. Our findings demonstrate that Multiple FSL exhibits competitive performance in defining generally relevant features and a superior performance in identifying label-specific feature relevance. In contrast to post-hoc explainability techniques such as LIME, SHAP and SAGE, the proposed method bypasses significant computational overhead by directly aligning feature weights with the model's predictions at the cost of a linear increase in memory usage. This intrinsic efficiency makes it highly suitable for demanding real-world applications.

It is also important to identify the limitations of the proposed work. The proposed Multiple FSL algorithm was specifically developed for, and is currently limited to, tabular datasets due to its reliance on fixed feature semantics. The method operates by learning global feature weights that are applied uniformly across the entire dataset. This mechanism inherently requires that the meaning of each individual feature remains constant across all samples. In unstructured data such as images, however, individual features (pixels) lack a fixed semantic meaning; a specific pixel's relevance changes entirely from one image to another and must be interpreted within the local context of that specific sample. Furthermore, because Multiple FSL optimizes these global weights to reduce or explicitly disable the impact of features it considers irrelevant, applying it to image data would permanently blind the neural network to specific spatial locations. This global suppression would severely restrict the network's flexibility to detect varying visual patterns.

Like most feature weighting methods, Multiple FSL depends on the predictive capacity of the underlying predictive model. When the model performs poorly, the identified features may not reflect true relevance to the real-world problem, such as true biomarkers in microarray datasets. Instead, the model might be relying on illegitimate shortcuts that lead to overfitting. However, isolating the features the model primarily relies on in these situations remains valuable, as it allows researchers to detect and mitigate the specific patterns causing the overfitting. Multiple FSL is designed for easy integration with any neural network model. For future work, this approach could be explored in two distinct scenarios: (1) applying a trained Multiple FSL to an untrained model to enhance performance and reinforce relevant labels; and (2) attaching an untrained Multiple FSL to a pre-trained model to learn relevant features. This second technique has already been successfully implemented using the original FSL [43]. Our experiments demonstrated the effectiveness of Multiple FSL in identifying label-specific features across diverse dataset types, including gene expression data. As a next step, these identified features could be analyzed from a biological perspective to validate them against known biomarkers and investigate potential novel ones. Additionally, future work could explore the integration of Multiple FSL with other neural network architectures, such as CNNs and Transformers.

CRedit authorship contribution statement

João Pedro Silveira e Silva: Writing – review & editing, Writing – original draft, Visualization, Software, Investigation. **Márcio Dorn:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition, Conceptualization. **Bruno Iochins Grisci:** Writing – review & editing, Resources, Project administration, Methodology, Funding acquisition, Conceptualization.

Funding

This research was funded by Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) [24/2551-0001392-0; 23/2551-0001894-2; 19/2551-0001906-8; 24/2551-0000725-3; 25/2551-0002116-2]. This work was supported by the [Serrapilheira Institute](#) (grant number [Serra - R-2501-51351](#)). This study was financed in part by the Coordination for the Improvement of Higher Education Personnel – Brazil (CAPES) – Finance Code 001.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Bruno Iochins Grisci reports that financial support was provided by Foundation for Research Support of the State of Rio Grande do Sul. Marcio Dorn reports that financial support was provided by Foundation for Research Support of the State of Rio Grande do Sul. Bruno Iochins Grisci reports that financial support was provided by Serrapilheira Institute. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary data for this article can be found online at doi: [10.1016/j.ins.2026.124273](https://doi.org/10.1016/j.ins.2026.124273).

Data availability

Paper source code is available at <https://github.com/JPedroSilveira/label-specific-fsl>. **Weighted t-SNE [15]** is available at https://github.com/sbcb/weighted_tSNE. The **Feature selection analysis framework [11]** is available at <https://github.com/sbcb/GenExpFS>. **CuMiDa datasets [39]** can be found at <https://sbcb.inf.ufrgs.br/cumida>. **Synthetic datasets** are available within the source code.

References

- [1] R. Wason, Deep learning: evolution and expansion, *Cogn. Syst. Res.* 52 (2018) 701–708. ISSN 1389-0417, <https://doi.org/10.1016/j.cogsys.2018.08.023>
- [2] R. Miotto, F. Wang, S. Wang, X. Jiang, J.T. Dudley, Deep learning for healthcare: review, opportunities and challenges, *Brief. Bioinform.* 19 (6) (05 2017) 1236–1246. ISSN 1477-4054, <https://doi.org/10.1093/bib/bbx044>
- [3] S.J.D. Prince, *Understanding Deep Learning*, The MIT Press, Paris, France, 2023. ISBN 9780262048644. <http://udlbook.com>.
- [4] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, F.A. Wichmann, Shortcut learning in deep neural networks, *Nat. Mach. Intell.* 2 (11) (2020) 665–673, <https://doi.org/10.1038/s42256-020-00257-z>
- [5] E. Bayamlioglu, The right to contest automated decisions under the general data protection regulation: beyond the so-called right to explanation, *Regulation & Governance* 16 (4) (2022) 1058–1078, <https://doi.org/10.1111/rego.12391>
- [6] C. Molnar, G. Casalicchio, B. Bischl, Interpretable machine learning – a brief history, state-of-the-art and challenges, in: *ECML PKDD 2020 Workshops*, Springer International Publishing, Cham, 2020, pp. 417–431, ISBN 978-3-030-65965-3, https://doi.org/10.1007/978-3-030-65965-3_28
- [7] L. Grinsztajn, E. Oyallon, G. Varoquaux, Why do tree-based models still outperform deep learning on typical tabular data? in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), *Advances in Neural Information Processing Systems*, vol. 35, Curran Associates, Inc, 2022, pp. 507–520, https://proceedings.neurips.cc/paper_files/paper/2022/file/0378c7692da36807bdec87ab043cdadc-Paper-Datasets_and_Benchmarks.pdf.
- [8] B.I. Grisci, M.J. Krause, M. Dorn, Relevance aggregation for neural networks interpretability and knowledge discovery on tabular data, *Inf. Sci.* 559 (2021) 111–129. ISSN 0020-0255, <https://doi.org/10.1016/j.ins.2021.01.052>
- [9] J. Figueroa Barraza, E. López Droguett, M.R. Martins, Towards interpretable deep learning: a feature selection framework for prognostics and health management using deep neural networks, *Sensors* 21 (17) (2021). ISSN 1424-8220, <https://doi.org/10.3390/s21175888>
- [10] W.J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, Y. Bin, Definitions, methods, and applications in interpretable machine learning, *Proc. Natl. Acad. Sci.* 116 (44) (2019) 22071–22080, <https://doi.org/10.1073/pnas.1900654116>
- [11] M.C. Barbieri, B.I. Grisci, M. Dorn, Analysis and comparison of feature selection methods towards performance and stability, *Expert Syst. Appl.* 249 (2024) 123667. ISSN 0957-4174, <https://doi.org/10.1016/j.eswa.2024.123667>
- [12] M.A. Tahir, A. Bouridane, F. Kurugollu, Simultaneous feature selection and feature weighting using hybrid tabu search/k-nearest neighbor classifier, *Pattern Recognit. Lett.* 28 (4) (2007) 438–446. ISSN 0167-8655, <https://doi.org/10.1016/j.patrec.2006.08.016>
- [13] J. Miao, L. Niu, A survey on feature selection, *Procedia Comput. Sci.* 91 (2016) 919–926. ISSN 1877-0509, <https://doi.org/10.1016/j.procs.2016.07.111>, <https://www.sciencedirect.com/science/article/pii/S1877050916313047>. Promoting Business Analytics and Quantitative Management of Technology: 4th International Conference on Information Technology and Quantitative Management (ITQM 2016).
- [14] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, K.-R. Müller, Evaluating the visualization of what a deep neural network has learned, *IEEE Trans. Neural Netw. Learn. Syst.* 28 (11) (2017) 2660–2673, <https://doi.org/10.1109/TNNLS.2016.2599820>
- [15] B.I. Grisci, M. Inostroza-Ponta, M. Dorn, Assessing feature scorer results on high-dimensional datasets with t-SNE, *Neurocomputing* 652 (2025) 130561. ISSN 0925-2312, <https://doi.org/10.1016/j.neucom.2025.130561>
- [16] J. Xia, Y. Zhang, J. Song, Y. Chen, Y. Wang, S. Liu, Revisiting dimensionality reduction techniques for visual cluster analysis: an empirical study, *IEEE Trans. Vis. Comput. Graph.* 28 (1) (2022) 529–539, <https://doi.org/10.1109/TVCG.2021.3114694>
- [17] J.M.G. Junior, F.M. Lopes, Interpretability with relevance aggregation in neural networks for absenteeism prediction, in: 2022 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI), 2022, pp. 01–04, <https://doi.org/10.1109/BHI56158.2022.9926870>
- [18] Y. Zhao, Z. Huang, L. Gong, Y. Zhu, Y. Qiao, Y. Gao, Evaluating the impact of data transformation techniques on the performance and interpretability of software defect prediction models, *IET Softw.* 2023 (1) (2023) 6293074, <https://doi.org/10.1049/2023/6293074>
- [19] D. Kobak, P. Berens, The art of using t-SNE for single-cell transcriptomics, *Nat. Commun.* 10 (1) (2019) 5416, <https://doi.org/10.1038/s41467-019-13056-x>
- [20] W. Awada, T.M. Khoshgoftaar, D. Dittman, R. Wald, A. Napolitano, A review of the stability of feature selection techniques for bioinformatics data, in: 2012 IEEE 13th International Conference on Information Reuse & Integration (IRI), 2012, pp. 356–363, <https://doi.org/10.1109/IRI.2012.6303031>
- [21] M.T. Ribeiro, S. Singh, C. Guestrin, why should i trust you?: Explaining the predictions of any classifier, in: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. 1135–1144, <https://doi.org/10.1145/2939672.2939778>

- [22] D. Slack, S. Hilgard, E. Jia, S. Singh, H. Lakkaraju, Fooling lime and SHAP: adversarial attacks on post hoc explanation methods, in: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, AIES '20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371100. 180–186, <https://doi.org/10.1145/3375627.3375830>
- [23] J. Shin, Feasibility of local interpretable model-agnostic explanations (lime) algorithm as an effective and interpretable feature selection method: comparative fnirs study, *Biomed. Eng. Lett.* 13 (4) (2023) 689–703, <https://doi.org/10.1007/s13534-023-00291-x>
- [24] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, *Adv. Neural Inf. Process. Syst.* 30 (2017). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf
- [25] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, *PLoS One* 10 (7) (07 2015) 1–46, <https://doi.org/10.1371/journal.pone.0130140>
- [26] S. Liang, Y. Zhang, K. Zheng, Y. Bai, Featurex: an explainable feature selection for deep learning, *Expert Syst. Appl.* 282 (127675) (2025). ISSN 0957-4174, <https://doi.org/10.1016/j.eswa.2025.127675>
- [27] I. Covert, S.M. Lundberg, S.-I. Lee, Understanding global feature contributions with additive importance measures, in: H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, H. Lin (Eds.), *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc, 2020, pp. 17212–17223, https://proceedings.neurips.cc/paper_files/paper/2020/file/c7bf0b7c1a86d5eb3be2c722cf2cf746-Paper.pdf
- [28] D. Singh, H. Climente-Gonzalez, M. Petrovich, E. Kawakami, M. Yamada, Fsnets: feature selection network on high-dimensional biological data, in: 2023 International Joint Conference on Neural Networks (IJCNN), 2023, pp. 1–9, <https://doi.org/10.1109/IJCNN54540.2023.10191985>
- [29] N. Gui, G. Danni, H. Ziyin, Afs: an attention-based mechanism for supervised feature selection, *Proc. AAAI Conf. Artif. Intell.* 33 (1) (Jul 2019) 3705–3713, <https://doi.org/10.1609/aaai.v33i01.33013705>
- [30] I. Lemhadri, F. Ruan, R. Tibshirani, LassoNet: neural networks with feature sparsity, in: A. Banerjee, K. Fukumizu (Eds.), *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, Volume 130 of *Proceedings of Machine Learning Research*, Apr 2021, pp. 10–18. PMLR, 13–15 <https://proceedings.mlr.press/v130/lemhadri21a.html>
- [31] S. Arik, T. Pfister, Tabnet: attentive interpretable tabular learning, *Proc. AAAI Conf. Artif. Intell.* 35 (8) (May 2021) 6679–6687, <https://doi.org/10.1609/aaai.v35i8.16826>
- [32] J.S. Aguilar-Ruiz, Class-specific feature selection for enhancing explainability in ensemble classification models, *Int. J. Data Sci. Anal.* (2024) 1–10, <https://doi.org/10.1007/s41060-024-00694-8>
- [33] L. Yangding, H. Xie, J. Dai, Label-specific multilabel feature selection based on fuzzy implication granularity information, *IEEE Trans. Fuzzy Syst.* 33 (3) (2025) 959–970, <https://doi.org/10.1109/TFUZZ.2024.3502073>
- [34] H. Kaiming, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778, <https://doi.org/10.1109/CVPR.2016.90>
- [35] K. Sun, S.-H. Huang, D.S.-H. Wong, S.-S. Jang, Design and application of a variable selection method for multilayer perceptron neural network with lasso, *IEEE Trans. Neural Netw. Learn. Syst.* 28 (6) (2017) 1386–1396, <https://doi.org/10.1109/TNNLS.2016.2542866>
- [36] Z. Zhu, Y.-S. Ong, J.M. Zurada, Identification of full and partial class relevant genes, *IEEE/ACM Trans. Comput. Biol. Bioinform.* 7 (2) (2010) 263–277, <https://doi.org/10.1109/TCBB.2008.105>
- [37] F. Kamalov, H. Suliman, A.K. Cherukuri, Synthetic data for feature selection, in: L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, J.M. Zurada (Eds.), *Artificial Intelligence and Soft Computing*, Springer, Cham Nature Switzerland, 2023, pp. 353–365, ISBN 978-3-031-42508-0, https://doi.org/10.1007/978-3-031-42508-0_32
- [38] I. Guyon, Madelon. UCI Machine Learning Repository, 2004. <https://doi.org/10.24432/C5602H>
- [39] B. César Feltes, E.B. Chandelier, B.I. Grisci, M. Dorn, Cumida: an extensively curated microarray database for benchmarking and testing of machine learning approaches in cancer research, *J. Comput. Biol.* 26 (4) (2019) 376–386, <https://doi.org/10.1089/cmb.2018.0238>
- [40] D. Slate, Letter recognition. UCI Machine Learning Repository, 1991. <https://doi.org/10.24432/C52P40>
- [41] R. Cole, M. Fandy, ISOLET UCI Machine Learning Repository, 1991. <https://doi.org/10.24432/C51G69>
- [42] C. Guo, G. Pleiss, Y. Sun, K.Q. Weinberger, On calibration of modern neural networks, in: D. Precup, Y.W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning*, Volume 70 of *Proceedings of Machine Learning Research*, Aug 2017, pp. 1321–1330. PMLR, 06–11 <https://proceedings.mlr.press/v70/guo17a.html>
- [43] J. Kenji Suwa, J. Pedro Silveira e Silva, B.I. Grisci, Post-hoc feature selection layer for neural networks interpretability, in: *International Joint Conference on Neural Networks (IJCNN)*, 2026, https://linkings.s3.amazonaws.com/organizations/WCCI/wcci2026/submissions/stype114/khrGU-ijcnn_pap3544s2.pdf