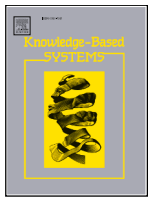
GLEm-Net: Unified framework for data reduction with categorical and numerical features

Knowledge-Based Systems 334 (2026) 115049

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



GLEm-Net: Unified framework for data reduction with categorical and numerical features

Francesco De Santis, Danilo Giordano *, Marco Mellia

Politecnico di Torino, Turin, Italy

ARTICLE INFO

Keywords:
Feature selection
Neural network
Categorical features embedding
Mixed data type

ABSTRACT

In an era of effortless data collection, the impact of machine learning — especially neural networks (NNs) — is undeniable. As datasets grow in size and complexity, efficiently handling mixed data types, including categorical and numerical features, becomes critical. Feature encoding and selection play a key role in improving NN performance, efficiency, interpretability, and generalisation. This paper presents GLEm-Net (Grouped Lasso with Embeddings Network), a novel NN-based approach that seamlessly integrates feature encoding and selection directly into the training process. GLEm-Net uses embedding layers to process categorical features with high cardinality, simplifying the model and improving generalisation. By extending the grouped Lasso regularisation to explicitly consider categorical features, GLEm-Net automatically identifies the most relevant features during training and returns them to the analyst.

We evaluate GLEm-Net on open and proprietary industry datasets and compare it to state-of-the-art feature selection methodologies. Results show that GLEm-Net adapts to each dataset by allowing the NN to directly select subsets of most important features, offering on par performance with the best state-of-the-art feature selection methods, while eliminating the need for the external feature encoding and selection steps that are now incorporated in the NN training stage.

1. Introduction

In today's world where data collection is effortless, the transformative potential of machine learning techniques, and neural networks in particular, is evident. Focusing on supervised machine learning, the need for efficient methodologies that can handle both categorical and numerical data types is becoming increasingly urgent as datasets become larger and more complex. Indeed, categorical and numerical data are ubiquitous in real-world data, and their seamless integration in the data encoding and feature selection steps is crucial for the development of effective, scalable and interpretable machine learning models. Feature selection has long been recognised as an essential tool for unlocking the potential of big data [1,2]. By reducing the dimensionality of the data into more manageable and informative subsets, these methodologies improve model performance and computational efficiency and contribute to its interpretability and generalisability.

In this paper, we present GLEm-Net (Grouped Lasso with Embeddings Network), a robust approach that targets Neural Networks (NNs) models and performs the encoding, dimensionality reduction and feature selection steps directly during the NN training phase. It builds on the previous methodology presented in [3] by providing a comprehensive analysis of the methodology and validating it on several additional datasets. Firstly, GLEm-Net takes advantage of embedding layers to effectively manage high cardinality categorical features and reduce their dimension by compressing information into a smaller space. Secondly, it seamlessly integrates the process of selecting the most important features into the neural network architecture while training the machine learning model. To this end, we build on the regularisation technique *Grouped Lasso*, where the concept of lasso regularisation was originally introduced to perform feature selection only for numerical features, see Zhang et al. [4], Lemhadri et al. [5].

We perform a comprehensive evaluation of GLEm-Net on various open and proprietary datasets collected in real-world environments and compare its performance with state-of-the-art (SOTA) feature selection methodologies. Our results show that GLEm-Net achieves performance comparable to the best SOTA methods while providing a unified framework that integrates categorical feature encoding and feature selection directly into the training phase of the neural network, eliminating the need to perform these steps separately. In contrast to existing methods GLEm-Net requires only a single training run to encode and select the

* Corresponding author.
E-mail addresses: francesco.desantis@polito.it (F. De Santis), danilo.giordano@polito.it (D. Giordano), marco.mellia@polito.it (M. Mellia).

https://doi.org/10.1016/j.knosys.2025.115049
Received 1 October 2024; Received in revised form 25 August 2025; Accepted 2 December 2025
Available online 15 December 2025

most important features, and to train the NN model. To facilitate reproducibility and further research, we are releasing the implementation of GLEm-Net as open source.[1]

The rest of the paper is organised as follows. Section 2 provides an overview of related work, while Sections 3 and 4 breaks down the methodology of GLEm-Net. In Section 5 we present the datasets, while in Section 6 we describe the experimental setup and in Section 7 we present our results. Finally, in Section 8 we discuss the limitations and capabilities and conclude the paper with a conclusion and possible future directions.

2. Related work

Data reduction techniques such as sample reduction and feature selection are crucial for managing big datasets and subsequently using them for supervised machine learning tasks. Sample reduction helps to reduce the size of the dataset by removing entire samples (i.e., rows). Feature selection focuses on identifying the most informative features (i.e., columns) and retaining all samples. In both scenarios, the goal is to reduce the size of the dataset while preserving the information it contains [6]. In this paper, we focus on feature encoding and selection as they improve interpretability by focusing on the most relevant features while mitigating the risk of overfitting [7]. Feature selection methods can be broadly categorized into filter, embedded, and wrapper approaches. To facilitate a more thorough comparison, Table 1 presents an updated overview of recent methods, emphasizing their theoretical foundations, implementation strategies, and highlighting key strengths and limitations.

Filter methods rank the features based on their correlation with the target variable. The authors of [8] introduce the mRMR method. mRMR is a supervised method for ranking the features to maximise their relevance to the prediction of the target variable while minimising their redundancy by assessing the mutual information between feature pairs. By selecting a subset of the features from this ranking, mRMR has proven to be successful for both continuous [9] and categorical features [8], even in complex contexts such as federated learning [10]. Herrera-Semenets et al. [11] present the Multi Measure Feature Selection Algorithm (MMFSA) algorithm, a supervised feature selection method that ranks features separately according to Information Gain, Chi-square, and ReliefF. It then calculates the average value for each metric and selects the features for each set that exid this average value, and finally merges all sets to find the best subset. The approach has shown improved classification performance in intrusion detection settings, although it relies on heuristic thresholds and can lead to data redundancy: each metric is run independently, so the union of their results does not take into account the correlations among the selected features. Similarly, Herrera-Semenets et al. [11] present the Weighted Fusion-Based Feature Selection (WFFS) algorithm, an ensemble approach in which the importance of features is calculated using different filtering and wrapping approaches. MMFA then combines these via a weighted voting scheme to identify the most informative features. The approach shows improved performance on traffic accident data, although it treats categorical features as numerical and imposes an artificial ordering that may not always be appropriate.

In embedding solutions, the best subset of features is determined as a by-product of the classification/regression process. Decision Trees and Random Forest [12,13] are two seminal works in this direction. Decision trees make it possible to determine the importance of features directly by analysing the generated tree for classification, while Random Forests rank features according to their importance in the classification process. Both methods are only suitable for categorical features to a limited extent, depending on the pre-processing. To overcome this limitation, Dorogush et al. propose Catboost [14], a method from the Gradient Boosted Trees family specifically designed for managing categorical features with ordered target encoding. Similar to other tree-based models [15], Caboost does not explicitly consider redundancy during the calculation of feature importance, which can lead to redundant feature selection.

In wrapper approaches, the performance of the classifier guides the selection of the best subset of features. BORUTA [16] has proven to be a valuable approach [17]. BORUTA uses a two-step approach in which the dataset is first duplicated and the values of the individual features are randomly shuffled. Then, an iterative process is applied in which a Random Forest model is fitted to compare the importance of the original feature and its shuffled version. Since BORUTA is based on the Random Forest classifier, the handling of categorical features that depend on the pre-processing decisions is limited. Another popular approach, RFE-SVM [18], applies recursive feature elimination by iteratively removing the features with the smallest weights in the SVM model until the desired number of genes is reached. This method has been shown to be effective in selecting informative gene subsets while improving model performance. However, iterative feature removal makes the RFE-SVM very computationally expensive and also does not provide a native way to process categorical features. Other recent solutions use optimisation strategies to find the best subset of features. Jun Ma et al. [19] propose a metaheuristic wrapper method, Multi-Strategy binary ensemble Hunger Games Search (MS-bHGS), for feature selection. MS-bHGS uses multiple strategies within an optimisation algorithm to select a subset of features, using k-NN for classification, and shows superior classification performance with the selected features. As a population-based metaheuristic, it requires extensive fitness evaluations for each candidate subset, which leads to high computational cost in feature selection. Lakshme and Kumar [20] propose a hybrid filter-wrapper approach for feature selection. They first apply filtering methods to reduce the feature set and then refine it using the MOBHHO algorithm that balances feature reduction and classification performance. LightGBM is used for classification and achieves high accuracy on microarray data. However, like MS-bHGS, MOBHHO also requires high computational costs due to the repeated fitness evaluations. Habib et al. [21] propose a wrapper approach that uses three biologically inspired optimisation algorithms to jointly select features and tune SVM parameters. Experiments on voice disorder detection show strong predictive performance, with SHAP analysis confirming the validity of the selected features. The methodology was evaluated on voice disorder datasets only.

Differently from these works, which often focus on specific data types or decouple the feature selection and classification phases, our aim is to design a unified Neural Network framework that integrates both tasks in an end-to-end manner, starting from the raw input data and directly optimising the classification performance.

In this direction, the authors in [22] propose a feature selection approach that combines the use of an MLP with the concept of “prominence” to identify a subset of relevant, non-redundant features for supervised pattern classification. The methodology is validated on nine small UCI datasets containing only numerical features. Lu et al. [23] propose DeepPINK, a neural network-based feature selection method that controls the false discovery rate. To this end, the model-X knockoffs framework - where synthetic “knockoff” features are created to mimic the correlation structure of the real features while being independent of the target - is integrated into an MLP by pairing each feature with its knockoff and learning to distinguish them. The method improves interpretability and selection accuracy on real and simulated data, but is limited to continuous features and depends on generating valid knockoffs, which can be difficult with mixed data types. Atashgahi et al. [24] propose NeuroFS, a neural feature selection method that uses sparse training with dynamic pruning and regrowth of input connections. Like GLEm-Net, it takes an embedded approach by selecting features during training. While it shows competitive performance in identifying single relevant features, it lacks native support for grouped or categorical inputs, limiting its applicability to mixed data types. Similar to us, Saha

[1] https://github.com/francescoTheSantis/GLEm-Net

and Pal [25] propose a neural feature selection method based on group-feature modulators that apply a redundancy-aware penalty to suppress correlated feature groups. Like GLEm-Net, it performs embedded group-wise selection with structured regularisation. However, it requires predefined feature groups and does not natively support mixed data types or implicit categorical groupings. Wydmański and Śmieja [26] propose GFSNetwork, a neural feature selection method that leverages Gumbel-Sigmoid sampling to learn binary feature masks in a differentiable manner. The approach allows the network to automatically determine the number of selected features during training and maintains constant computational cost regardless of input dimensionality. While GFSNetwork focuses on learning stochastic masks through reparameterised sampling, GLEm-Net adopts a deterministic regularisation-based strategy that integrates feature selection into the loss function via structured sparsity penalties. This results in different optimisation dynamics: GFSNetwork relies on sampling and temperature annealing, whereas GLEm-Net enforces sparsity through continuous penalties applied to model weights. Zimmer et al. [27] propose EntryPrune, a neural feature selection method based on iterative pruning and regrowth of input features using relative gradient change scores. In contrast to the regularisation-based approach of GLEm-Net, EntryPrune selects features through dynamic scoring and stochastic updates during training. This leads to a more adaptive but less structured selection process. The authors in [28] present a prediction model to extract the most promising subset of features and a feedforward neural network to optimise the subsets of features and fine-tune them using a genetic algorithm. The authors in [29] induce a neural network to perform feature selection using an iterative two-stage strategy. [5] propose LassoNet, a method for integrating feature selection mechanisms into a multilayer perceptron (MLP). LassoNet forces the MLP to eliminate irrelevant features by introducing a residual layer that connects the input layer directly to the loss function. Another approach to integrate feature selection into a neural network is proposed by [30]. More specifically, they used L0 regularisation to reduce the contribution of numerical input features. The authors in [4,31] present a different approach to integrate feature selection in this context. They use an MLP with a single hidden layer for feature selection by incorporating a loss function that contains a regularisation term based on the grouped Lasso penalty function.

In Solorio-Fernández et al. [32] the authors summarise possible strategies for dealing with mixed data types: encoding of categorical features into numerical ones ([33–35]), treating individual features, and applying a unified principle-based metric tailored to each data type ([36–38]). However, each of these methods also has its disadvantages. The first approach may result in the original variable having no inherent order or spacing. The second strategy uses different metrics for different data types, which makes it difficult to compare the results. In the third strategy, a uniform metric is used, but its calculation is customised for different data types, which causes similar difficulties when comparing features. Therefore, the development of methods to directly use a mixture of categorical and numerical features remains a challenge for the feature selection research community. Regarding feature selection for mixed data types, Ghouzam et al. [39] provide a comprehensive survey of filter, wrapper and embedded approaches. They highlight how most existing methods are based on shallow models and only a few neural network-based solutions explicitly consider heterogeneous feature types.

Our goal is to fill this gap by automatically process both categorical and numerical features and make the neural network learn the best representation for both types of features. To compare our feature selection capability with existing methods and show that we can achieve comparable or slightly better performance with a single framework, we apply it to tabular data in the context of a Feedforward Neural Network (FFNN). However, the framework could also be extended to other data types, such as time-series data, where many of the previously mentioned methods have significant limitations due to their reliance on manual feature engineering or assumptions tied to static input representations. In this direction, Dinh and Ho [40] propose CancelOut, a feature selection method for deep neural networks in which an L1 penalty is applied to the weights of the input layer to promote sparsity by zeroing irrelevant features. They evaluate the approach on small datasets for classification and regression tasks. While the CancelOut layer is effective, it relies on a standard L1 penalty that is computed separately for each input neuron without modelling grouped structures. GLEm-Net instead firstly applies a grouped regularisation term to better handle categorical features, and secondly explores different regularisation functions to evaluate their impact on feature selection. Rossi et al. [41] have developed a Deep Learning framework for dimensionality reduction and validated their results with cross-sectional data and time-series data. Similar to us, they use a layer between the input layer and the hidden layers, namely a variationally explainable layer. Here, they map each input neuron to a neuron in the variational explainable layer, which is used for dimensionality reduction. They then modify the loss function to include a penalty value to weight the importance of feature reduction with respect to classification loss.

Unlike them, here we implement our reduction layer as well as our loss function to account for both numerical and categorical features by treating all input neurons of each categorical feature together and using a modified version of the grouped lasso penalty term. Subsequently, our embedding layer can be used to simplify the architecture of the neural network.

This paper extends our preliminary work [3] by providing an in-depth analysis of the methodology. To demonstrate the benefits of GLEm-Net we have extended our literature review to consider progress in the area of feature selection and validate our methodology against several publicly available datasets (Adult income [42], Sleep health and lifestyle [43], PHME data challenge [44], Secondary Mushroom [45] and KDD cup 1999 [46]) to show the generality of our approach. Finally, we consider more challenging dataset that comes from a real-world scenario and is characterised by several categorical features that, once encoded, make the number of input features explode, posing a challenge for neural networks and other models [47]. This is the only dataset we analysed in Santis et al. [3].

3. Methodology

Grouped Lasso is a regularisation technique used in machine learning and statistics to select groups of features [49]. This allows the model to focus on the most important features while penalising the use of less important inputs. Its application to neural networks enables the creation of predictive models and reduces feature dimensionality, improving model performance, stability and interpretability. The methodology originally proposed in Hastie et al. [49] cannot handle categorical features that frequently occur in real-world tabular datasets. To overcome this limitation, we propose here GLEm-Net, a framework that enables the effective processing of mixed-type features and adapts the Grouped Lasso regularisation technique to categorical features.

3.1. Background

The Least Absolute Shrinkage and Selection Operator (*Lasso*) is a regularisation technique used in linear regression and related models. It adds a penalty term to the loss function based on the absolute values of the model coefficients. This promotes sparsity and automatic feature selection and prevents overfitting of the model by limiting excessive increases in model coefficients and reducing the coefficients of variables with low significance. Thus, if a coefficient of the model assumes a value close to zero, the feature associated with this coefficient is considered useless for prediction.

Consider a multilayer perceptron (MLP), a feedforward artificial neural network, with p neurons in the input layer and k neurons in the first hidden layer. Each neuron x_i in the input layer is connected to the first

Table 1
Comparison of GLEm-Netwith existing feature selection methods.

	Feature Selection Mechanism	Mixed Data	Strengths	Limitations
Filtering				
mRMR [8]	Mutual information-based ranking	✓	Low complexity, interpretable	Can miss dependencies among multiple features
MMFSA [11]	Multi-criteria ranking	✓	Automatic cardinality selection, multi-creteria	Heuristic mean-score threshold, may introduce redundancy
WFFS [48]	Aggregated scores from multiple selectors + threshold	✓	Flexible, robust ensemble approach	Assumes numeric ordering for categorical features
Embedded				
Decision Trees/RF [12,13]	Importance derived from tree structure	✓	Interpretable, widely used	Limited capability in hading categorical features
Catboost [14]	Importance derived from tree structure	✓	Interpretable, natively manage categorical features	May select redundant features due to lack of redundancy control.
Wrapper				
BORUTA [16]	Iterative random forest-based importance testing	✓	Strong for tree-based models	Computationally expensive, does not natively handle categorical features
RFE-SVM [18]	Recursive elimination guided by SVM margin	✗	Effective SVM-based selection	Computetionally expensive, no support for categorical features
MS-bHGS [19]	Optimisation strategy + k-NN evaluation	✗	High classification performance	High computational cost due to the required extensive fitness evaluation
Lakshme & Kumar [20]	Multi-step filtering + MOBHHO-based optimisation	✗	Accuracy with reduced features	Manual threshold tuning
Habib et al. [21]	Bio-inspired optimisation + SVM tuning	✗	SHAP explainability, accurate	Validated only in the voice disorder scenario
Embedded Neural Network				
Gasca [22]	MLP with prominence	✗	Integrated into MLP training	Limited validation with no categorical features
LassoNet [5]	Residual input layer + sparsity penalty	✗	Integrated into MLP training	No categorical support
DeepPINK [23]	Knockoff-based paired input in MLP	✗	Controls false discovery rate, interpretable	Requires valid knockoffs, limited to continuous features
NeuroFS [24]	Sparse training with iterative pruning/regrowth	✗	Efficient, adaptive feature selection	No native support for grouped or categorical features
GroupFS [25]	Group-feature modulators + redundancy-aware penalty	✗	Group-wise selection with redundancy control	Requires predefined groups, limited support for mixed data
GFSNetwork [26]	Differentiable binary mask via Gumbel-Sigmoid sampling	✗	Automatic feature count selection, constant cost	Sampling-based, no structured sparsity control
EntryPrune [27]	Gradient-based iterative pruning and stochastic regrowth	✗	Lightweight, adaptive selection	Unstructured, stochastic feature dynamics
Grouped Lasso MLP [4]	Grouped regularisation in loss function	✗	Structure-aware selection	Categorical data unsupported
CancelOut [40]	L1 penalty on input-layer weights (CancelOut layer)	✓	Simple, effective feature pruning with theoretical support	No structured grouping; limited evaluation on small datasets
Rossi et al. [41]	Variational explainable layer + penalty loss	✗	Extends to time-series with Deep Learning architecture	Loss function does not consider the presence of categorical features
GLEm-Net	**Learned reduction layer + grouped Lasso penalty with Deep Learning architecture**	✓	**End-to-end, unified, supports high-cardinality categorical features**	**Cannot directly control the number of selected features**

hidden layer by a series of weighted connections $w_{i,j}$. In this architecture, using the Lasso regularisation term is not feasible because each input variable (i.e., each input neuron) is described by multiple parameters, i.e., all the weights connecting it to all the neurons in the second layer. To overcome this limitation, the *grouped Lasso* extends the Lasso regularisation term by considering the grouped sets of all coefficients $\{w_{i,j},\ j = 1, \dots, k\}$ as a penalty. In this case, the input variable x_i becomes useless for the prediction if its weights $w_i = (w_{i1}, \dots, w_{ik})^T$ have all components close to zero, i.e., $||w_i||_2 \to 0$.

Consider now to modify the *Loss function* L used in the training of the neural network as follows:

$$L = L_{std} + \underbrace{\lambda L_{GL}}_{\textit{regularisation term}} \tag{1}$$

The first term L_{std} represents a standard loss, such as the Cross-Entropy in the case of a classification task; it is used to reduce the error of the model for the task. The second term L_{GL} represents the penalty of the *Grouped Lasso* regularisation term, which is used to reduce the number of features [49]. We introduce a parameter $\lambda \geq 0$ that weights the importance of feature selection and model performance. If $\lambda = 0$, no feature selection is performed. The higher λ is, the more the model reduces the number of features, regardless of the performance of the model.

We define L_{GL} as follows:

$$L_{GL} = \sum_{i=1}^{p} h(||w_i||_2) \tag{2}$$

To consider all weights of a feature as a group, we calculate for each input neuron x_i, $i \in \{1.p\}$ the norm $||w_i||_2$ of its weight vector w_i. Then we sum all norms and multiply them by the term λ to calculate the

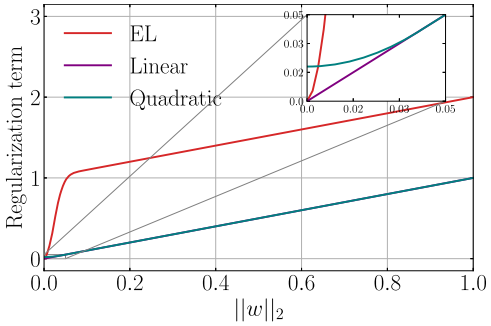


Fig. 1. Example of three regularisation functions.

regularisation term. If a feature x_i does not contribute to the performance improvement in L_{std} during model training, the norm of its vector w_i (and thus of all weights $w_{i,j}$) can be lowered to a value close to zero, so that this input neuron can be removed from the network (with all $w_{i,j} \to 0$). Unfortunately, using the norm as a regularisation term leads to a non-differentiable function at the origin due to a discontinuity that leads to possible numerical oscillations when $||w_i||_2$ is close to zero. As suggested by [4], the introduction of a *regularisation function* $h(\cdot)$ solves this problem by making the regularisation term differentiable at the origin and facilitating the convergence of the input vector norms.

3.2. Regularisation function h

The goal of the regularisation function is to streamline the feature selection process. To this end, we propose here a function $h(\cdot)$ that boosts the gradient of the regularisation term as the norm approaches zero by using the following exponential-linear (EL) function:

$$h(||w_i||_2)_{EL} = 1 - e^{\frac{-x^2}{\beta}} + \gamma x \tag{3}$$

Fig. 1 shows the function $h(.)_{EL}$ (red curve) with $\beta = 10^{-4}$ and $\gamma = 1$. For small values of $||w_i||_2$, the function shows exponential behaviour, with β adjusting the support of this exponential segment. The function then enters a linear phase in which the slope is equal to γ. The parameter γ provides an additional degree of freedom by allowing us to penalise the norms of those variables that the network considers less important.

The linear term of $h(\cdot)_{EL}$ would potentially introduce numerical oscillations for values of $||w_i||_2$ close to 0. To limit this, we freeze the weight vector $||w_i||_2$ when its norm is smaller than a threshold ϵ to stabilise the training.

To set β and γ, we study the initialisation weights to ensure that all weights fall within the linear range after initialisation. This allows all features to start from a neutral region characterised by a limited penalty gradient. ϵ, on the other hand, must be small enough to ensure that only the norms of a weight vector considered negligible are frozen during training. To compare the performance of $h(\cdot)_{EL}$, we compare it with two functions proposed in the past:

$$h(||w_i||_2)_{Linear} = ||w_i||_2 \tag{4}$$

$$h(||w_i||_2)_{Quadratic} = \begin{cases} ||w_i||_2, & ||w_i||_2 \geq \alpha, \\ \frac{||w_i||_2^2}{2\alpha} + \frac{\alpha}{2}, & ||w_i||_2 < \alpha. \end{cases} \tag{5}$$

We compare the regularisation function in Fig. 1. The quadratic function (Eq. (5)) was introduced by [4]. It consists of a quadratic term in the vicinity of zero and a linear term outside the vicinity. This function is differentiable at the origin, so that it should accelerate the convergence of the norms of the input vectors and avoid oscillations around the origin. The linear function (Eq. (4)) is instead the standard regularisation term without any modification. It was used by [50] and [29].

4. Incorporating categorical features in neural networks

Pre-processing categorical features is crucial when using machine learning and deep learning models to ensure that the information contained in these features is represented correctly, improving the model's ability to generalise and make accurate predictions. As authors in [51] noted, traditional categorical encoding methods, such as one-hot encoding, are inadequate for managing high-cardinality categorical features. Recently, more advanced encoding strategies, called categorical feature embedding, have shown comparable or even higher performance for categorical encoding [52,53]. We leverage these advances to develop a methodology that can efficiently process categorical features.

4.1. Embedding layers

An embedding layer consists of an additional set of fully connected neurons inserted between the input and the first hidden layer. Its goal is to transform input data into continuous vector representations [52], which enables efficient processing and improves generalisation for neural network tasks. In the case of a categorical variable with a large number of possible values, it helps to represent the possible values in a lower dimensional space and compress the information compared to a one-hot encoding method.

Fig. 2 shows an example of an MLP architecture with two input features: f_1, numeric, and f_2, categorical. f_1 serves as input for x_1, which is directly connected to the first hidden layer. For f_2, we first assign each of its values to an input neuron x_i $i \in [2, 5]$ using a one-hot encoding representation (4 neurons in this example). Then we connect each of the categorical neurons to the embedding layer e_2 of f_2, which has $R = |e_2|$ neurons ($R = 2$ in the example) to compress the categorical feature information by R. Then we connect the embedding layer e_2 to the first hidden layer of the network. We repeat this process for each categorical feature f_i for which we introduce a different embedding layer e_i. In Section 4.2 we describe how to set R in these layers.

We use the embedding neurons to first map the categorical variable into coordinates in latent space. These coordinates are then the input for the connections between the embedding and hidden layer. As they are part of the neural network, these weights can be learnt during the training process together with all other weights in the network.

Consider the neural network in Fig. 2. The weight matrix W of the input neurons of the first hidden layer is:

$$W = \begin{bmatrix} w_{(1,1)} & w_{(2,1)} & w_{(3,1)} \\ w_{(1,2)} & w_{(2,2)} & w_{(3,2)} \\ w_{(1,3)} & w_{(2,3)} & w_{(3,3)} \end{bmatrix}$$

The first column $w_{(1,*)}$ represents the numerical feature directly linked to f_1. The last two columns $w_{(2,*)}$ and $w_{(3,*)}$ refer to the embedding layer e_2 of the categorical feature f_2. Our goal is to reduce the weights of these connections as much as possible by applying the Grouped Lasso penalty function. On the one hand, for the numerical feature, we can directly apply the L2 norm of the vector $w_{1,*}$ to calculate the Grouped Lasso penalty function as in Eq. (2). On the other hand, this is not possible for the categorical feature as it is described by two vectors: $w_{(2,*)}$ and $w_{(3,*)}$. For a simple approach, one could use the Frobenius norm on the submatrix $W_2 = [w_{(*,2)}, w_{(*,3)}]$. However, it is important to note that the resulting value is not directly comparable to the L2 norm calculated for the numeric variable. Therefore, we consider the *expected value* of the norms of the weight vectors of the embedding layer. In this way, we can balance the importance of numerical and categorical features within the regularisation term.

In general, we consider n features that contain both numerical and categorical features. Each feature i is first associated with a set $l(i)$ of columns in the weight matrix W. For a numerical feature i, $|l(i)| = 1$, while for a categorical feature i, $|l(i)|$ corresponds to the number of neurons in the embedding layer e_i. As a result, the following equation

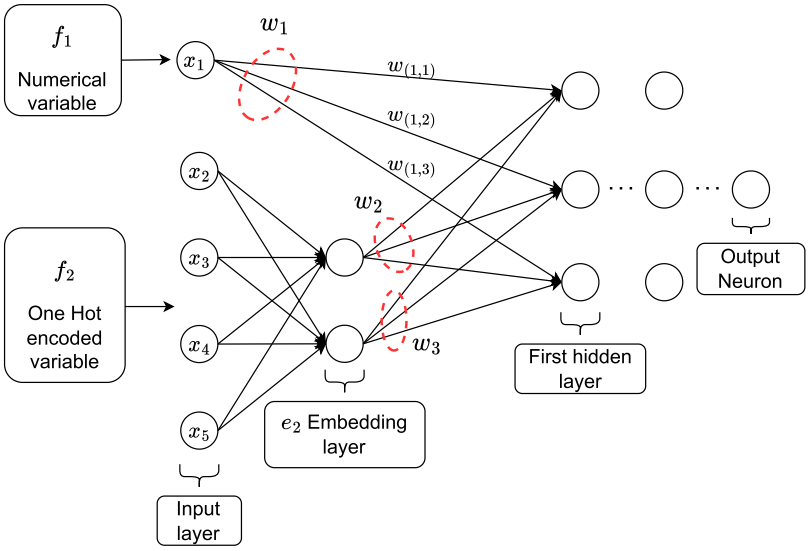


Fig. 2. Generic MLP with an embedding layer, having a categorical and a numerical variable in the input layer.

represents the generalised version of the loss function:

$$L_{GL} = \underbrace{\sum_{i=1}^{n} h\left(\frac{1}{|l(i)|}\sum_{j\in l(i)} ||w_j||_2\right)}_{\textit{regularisation term}} \tag{6}$$

L_{GL} extends the concept of Grouped Lasso penalty into a network that can handle both numeric and categorical variables. This loss L can be applied when the embedding layers are excluded and the regularisation is applied directly to the connections between the one-hot-encoded variables and the first hidden layer. However, it should be noted that in the case of a categorical feature with high cardinality and low information, this approach could cause the neural network to use an excessively large number of parameters, making model training more difficult.

4.2. Embedding size choice

We need to determine the number of neurons in the embedding layer. Recall that here we use embeddings for dimensionality reduction and compress the information in the case of high cardinality categorical features. Reducing the size of the vector used to represent these features also simplifies the architecture of the neural network by reducing the number of weights required. Suppose we have a categorical feature with a cardinality of C, an embedding layer with a dimension of E neurons, and the first hidden layer of the neural network with H neurons. If we use one-hot encoding, the total number of weights is $C \cdot H$. However, if we introduce the embedding layer, this total number of weights becomes $C \cdot E + E \cdot H$. So to reduce the total number of weights, we need to satisfy the inequality:

$$E \leq \frac{C \cdot H}{C + H}$$

Thus, to compress the information of all categorical features in a less dimensional space, we introduce a linear compression factor R, such as:

$$E = \frac{C \cdot H}{(C + H) \cdot R} \quad R \in \mathbb{R} \; with \; R > 1$$

With this approach, we compress the information of each categorical feature and thus reduce the number of required weights from $C \cdot H$ to $\frac{C \cdot H}{R}$.

5. Dataset

To perform a comprehensive analysis of GLEm-Net we use four different datasets: four well-known and open datasets and two that we collected in a real industrial environment. All datasets consist of both categorical and numerical features and describe binary classification problems. Table 2 summarises the main characteristics of each dataset.

5.1. Adult income

The *Adult income* dataset is a well-known dataset introduced by Becker and Kohavi [42]. It contains socioeconomic and income information for individuals from the 1994 census database. We use this dataset to tune the parameters of GLEm-Net and compare it with SOTA. The dataset includes 48,842 records, each described by 14 features: 8 categorical and 6 numerical. The categorical feature with the highest cardinality is the native country, with 42 different values. The class label indicates whether a person's income is below (*Class 0*) or above $50,000 (*Class 1*). The task is to predict each individual class based on the provided features.

5.2. Sleep health and lifestyle

The dataset *Sleep health and lifestyle* provides information on individual sleep patterns, health status and lifestyle habits, see [43] for details. This dataset is very small, with only 374 rows, each described by 12 features (3 categorical and 9 numerical features). We use this dataset to evaluate the ability of GLEm-Net to deal with small datasets. Similar to the *Adult Income* dataset, the categorical features have limited cardinality, with the"occupation" having the highest cardinality (11 different categories). The dataset consists of three classes. Here we simplify the task to a binary classification problem: individuals with or without sleep disorders.

5.3. Production line

The dataset *Production line* contains data collected in a real industrial environment, in particular detailing several steps of the production line for the assembly of printed circuit boards (PCBAs) used in electronic devices [3]. We have introduced this dataset in our preliminary assessment of GLEm-Net. It comprises 2,964,178 rows, each representing a soldered pin on the PCBAs during a production week. Each row is described by 26 features, 7 of which are categorical. The ability to compress and encode categorical features plays an important role here, as some categorical features comprise up to 1000 different categories.

The labels are derived from the results of the Automatic Optical Inspection (AOI) and indicate whether a pin is classified as a potential

Table 2
Dataset description.

	Adult income	Sleep health and lifestyle	Production line	Data challenge	KDD Cup 1999 (10 %)	Secondary Mushroom
Number of Rows	48,842	374	2,964,178	30,748	494,021	61,069
Categorical	8	3	7	5	9	17
Numerical	6	9	19	12	32	3
Class 0	11,687 (23.93 %)	219 (58.56 %)	17,193 (0.58 %)	951 (3 %)	97,278 (19.7 %)	27,481 (45 %)
Class 1	37,155 (76.07 %)	155 (41.44 %)	2,946,985 (99.42 %)	29,797 (97 %)	396743(80,3 %)	33,587 (55 %)
Reference	[42]	[43]	[3]	[44]	[46]	[45]

defect. If the AOI detects a potential defect, the board undergoes visual inspection by an operator. As expected in real production lines, the dataset is very unbalanced, with only 0.58% of the rows representing defective pins (*Class 0*).

5.4. Data challenge

The *Data challenge* dataset was collected in a real industrial environment [44]. It describes detailed data from the Solder Paste Inspection (SPI) machine, that inspects the solder paste before placing and soldering the components on the board. The dataset contains defective pins identified by the AOI test. Each pin has 17 features with 5 categorical features, with some categorical features having more than 300 different categories. If the AOI test fails, a human operator visually inspects the pins to verify them and finally label those where the soldering paste is defective. The dataset contains 30,478 rows, which are less unbalanced than the previous one as 3 % of the samples are labelled as defective (*Class 0*) and 97 % as *Class 1*.

5.5. KDD cup 99

The *KDD cup 99* dataset [46] is a widely used benchmark for intrusion detection tasks [11]. It contains simulated network traffic data covering both normal connections and a wide range of intrusions network environment. More specifically, we used the 10 % sampled version of the KDD dataset to perform experiments without the computational overhead of the full dataset. The dataset includes 494,021 records, each described by 41 features: 9 discrete and 32 continuous. The original labelling distinguishes 22 types of attacks in addition to the normal class. As is common in related work, we group all attack types under a single class label (*Class 1*) and treat normal traffic as a negative class (*Class 0*). The task is to predict whether each connection is a normal connection or an attack based on the provided features.

5.6. Secondary mushroom

The *Secondary mushroom* dataset is a synthetic dataset generated from a primary dataset of annotated mushroom species [45]. It contains 61,069 rows representing hypothetical mushrooms, structured to simulate 173 species with 353 samples per species. Each sample is described by 20 features, of which 17 are categorical and 3 are numerical. The classification task is to predict whether a mushroom is edible (*Class 0*) or poisonous (*Class 1*).

6. Experiments

In this section, we present how we conduct our experimental campaign. First, we describe how we split the datasets to perform feature selection, tune the classifier, train the model, and evaluate the performance. Then we describe the performance metrics, the classifier we use, and finally, the algorithms we use as the baseline for feature selection and how we set up the GLEm-Net methodology.

6.1. Model training, tuning, validation and test

We split our datasets into two parts: the *training set*, which we use to select the subset of features fit and tune the machine learning models. And the *test set*, which we only use to evaluate model performance.

In particular, for the datasets *Adult income* and *Data challenge*, we rely on the split of training and test proposed by the original authors [42,44]. For the datasets *Sleep health and lifestyle*, *Secondary mushroom*, *KDD cup 1999* and *Production line*, we keep 80 % of the samples for the training set and the remaining 20 % for the test set. Before model training, we preprocess all datasets by removing duplicate entries and discarding rows with missing values.

To optimise the hyperparameters of the classifier, we perform a grid search to find the combination of hyperparameters that maximises model performance. For this, we further split the training set into two parts:

- 90 % of the data is used to run the feature selection algorithms and train different classification models
- the remaining 10 % of the data is used as *validation set* to select the best hyperparameters of the classifier for each feature subset.

Once we identify the best classifier for each feature subset in the input, we evaluate the final performance using the *test set*.

6.2. Performance metrics

To evaluate the model performance we rely on four metrics: the macro F1-score, Accuracy, Area Under the Curve (AUC) and Matthews Correlation Coefficient (MCC).

- The **F1-score** is calculated as the harmonic mean of precision (the correctness of the positive predictions) and recall (the ability to identify all positive instances).

$$F1\text{-}score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

  The macro F1-score is an average F1-score for each class that treats all classes equally, regardless of their support (the number of true instances for each class). This approach ensures that each class contributes equally to the final metric, making it particularly useful when the class distribution is unbalanced.
- The **Accuracy** measures the proportion of correct predictions out of the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

  Accuracy complements the macro F1-score by providing a simple and intuitive measure of overall performance. However, it can be misleading in unbalanced data sets as it can be influenced by the majority class. To improve the readability of the main text, we report Accuracy results in Appendix B.
- The **Matthews Correlation Coefficient (MCC)** evaluates the quality of binary (or multi-class) classifications. It is considered a balanced metric, even if there is a class imbalance. classifier It is computed

Table 3
Training hyperparameter space.

Training parameters	Values
Learning rate	$\{10^{-2}, 10^{-3}\}$
Epochs	{30, 100, 150}
Step size (SGD parameter)	{30, 100}
Gamma (SGD parameter)	{0.1, 0.5, 1}
Optimizer	{SGD}
Classification threshold	[0.0,1.0] step 0.1

after classification as the ratio between correctly and incorrectly classified instances.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

The MCC ranges from -1 to $+1$: $+1$ indicates a perfect prediction, 0 for a random prediction and -1 for a complete deviation between predictions and actual labels.

- The **Area Under the Curve (AUC)** quantifies the ability of a model to distinguish between classes. Unlike MCC, AUC is computed before a classification threshold is applied to assess how effectively the model assigns higher confidence scores to positive instances than negative ones. AUC therefore measures the quality of the model's ranking function across all possible decision thresholds used to classify instances. To this end, AUC computes the True and False Positive Rates across different classification thresholds, constructs the Receiver Operating Characteristic) curve and measures the area underneath it.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

$$AUC = \int_0^1 TPR(FPR)\, dFPR$$

The AUC ranges from 0 to 1: An AUC of 1.0 indicates perfect separability, an AUC of 0.5 implies no discriminatory power (equivalent to random guessing), while an AUC < 0.5 means that the classifier is worse than random guessing.

6.3. Classifer architecture

For the classifier, we use a multilayer perceptron (MLP) consisting of two hidden layers, with 16 neurons in the first hidden layer, 8 neurons in the second hidden layer, a ReLU activation function, and a single final output neuron to which a Sigmoid function is applied. For categorical features, we incorporate embedding layers with a compression factor of $R = 2$, effectively reducing the number of parameters in the neural network by up to 33 %.

Given the feature subset used in the input, we define the architecture of the MLP and tune the hyperparameters listed in Table 3. More specifically, for each set of hyperparameters, we select those that maximise the macro F1-score over the validation set.

6.4. Feature selection baselines

To compare the performance of GLEm-Net with different SOTA feature selection methods, we choose two filtering methodologies and a wrapper method. In detail, we consider:

- *mRMR*: ranks features based on *Mutual Information Difference*, a metric that combines the importance of each feature (measured as correlation with the target class) with the redundancy that the feature [8] would introduce.
- Feature importance for *Catboost*: is a method within the Gradient Boosted Trees family specifically designed for managing categorical features. We perform a 5-fold grid-search cross-validation over the training set and select the model that provides the best performance (macro F1-score). Then we use its *Loss Function Change* (LFC) to rank the features [54].

After deriving the ranked lists, we iteratively create different subsets of features $S_{R_i}(j)$, where each subset consists of the top j features of the ranked list R_i:

$$S_{R_i}(j) = \bigcup_{k=1}^{j} R_i(k) \quad j \in (1, ., |R_i|)$$

For the wrapping approach we use:

- *Catboost* based backward feature elimination [55]: is a wrapper approach that uses Catboost to identify the best subset of features. In contrast to the previous cases where we use a single ranking, here we compute rankings and re-fit the model iteratively. Specifically, we start with all features and perform a grid search to identify the best Catboost model using 5-fold cross-validation over the training set. From this model, we use LFC to rank the features and remove the least important feature. We save the remaining features as candidate features and then repeat the grid search, updating the LFC rank and eliminating the least important feature. This process continues until only one feature remains.

After this process, we have different subsets of features $S_{R_i}(j)$ for each rank R_i.

6.5. GLEm-Net methodology

For GLEm-Net, we use the same MLP architecture as described above and modify the loss function so that it contains the term L_{GL}. Then we change the value of λ to select different subsets of the features. When λ is set, the network automatically identifies the best subset of features without having to specify the number of desired features. At the end of training, all features whose norm is below the threshold ϵ are eliminated and the resulting model is used for testing. To eliminate the features during testing, we set 0 for all neurons in the input layer that are associated with these features. When considering the EL function, we choose $\beta = 0.05$ and $\gamma = 1$ to ensure that all initialisation weights fall within the linear range after initialisation. If multiple values of λ select subsets of the same size, we choose the subset that performs the highest according to the macro F1-Score.

7. Results

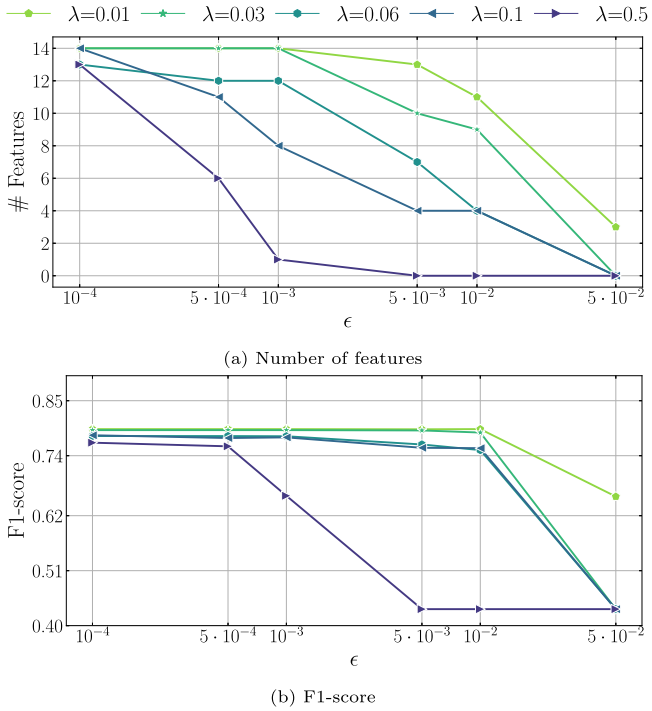
In this section, we present how we tune GLEm-Net parameters and compare its performance with respect to the other SOTA feature selection methodologies.

7.1. Tuning ϵ

The hyperparameter ϵ serves as a critical threshold for feature selection process. Remember that we eliminate all features whose norm is below ϵ. If ϵ is set too low, this filter and the feature selection become ineffective. Conversely, if ϵ is set too high, the norms fall below the threshold too easily, resulting in a large number of features being eliminated. Therefore, careful calibration of ϵ is essential to balance between retaining too many features and being too selective.

In Fig. 3 we show how the choice of ϵ affects the size of the feature set (top plot) and the performance for different values of λ for the Adult income dataset.

Consider Fig. 3(a), where we show the number of selected features. The higher ϵ is, the more aggressive the feature selection is. Conversely,



(a) Number of features

(b) F1-score

Fig. 3. Impact of ϵ for different λ values using the Adult income dataset.

for very small values of ϵ (e.g., 10^{-4}), almost all λ do not perform feature selection. However, as ϵ increases, the number of features for all values of λ decreases.

Next, in Fig. 3(b) we analyse the trend of the macro F1-score as a performance metric. As expected, if we choose ϵ too small, we observe no difference in performance as all feature subsets are similar. Instead, with a high value of ϵ, λ impacts the performance and the classification performance decreases as expected. Similar behaviours occur with the other dataset (not reported here for brevity).

Considering the stability of the performance and the interaction with λ for a certain value of ϵ, we set $\epsilon = 10^{-3}$ for all experiments in the following.

7.2. GLEm-Net and SOTA comparison

Next, we compare the classification performance of GLEm-Net with the SOTA methodologies described in Section 3. To do this, we run the feature selection algorithm until we reach a minimum of three features. Fig. 4 shows the macro F1-score, AUC and MCC of GLEm-Netcompared to the SOTA methodologies in all datasets. For completeness, we report the Accuracy results in the Appendix B.

First, we focus on the results for the Adult income dataset (top figures). GLEm-Net achieves a classification performance comparable to other SOTA methodologies. Looking first at the macro F1-score, GLEm-Net and mRMR show similar performance even when only a few features are selected. In contrast, Wrapper and Filter (LFC) show a significant deterioration in performance when only a few features are selected. AUC and MCC metrics show the same performance trend, with mRMR and GLEm-Net showing an MCC of 0.5, demonstrating good predictive ability even with few features. This result is consistent with previous work on the same dataset, such as the authors in [56].

When applied to the Sleep health and lifestyle dataset, i.e., the smallest of the six datasets, GLEm-Net confirms the best performance in selecting the subset of features with an even higher performance than the other SOTA feature selection methodologies. In particular, it outperforms the other SOTA methodologies by achieving the best performance with the 5-feature subset across all evaluation metrics.

Next, we analyse the results obtained on the Data challenge dataset, which is characterised by high cardinality of categorical features and strong class imbalance. Our results are consistent with previous findings reported in [57–60] showing similar or better classification performances. In particular, GLEm-Net consistently achieves higher macro F1-scores and outperforms all baselines on the MCC and AUC metrics, especially when analysed with reduced feature subsets.

We then analyse the results for the Production line dataset, which is the largest dataset and has the largest class imbalance. As expected, all methods show limited classification performance across all metrics and feature subsets due to the sevear class imbalance. Nevertheless, GLEm-Net remains competitive and is consistently on par with the best baselines for macro F1-score, MCC and AUC. Of particular note is that GLEm-Net continues to show slight performance advantages with very aggressive feature selection when looking at F1-score and MCC.

For both Secondary Mushrooms and KDD Cup, GLEm-Net is on par with the best-performing baseline methodologies, especially Wrapper and mRMR, in all evaluated metrics. These results confirm the robustness of our method on different datasets and different dimensions of the feature subset. It is worth noting that for the KDD cup dataset, the performance for subsets of more than 25 features is comparable to the performance of the neural network without feature selection, while the performance for subsets of less than 5 features decreases significantly. To improve readability, we therefore restrict the x-axis in the figure to only show results for feature subsets between 5 and 25.

To statistically evaluate the performance differences of GLEm-Net and the SOTA methodologies, In Fig. 5 we report the Critical Difference (CD) diagram computed across all evaluation metrics. CD diagrams [61] summarise the performance of each method based on the mean ranking. A post-hoc Nemenyi test is then performed to determine if differences are significant. Fig. 5 reports the Critical Difference Diagrams for all performance metrics. The number indicates the mean of the ranking for each method, while the solid line indicates the statistical significance of the performance differences. The results confirm that GLEm-Net is among the methodologies with the highest mean ranking. The results of the Nemenyi test show that the performance of GLEm-Net is in line with the best-performing baselines (mRMR and Wrapper – depending on the metric). Indeed, the feature selection stage is important, but we do not expect a particular algorithm to always take the lead. Depending on the task and data characteristics, different feature importance algorithms would lead to different results. In this respect, GLEm-Net is one of the top-performing algorithms and is very simple: it integrates feature encoding and selection into the training of the neural network, thus avoiding a separate feature selection step before training the network. Thus, GLEm-Net simplifies the process by fitting the MLP model only once and delivering both a selected feature subset and a trained classifier in a single step. This increases the overall efficiency of the feature selection process.

7.3. Impact of λ

In the following we investigate how λ impacts the number of selected features. Fig. 6 shows the impact of λ on the number of selected features for all datasets. The red area highlights the values of λ that have less than three features for which the problem degenerates.

As expected, the higher the λ value, the lower the number of selected features. Consider the first three datasets. For a very small value of $\lambda = 0.01$, no feature selection is performed, while a value of $\lambda \geq 1.00$ results in almost all features being eliminated, i.e., the loss function is dominated by L_{GL} [2] In contrast to the other datasets, the Production line dataset requires a completely different range of λ. Even very low values

[2] In a few cases, the number of features increases with a small increase in the λ value, e.g., for $\lambda = 0.10$ in the Sleep health and lifestyle dataset. This is due to the effect of λ in GLEm-Net, which does not directly force the elimination of features, but rather influences the reduction of the norm of the features. Therefore, the

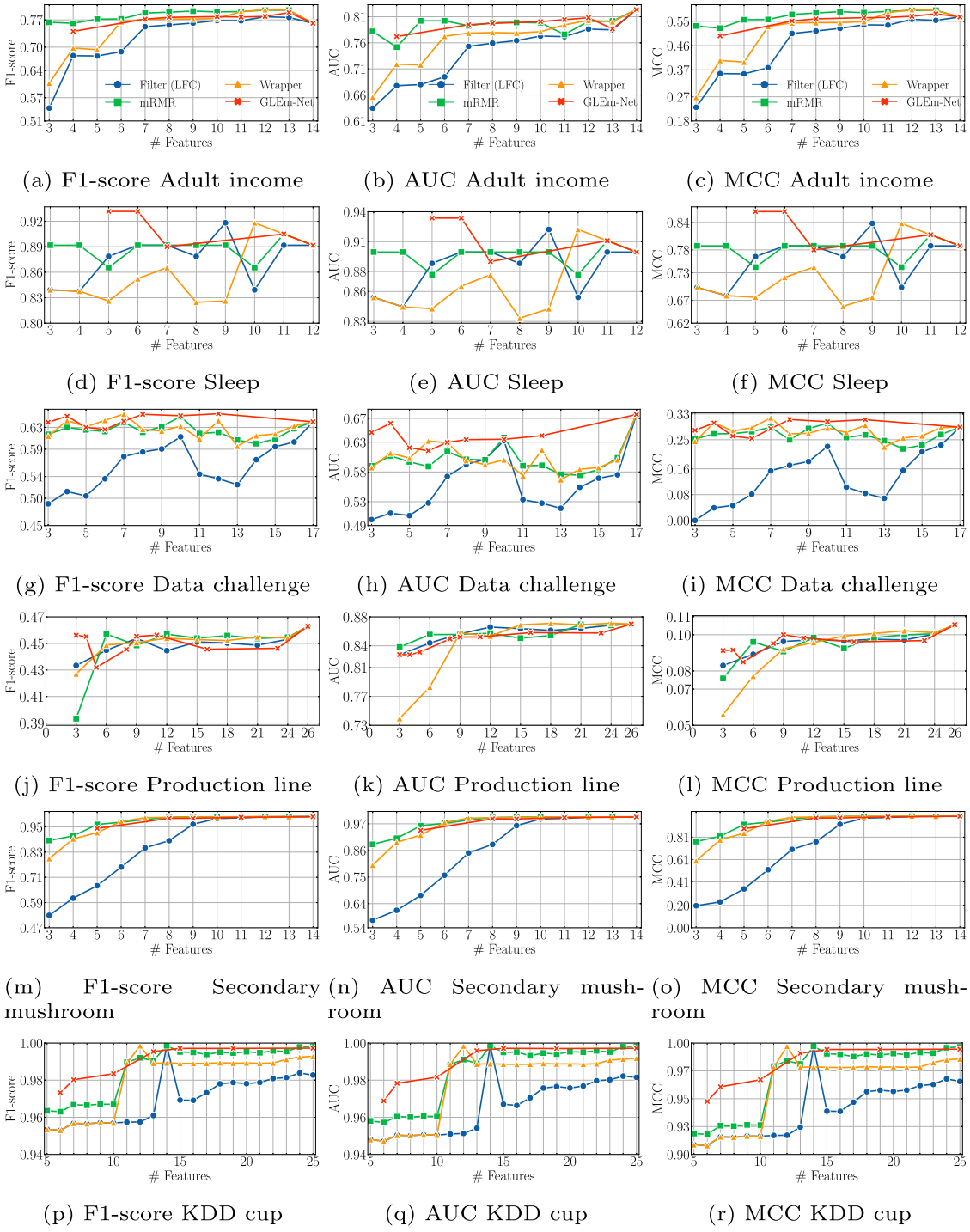


Fig. 4. F1-score, AUC and MCC across datasets for different feature subsets.

of λ result in aggressive feature selection, with GLEm-Net removing all features with a $\lambda \geq 0.012$. This is because there is little information available for prediction in this dataset.

To analyse how GLEm-Net works internally in this scenario, we show in Fig. 7 the evolution of the loss function (bottom) during training and the norms of the input features (top). Looking at the loss functions in Fig. 7(a), we observe that both L_{std} on the training and validation datasets show a limited reduction after two epochs. Conversely, L_{GL} continues to decrease significantly until the fifth epoch. This indicates that the MLP is able to effectively discard features without degrading performance and identify those features that provide little relevant information for the target task. Note that the L_{GL} term is independent of the input data and depends only on the weights of the MLP; thus, there is no difference between the training and validation terms.

Now focus on Fig. 7(a), which shows how the per input aggregated norm evolves during training. In some of the curves, the number of

neural network can seldom reduce the norms of multiple features during training and keep all of them slightly above the ϵ threshold instead of reducing the norm of a particular feature below the ϵ threshold. We investigate this scenario in Appendix A.

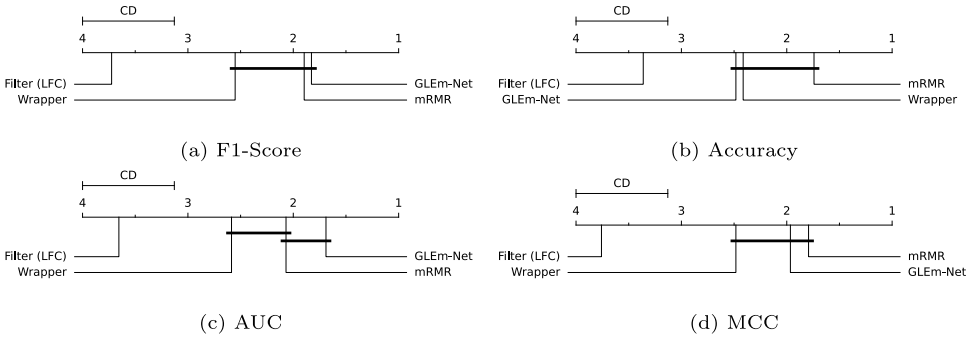


Fig. 5. Critical difference diagrams comparing the performance of the feature selection methods across the evaluation metrics.

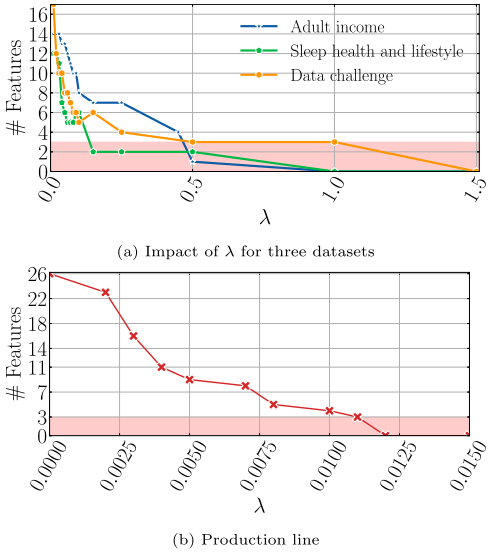


Fig. 6. Impact of λ on the different datasets.

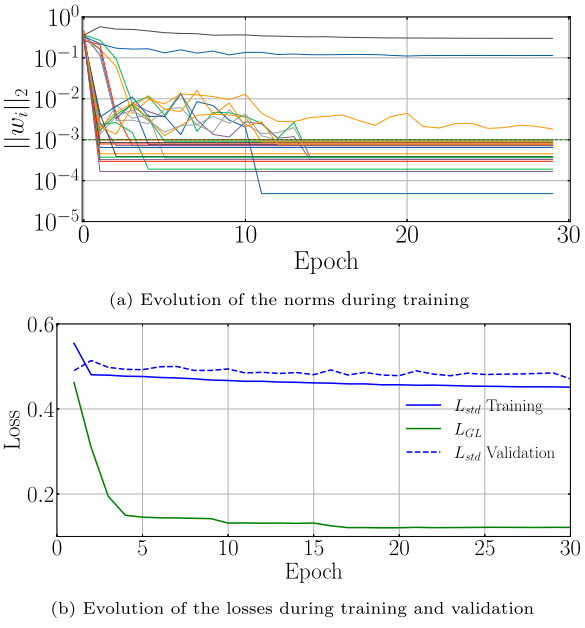


Fig. 7. Norms and losses evolution for the Production line dataset with $\lambda = 0.011$.

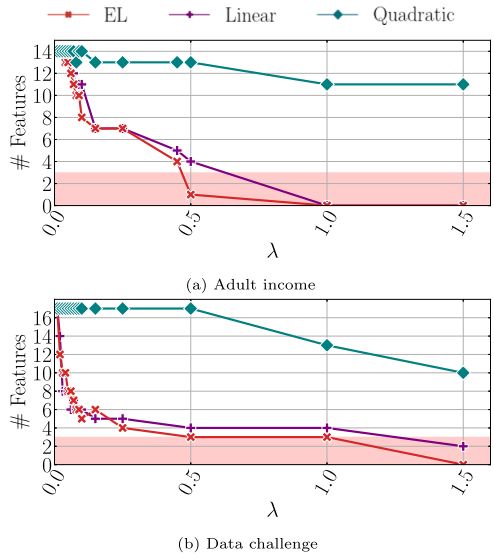


Fig. 8. Impact of the regularisation function on the number of selected features.

features decreases monotonically as the λ value increases. These are the input features that are not important. Conversely, the importance of three features remains significant (greater than ϵ). These are the features that GLEm-Net considers important.

7.4. Regularisation functions

We investigate the effects of the regularisation function on both feature selection and performance. To this end, we repeat the experiments from the previous section and compare the proposed EL function with a linear and a quadratic function.[3] For brevity, we only report the results for the Adult income and Data challenge datasets.

Focusing on the Adult income dataset in Fig. 8(a), we observe that both the linear and EL functions exhibit similar behaviour, producing feature subsets of comparable size, with EL being by construction more aggressive in reducing the number of selected features. The quadratic function, on the other hand, exhibits a very different pattern and offers only limited flexibility in selecting different subsets of features. This suggests that the quadratic function is less effective at selecting different subsets of features compared to the linear and EL approaches.

Similarly, the Data challenge dataset (Fig. 8(b)) shows that both the linear and EL functions have comparable performance. In contrast, the quadratic function shows little effectiveness when selecting a subset of features.

[3] We use the same parameters as suggested by the authors in the original papers.

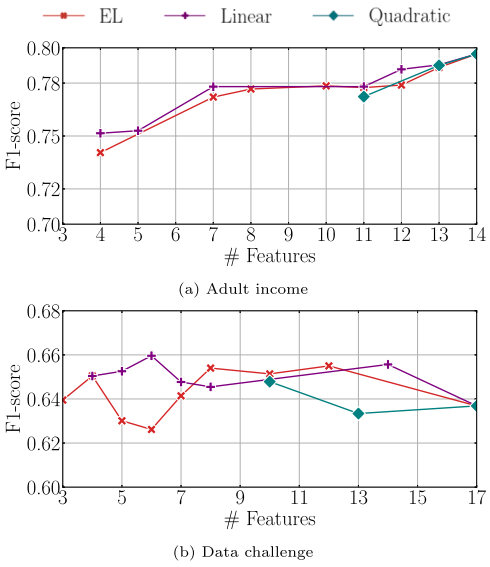


Fig. 9. Impact of the regularisation function on the classification performance.

Next, we examine the macro F1-score in Fig. 9. In the Adult income dataset (Fig. 9(a)), all functions show comparable performance.

In the Data challenge dataset (Fig. 9(b)), an interesting pattern emerges: the subset of six features selected by the linear function apparently shows the highest performance, outperforming the model with the same number of features selected by the EL function. However, note the scale of the macro F1-score – which are all similar in practise.

Overall, either the linear or EL functions show the best performance, with the EL function showing a finer ability to select optimal feature subsets. In contrast, the quadratic function shows limited effectiveness in selecting diverse and meaningful subsets of features.

7.5. Interpretability

Finally, we analyse the interpretability of GLEm-Net by examining how it works internally and comparing the features selected by the different feature selection methodologies.

Fig. 10 shows the evolution of the norms of the input features during the training of the Adult income dataset for three values of λ.

For $\lambda = 0.01$ – top plot, we see that the norm takes longer to reach a stable solution, with more features being selected at the end of the process. A higher value – $\lambda = 0.1$, bottom plot – on the other hand, quickly brings the norms close to zero. Those that go below ϵ are frozen and then eliminated. $\lambda = 0.05$ – middle plot – show intermediate behaviours.

Next, we examine the subset of features that are selected by the different methods as they decide on the same number of features. For brevity, we show in Fig. 11 the selected features for the Adult income and Data challenge datasets. The blue squares mark the features selected by the different methods, while the vertical green bar separates categorical and numerical features on the left and right sides, respectively. We sort the features by the number of methods that select them. In the Adult income dataset (Fig. 11(a)), the top two features are consistently selected by all methods. For the remaining features, however, each method has different selection patterns. In particular, GLEm-Net tends to favour a subset consisting predominantly of categorical features, similar to the filtering method (LFC), while the other methods show a more diverse selection.

In the Data challenge dataset (Fig. 11(b)), the situation is more heterogeneous. The Filter (LFC) method selects mostly numerical features and shows the worst performance of all methods. In contrast, the other

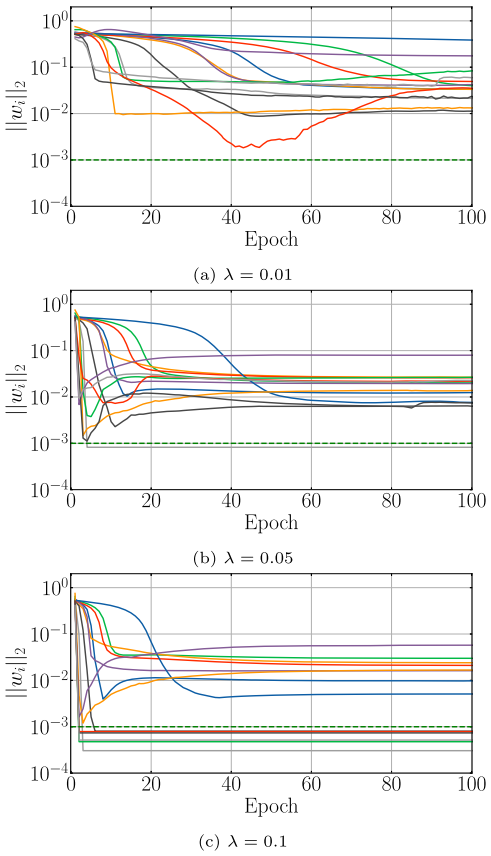


Fig. 10. Evolution of the feature norms during training for different λ values for the Adult income dataset.

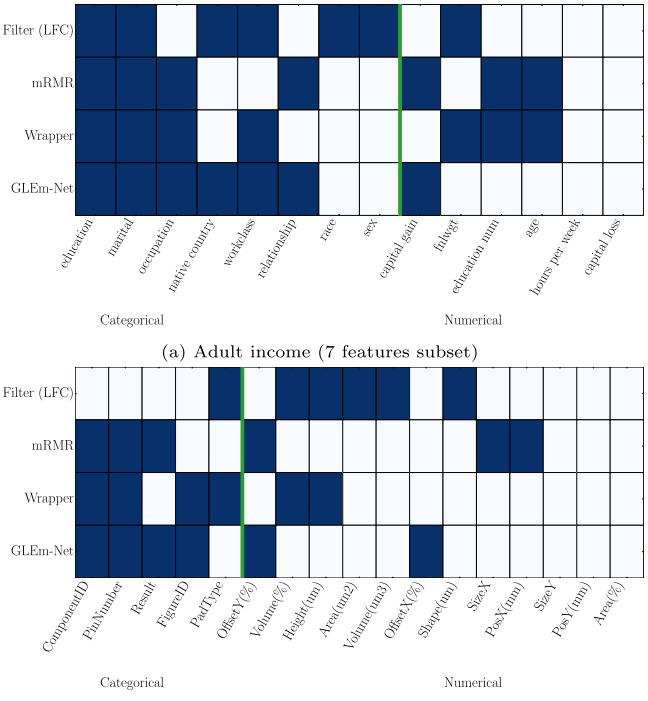


Fig. 11. Feature Selected for different datasets.

Table 4
Comparison of the complexity of the different algorithms for feature selection.

Method	Feature Selection Cost	NN Training Cost
mRMR	$O(n^2m)$	$O(k \cdot T_{\text{NN}})$
CatBoost Filter	$O(gcT_{\text{CB}})$	$O(k \cdot T_{\text{NN}})$
CatBoost Wrapper	$O(kgcT_{\text{CB}})$	$O(k \cdot T_{\text{NN}})$
GLEm-Net	$O(1)$	$O(k \cdot T_{\text{NN}})$

methods use both categorical and numerical features with different subsets of features and show comparable performances.

7.6. Computational complexity and execution time

Finally, to understand the benefits of GLEm-Net on performing a single training, we compute the asymptotic computational complexity and stimated execution time of each methodology.

Considering n as the number of features, m the number of samples, k the number of evaluated feature subsets, T_{NN} the cost of training the neural network, T_{CB} the cost of training CatBoost, g the number of hyperparameter configurations and c the number of cross-validation folds. The total costs for training the neural network over k feature subsets are $O(k \cdot T_{\text{NN}})$.

In comparison, the computational costs associated with feature selection for the different methods are:

- **mRMR**: $O(n^2m + k)$
  n^2m from computing pairwise mutual information for mRMR.
- **CatBoost Filter**: $O(gcT_{\text{CB}})$
  One CatBoost execution to rank features.
- **CatBoost Wrapper**: $O(kgcT_{\text{CB}})$
  Multiple CatBoost executions: one for each subset.
- **GLEm-Net**: $O(1)$
  The feature selection is embedded in the training of the neural network.

We report in Table 4 a comparison of the computational complexity of the different algorithms, taking into account the components feature selection and training of the neural network. GLEm-Net has lower computational costs compared to the baselines, as it avoids the expensive steps of separate feature selection and external model training by relying solely on a regularised neural network. As the integration of neural networks becomes increasingly important in a variety of applications, methods that embed feature selection directly into the training of neural networks are particularly valuable. This integration not only simplifies the pipeline by eliminating the need for separate feature selection steps, but also ensures that the relevance of the features is evaluated in the context of the model itself.

Next, we compare the execution time of the different methods. To ensure a fair and reproducible comparison, we estimate running times based on the theoretical complexity given in Table 4, calibrated with representative empirical measurements. In particular, we measure the time per CatBoost iteration (T_{CB}) and per neural network training step (T_{step}). We then estimate the NN training costs as $T_{NN} = \frac{m}{\text{batch size}} \cdot \text{epochs} \cdot T_{\text{step}}$, where T_{step} is the average time to process a batch. This approach provides consistent, dataset-specific execution time estimates while minimising variability due to implementation- or hardware-specific factors.

According to Table 5, GLEm-Net consistently achieves the lowest execution time. This is expected given its constant-time feature selection mechanism. The mRMR method has moderate to high computational costs, especially for datasets with a large number of features or samples (e.g., Production Line). The CatBoost Filter performs more efficiently than the CatBoost Wrapper. The latter incurs much higher costs due to the need to repeat the model training across multiple feature subsets.

Table 5
Estimated total execution times in seconds for different selection algorithms across datasets. The lower, the better. Best in bold.

Dataset	mRMR	CatBoost Filter	CatBoost Wrapper	**GLEm-Net**
Adult Income	153.59	352.63	2152.63	**152.63**
Sleep Health and Lifestyle	**1.17**	201.17	2001.17	**1.17**
Production Line	9463.43	9463.06	11263.06	**9263.06**
Data Challenge	96.98	296.09	2096.09	**96.09**
KDD Cup 1999	160.84	352.63	2152.63	**152.63**
Secondary Mushroom	193.28	390.84	2190.84	**190.84**

These results are in line with the previously reported theoretical complexities and emphasise the practical efficiency of GLEm-Net. While the differences are relatively small for the datasets considered here, the performance gap is expected to widen when the model is deployed on much larger datasets. Indeed, even within our experiments, the gap already increases noticeably when moving from Sleep Health and Lifestyle (the smallest dataset) to Production Line (the largest dataset).

8. Conclusions

In this paper, we presented GLEm-Net, a novel neural methodology that extends grouped Lasso regularization to mixed data types. Specifically, GLEm-Netmodifies the regularisation term by computing the expected value over the norms representing each categorical feature within the embedding layer. This extension enables the selection of features using neural networks for both categorical and numerical features while simultaneously reducing the number of parameters in the input layer. The resulting methodology provides both a selected feature subset and a trained classifier that can make predictions using this subset in a single training run.

According to our experimental results, GLEm-Net achieves performance comparable to the best state-of-the-art feature selection methods, while providing the additional benefit of a unified framework that integrates categorical feature encoding and feature selection directly into the training phase of the neural network.

The main limitation of GLEm-Net is the inability to directly specify the exact number of selected features. Instead, the hyperparameter λ controls the strength of feature selection and must be included in the hyperparameter optimisation process to determine the value that maximises performance. Despite our extensive evaluation of various state-of-the-art methods and datasets, we aim to improve GLEm-Net in future work by optimising the downstream classification task together with feature selection to further increase effectiveness.

CRediT authorship contribution statement

Francesco De Santis: Writing – original draft, Software, Methodology, Investigation, Data curation, Conceptualization; **Danilo Giordano:** Writing – original draft, Methodology, Investigation, Data curation, Conceptualization; **Marco Mellia:** Writing – review & editing, Supervision.

Data availability

We publish the entire code for replicating the experiments as open source and use open datasets to enable reproducibility.

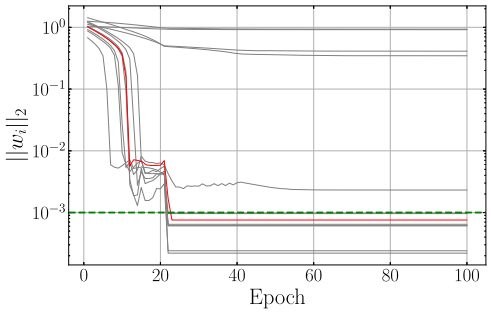
Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

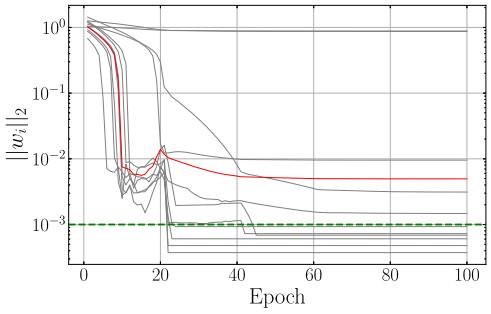
Acknowledgement

This research was supported by the SmartData@PoliTO research center and the project "National Centre for HPC, Big Data and Quantum Computing", CN00000013 (approvato nell'ambito del Bando M42C – Investimento 1.4 – Avviso "Centri Nazionali" – D.D. n. 3138 del 16.12.2021, ammesso a finanziamento con Decreto del MUR n. 1031 del 17.06.2022).

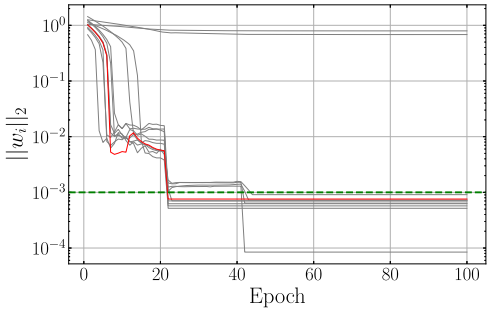
Appendix A. Investigate λ trend



(a) $\lambda = 0.08$, 5 features selected.



(b) $\lambda = 0.1$, 6 features selected.



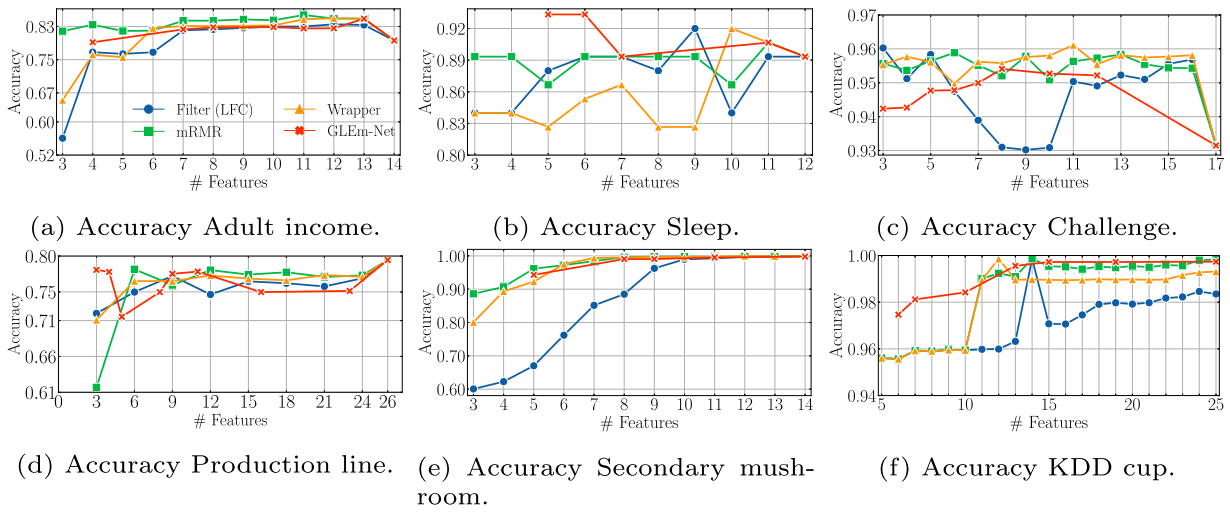
(c) $\lambda = 0.15$, 2 features selected.

Fig. A.12. Evolution of the features norms for the Sleep health and lifestyle dataset in the surroundings of the peak present in the λ analysis. The green dotted line represents the ϵ threshold whereas the red line represents the *blood Pressure min* feature.

To investigate the behaviour of GLEm-Net when the number of features increases with a small increase in the λ value, we report in Fig. A.12 the evolution of feature norms during training for the Sleep health and lifestyle dataset across three different λ values. Specifically, we select $\lambda = 0.10$, where we identify the anomalous increase in the number of features, the previous and the next λ value. At $\lambda = 0.08$, (Fig. A.12(a)), GLEm-Net selects 5 features. More precisely, 4 of them have a very high feature norm value close to 1, one is in the order of 10^{-2}, while the norm of all other features is below the threshold ϵ. At $\lambda = 0.10$ (Fig. A.12(b)), we see an increase in the number of selected features, with only two features whose norm is kept at a high value close to 1, while the other four selected features have a significantly lower norm. If we compare this subset with the previous one, we notice that in this case GLEm-Net retains the feature *blood pressure min* as an additional feature, represented by the red line. In this case, the regularisation term manages the feature norms differently than in the previous case with $\lambda = 0.08$, which causes the model to select an additional feature. When $\lambda = 0.15$ (Fig. A.12(c)), GLEm-Net selects only 2 features with a high value of their norms close to 1 and removes all other features, including the feature *blood pressure min* (i.e., the red line). This is due to the presence of mixed future subsets leading to similar performance and causing the model to select different subsets of features with small differences of λ.

Appendix B. Accuracy results

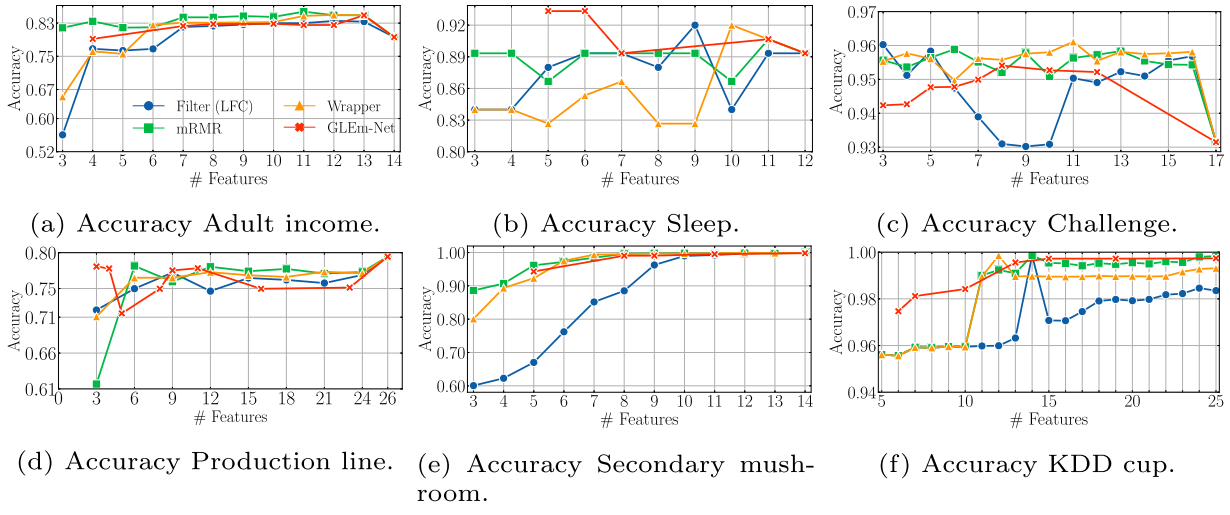
To complement the macro F1-scores, AUC and MCC reported in the main text, we include Accuracy results for all methods, datasets and feature subsets in this appendix. While Accuracy provides an intuitive measure of overall performance, it may not fully capture method effectiveness in imbalanced settings (Adult income, Production line, Data challenge and KDD cup). Therefore, we report it separately here to maintain clarity and focus in the main analysis Figs. B.13 and B.14.



(a) Accuracy Adult income. (b) Accuracy Sleep. (c) Accuracy Challenge.

(d) Accuracy Production line. (e) Accuracy Secondary mushroom. (f) Accuracy KDD cup.

Fig. B.13. Accuracy across datasets.



(a) Accuracy Adult income. (b) Accuracy Sleep. (c) Accuracy Challenge.

(d) Accuracy Production line. (e) Accuracy Secondary mushroom. (f) Accuracy KDD cup.

Fig. B.14. Accuracy results for the different metrics over the different datasets.

Appendix C. Robustness to missing or noisy data

To deepen GLEm-Net analysis and evaluate its robustness in the presence of missing and noisy data, we conduct experiments on the *Adult Income* dataset by gradually introducing missing values and noise, and compare the performance with that of state-of-the-art feature selection methods.

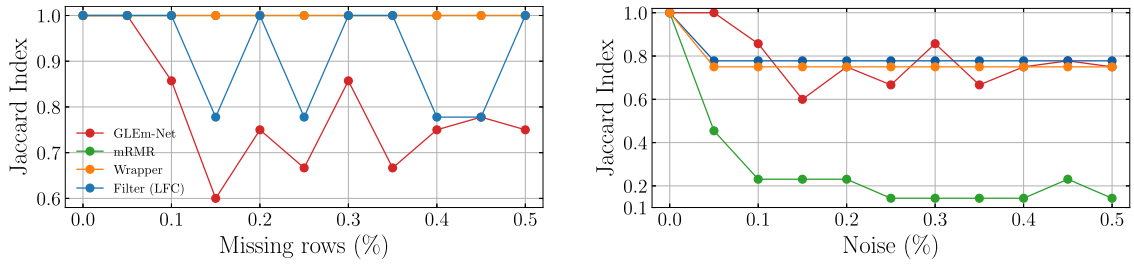
For this purpose, we consider the case where all algorithms select 7 features. Then, we alter the dataset introducing missing values or noise. For the *missing data* scenario, we randomly remove rows from the training set with increasing probabilities $p \in [0.05, 0.5]$. For the *noisy data* scenario, we inject perturbations: we add Gaussian noise to numerical features, while categorical features are randomly replaced by alternative values of the same feature. According to the existing literature, the stability of a feature selection algorithm is defined as the consistency of its feature preferences: an algorithm is deemed unstable if minor perturbations in the data result in substantial changes to the selected feature subset [62]. In line with [63], we employ the *Jaccard index* to quantify the similarity between the feature subsets obtained from each perturbed dataset and the original (unperturbed) subset:

$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$, where A and B are two feature subsets. A Jaccard index of 1 indicates identical subsets, while 0 denotes no overlap.

As shown in Fig. C.15, the Jaccard index decreases in both cases as the degree of alteration increases, which is expected given the increasing deviation from the original dataset. However, the trends are different for the different methods:

- **Missing Data:** mRMR and the Wrapper-based CatBoost methods retain stable feature subsets even with a high probability of missing data. In contrast, the Filtering-based CatBoost and GLEm-Net show greater adaptability, as evidenced by a gradual decrease in the Jaccard index. In fact, GLEm-Net automatically adapts to the dataset specificity during training, thus changing the feature selection accordingly.
- **Noisy Data:** mRMR is significantly affected by noise, with the Jaccard index decreasing sharply with increasing noise probability, reaching a value below 0.2. This testifies to the difficulties for an a priori method to select the correct features when these are noisy. In contrast, other methods remain stable and maintain a high similarity in the selected feature, which they select directly during training.

These results suggest that, while mRMR and CatBoost are more consistent in missing data scenarios, GLEm-Net exhibits strong robustness under noisy conditions. Overall, GLEm-Net maintains stable and meaningful feature selection across both types of data alteration, highlighting the reliability of its selection mechanism.



(a) Missing data (b) Noisy data

Fig. C.15. Jaccard index for selected feature subsets under increasing levels of missing and noisy data.

References

[1] Y. Wu, A. Zhang, Feature selection for classifying high-dimensional numerical data, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Washington, DC, 2004. Conference session presentation.

[2] A. Rouhi, H. Nezamabadi-Pour, Feature selection in high-dimensional data, Optimiz. Learn. Control Interdep. Compl. Netw. 1123, 2020, 85. https://doi.org/10.1007/978-3-030-34094-0_5

[3] F.D. Santis, D. Giordano, M. Mellia, A. Damilano, Glem-Net: unified framework for data reduction with categorical and numerical features, in: IEEE International Conference on Big Data, sorrento, Italy, 2023.

[4] H. Zhang, J. Wang, Z. Sun, J.M. Zurada, R. N, Feature selection for neural networks using group lasso regularization, IEEE Trans. Knowl. Data Eng. 32 (4) (2020) 659–673. https://doi.org/10.1109/TKDE.2019.2893266

[5] I. Lemhadri, F. Ruan, L. Abraham, R. Tibshirani, LassoNet: a neural network with feature sparsity, J. Mach. Learn. Res. 22 (2019) 29. http://jmlr.org/papers/v22/20-848.html.

[6] A. Jović, K. Brkić, N. Bogunović, A review of feature selection methods with applications, Conference session presentation at the International Convention on Information and Communication Technology, Electronics and Microelectronics, opatija, Croatia, 2015.

[7] D.M. Hawkins, The problem of overfitting, J. Chem. Inf. Comput. Sci. 44 (1) (2004) 1–12. https://doi.org/10.1021/ci0342472

[8] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, IEEE Trans. Pattern Anal. Mach. Intell. 27 (8) (2005) 1226–1238. https://doi.org/10.1109/TPAMI.2005.159

[9] D. Giordano, E. Pastor, F. Giobergia, T. Cerquitelli, E. Baralis, M. Mellia, A. Neri, D. Tricarico, Dissecting a data-driven prognostic pipeline: a powertrain use case, Expert Syst. Appl. 180 (2021) 115109. https://doi.org/10.1016/j.eswa.2021.115109

[10] J. Hermo, V. Bolón-Canedo, S. Ladra, Fed-MRMR: a lossless federated feature selection method, Inf. Sci. 669 (2024) 120609.

[11] V. Herrera-Semenets, L. Bustio-Martínez, R. Hernández-León, J.V. Den, Berg, A multi-measure feature selection algorithm for efficacious intrusion detection, Knowl. Based Syst. 227 (2021) 107264.

[12] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1986) 81–106.

[13] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.

[14] A.V. Dorogush, V. Ershov, A. Gulin, Catboost: gradient boosting with categorical features support, arxiv:1810.11363, 2018.

[15] C. Strobl, A.-L. Boulesteix, A. Zeileis, T. Hothorn, Bias in random forest variable importance measures: Illustrations, sources and a solution, BMC Bioinf. 8 (2007) 1–21.

[16] M.B. Kursa, W.R. Rudnicki, Feature selection with the boruta package, J. Stat. Softw. 36 (11) (2010) 1–13. https://doi.org/10.18637/jss.v036.i11

[17] X.-O. Ping, Y.-J. Tseng, Y.-P. Lin, H.-J. Chiu, F. Lai, J.-D. Liang, G.-T. Huang, P.-M. Yang, A multiple measurements case-based reasoning method for predicting recurrent status of liver cancer patients, Comput. Ind. 69 (2015) 12–21. https://doi.org/10.1016/j.compind.2015.01.007

[18] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Mach. Learn. 46 (1) (2002) 389–422. https://doi.org/10.1023/A:1012487302797

[19] B.J. Ma, S. Liu, A.A. Heidari, Multi-strategy ensemble binary hunger games search for feature selection, Knowl. Based Syst. 248 (2022) 108787.

[20] R.P. Lakshme, S.G. Kumar, Feature selection using binary horse herd optimization algorithm with lightgba ensemble classification in microarray data, Knowl. Based Syst., 2025, 113168.

[21] M. Habib, V. Vicente-Palacios, P.G. á Sánchez, Bio-inspired optimization of feature selection and svm tuning for voice disorders detection, Knowl. Based Syst. 310 (2025) 112950.

[22] E. Gasca, J. Sánchez, R. Alonso, Eliminating redundancy and irrelevance using a new mlp-based feature selection method, Pattern Recognit. 39 (2) (2006) 313–315. https://doi.org/10.1016/j.patcog.2005.09.002

[23] Y. Lu, Y. Fan, J. Lv, W.S. Noble, Deeppink: reproducible feature selection in deep neural networks, Adv. Neural Inf. Process. Syst., 31 2018.

[24] Z. Atashgahi, X. Zhang, N. Kichler, S. Liu, L. Yin, M. Pechenizkiy, R. Veldhuis, D.C. Mocanu, Supervised feature selection with neuron evolution in sparse neural networks, Trans. Mach. Learn. Res. 2023 (02) (2023). https://doi.org/10.48550/arXiv.2303.07200

[25] A. Saha, R. N, Groupfeature (sensor) selection with controlled redundancy using neural networks, 2023, arXiv:2310.20524.

[26] W. Wydmański, M. Śmieja, Gfsnetwork, Differentiable feature selection via gumbel-sigmoid relaxation, 2025, arXiv:2503.13304.

[27] F. Zimmer, P. Okanovic, T. Hoefler, Entryprune, Neural network feature selection using first impressions, 2024, arXiv:2410.02344.

[28] R. Zhang, X. Ma, C. Zhang, W. Ding, J. Zhan, GA-FCFNN: a new forecasting method combining feature selection methods and feedforward neural networks using genetic algorithms, Inf. Sci. 669 (2024) 120566.

[29] K. Sun, S.-H. Huang, S. D, -H. Wong, S.-S. Jang, Design and application of a variable selection method for multilayer perceptron neural network with lasso, 28, 2016. https://doi.org/10.1109/TNNLS.2016.2542866

[30] K. Jia, M. Rinard, Effective neural network l_0 regularization with binmask, 2023, arXiv:2304.11237.

[31] J. Wang, H. Zhang, J. Wang, Y. Pu, R. N, Feature selection using a neural network with group lasso regularization and controlled redundancy, IEEE Trans. Neural Netw. Learn. Syst. 32 (3) (2021) 1110–1123. https://doi.org/10.1109/TNNLS.2020.2980383

[32] S. Solorio-Fernández, J.A. Carrasco-Ochoa, J.F. Martínez-Trinidad, A survey on feature selection methods for mixed data, Artif. Intell. Rev. 55 (4) (2022) 2821–2846. https://doi.org/10.1007/s10462-021-10072-6

[33] M. Wei, T.W. Chow, R.H. Chan, Heterogeneous feature subset selection using mutual information-based feature transformation, Neurocomputing 168 (2015) 706–718. https://doi.org/10.1016/j.neucom.2015.05.053

[34] S. Sharmin, M. Shoyaib, A.A. Ali, M.A.H. Khan, O. Chae, Simultaneous feature selection and discretization based on mutual information, Pattern Recognit. 91 (2019) 162–174. https://doi.org/10.1016/j.patcog.2019.02.016

[35] D.M. Doan, D.H. Jeong, S.-Y. Ji, Designing a feature selection technique for analyzing mixed data, in: Conference Session Presentation at the Annual Computing and Communication Workshop and Conference, las Vegas, NV, 2020.

[36] W. Tang, K.Z. Mao, Feature selection algorithm for mixed data with both nominal and continuous features, Pattern Recognit. Lett. 28 (5) (2007) 563–571. https://doi.org/10.1016/j.patrec.2006.10.008

[37] J. Paul, P. Dupont, et al., Kernel methods for heterogeneous feature selection, Neurocomputing 169 (2015) 187–195. https://doi.org/10.1016/j.neucom.2014.12.098

[38] S.-Y. Jiang, L.-X. Wang, Efficient feature selection based on correlation measure between continuous and discrete features, Inf. Process. Lett. 116 (2) (2016) 203–215. https://doi.org/10.1016/j.ipl.2015.07.005

[39] M.E. Warkiani, M.H. Moattar, A comprehensive survey on recent feature selection methods for mixed data: challenges, solutions and future directions, Neurocomputing 623 (2025) 129372. https://www.sciencedirect.com/science/article/pii/S092523122500044X. https://doi.org/10.1016/j.neucom.2025.129372

[40] V. Borisov, J. Haug, G. Kasneci, Cancelout: a layer for feature selection in deep neural networks, in: International Conference on Artificial Neural Networks, Springer, 2019, pp. 72–83.

[41] R. Rossi, A. Murari, M. Gelfusa, A deep learning framework for feature selection and dimensional analysis: variational explainable neural networks, 2025, p. 113940. https://www.sciencedirect.com/science/article/pii/S0950705125009852. https://doi.org/10.1016/j.knosys.2025.113940

[42] B. Becker, R. Kohavi, Adult income dataset, in: UCI Machine Learning Repository, 1996. https://doi.org/10.24432/C5XW20

[43] L. Tharmalingam, Sleep Health and Lifestyle dataset, Kaggle Repository, 2022. https://www.kaggle.com/datasets/uom190346a/sleep-health-and-lifestyle-dataset.

[44] D. Giordano, M. Trevisan, European Conference of the Prognostics and Health Management Society, 2022. Data Challenge, https://github.com/PHME-Datachallenge/Data-Challenge-2022.

[45] D. Wagner, D. Heider, G. Hattab, Secondary Mushroom, UCI Machine Learning Repository, 2021. https://doi.org/10.24432/C5FP5Q

[46] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, P. Chan, KDD Cup 1999 Data, UCI Machine Learning Repository, 1999. http://kdd.ics.uci.edu/databases/kddcup99/.

[47] F. Sigrist, A comparison of machine learning methods for data with high-cardinality categorical variables, 2023. https://doi.org/10.48550/arXiv.1604.06737

[48] A. Rajee, M.S. Satu, M.Z. Abedin, K.A. Ali, S. Aloteibi, M.A. Moni, Wffs-an ensemble feature selection algorithm for heterogeneous traffic accident data analysis, Knowl. Based Syst., 2025, 113089.

[49] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, 2001. Linear Methods for Regression, (2nd ed.).
[50] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, J. R. Stat. Soc. Ser. B (Stat. Methodol.) 67 (2) (2005) 301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x
[51] B. Avanzi, G. Taylor, M. Wang, B. Wong, Machine learning with high-cardinality categorical features in actuarial applications, ASTIN Bulletin J. IAA 54 (2024) 213–238. https://doi.org/10.1017/asb.2024.7
[52] C. Guo, F. Berkhahn, Entity embeddings of categorical variables, 2016. https://doi.org/10.48550/arXiv.1604.06737
[53] H.D. Meulemeester, B.D. Moor, Unsupervised embeddings for categorical variables, in: International Joint Conference on Neural Networks, 2020. Conference session.
[54] P. Anuradha, V.K. David, Feature selection and prediction of heart diseases using gradient boosting algorithms, in: International Conference on Artificial Intelligence and Smart Systems, Coimbatore, India, 2021. Conference session.
[55] K.Z. Mao, Orthogonal forward selection and backward elimination algorithms for feature subset selection, IEEE Trans. Syst. Man Cybernet. Part B (Cybernet.) 34 (1) (2004) 629–634. https://doi.org/10.1109/TSMCB.2002.804363
[56] N. Chakrabarty, S. Biswas, A statistical approach to adult census income level prediction, in: International Conference on Advances in Computing, Communication Control and Networking, India, Noidia, 2018. Conference session.
[57] A. Gaffet, N. Roa, P. Ribot, E. Chanthery, C. Merle, A hierarchical xgboost early detection method for quality and productivity improvement of electronics manufacturing systems, in: Conference Session Presentation at the European Conference of the Prognostics and Health Management Society, 2022.
[58] I. Schmidt, L. Dingeldein, D. Hünemohr, H. Simon, M. Weigert, Application of machine learning methods to predict the quality of electric circuit boards of a production line, Prognostics and Health Management Society, 2022. Conference session.
[59] H. Tang, Y. Tian, J. Dai, Y. Wang, J. Cong, Q. Liu, X. Zhao, Y. Fu, Prediction of production line status for printed circuit boards, European Conference of the Prognostics and Health Management Society, 2022. Conference session.
[60] J. Taco, P. Gore, T. Minami, P. Kundu, A. Suer, J. Lee, A novel methodology for health assessment in printed circuit boards, Conference Session Presentation at the European Conference of the Prognostics and Health Management Society, 2022.
[61] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[62] S. Nogueira, K. Sechidis, G. Brown, On the stability of feature selection algorithms, J. Mach. Learn. Res. 18 (174) (2018) 1–54.
[63] 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.