GLEm-Net: Unified Framework for Data Reduction with Categorical and Numerical Features

Francesco De Santis, Danilo Giordano, Marco Mellia
Department of Control and Computer Engineering
Politenico di Torino
Torino, Italy
{francesco.desantis, danilo.giordano, marco.mellia}@polito.it

Alessia Damilano
Electronic Division, Quality & Process Digitalization
Deparment, Bitron
Grugliasco, Italy
alessia.damilano@gru.bitron-ind.com

***Abstract*—In the era of Big Data, effective data reduction through feature selection is of paramount importance for machine learning. This paper presents GLEm-Net (Grouped Lasso with Embeddings Network), a novel neural framework that seamlessly processes both categorical and numerical features to reduce the dimensionality of data while retaining as much information as possible. By integrating embedding layers, GLEm-Net effectively manages categorical features with high cardinality and compresses their information in a less dimensional space. By using a grouped Lasso penalty function in its architecture, GLEm-Net simultaneously processes categorical and numerical data, efficiently reducing high-dimensional data while preserving the essential information. We test GLEm-Net with a real-world application in an industrial environment where 6 million records exist and each is described by a mixture of 19 numerical and 7 categorical features with a strong class imbalance. A comparative analysis using state-of-the-art methods shows that despite the difficulty of building a high-performance model, GLEm-Net outperforms the other methods in both feature selection and classification, with a better balance in the selection of both numerical and categorical features.**

***Index Terms*—data reduction, feature selection, neural network, categorical features, embeddings**

I. Introduction

In the era of big data, the transformative potential of machine learning techniques to extract actionable insights from large datasets is evident. However, the larger and more complex the datasets become, the more urgent the need for efficient data pre-processing methods that can handle mixed feature types. Categorical and numerical features are ubiquitous in real-world data, and the seamless integration of both types of features in data reduction and feature selection is crucial for the development of effective and interpretable machine learning models.

Data reduction techniques have long been recognized as essential tools for unlocking the potential of small and big data. By reducing high-dimensional data into more manageable and informative subsets, these methods not only improve computational efficiency but also contribute to the interpretability and generalizability of machine learning models. In this context, the selection of features is of central importance. It enables the identification and retention of only the essential features. So far, the focus has been on data reduction and feature selection for either numerical or categorical data, with only a few methods being able to process both types of features together. The fusion of these two feature types is crucial as it reflects the multidimensionality of data in the real-world.

In this paper, we introduce GLEm-Net (Grouped Lasso with Embeddings Network), a neural comprehensive feature subset selection framework that reduces the data by processing categorical and numerical features together and classifying the data simultaneously. GLEm-Net takes advantage of embedding layers to effectively manage high cardinality categorical features and compress their information in a less dimensional space. It directly integrates into the architecture of a neural network a feature selection technique to select the most important features. To this end, we take as a starting point the authors' work in [1], where they present the use of a *grouped Lasso* penalty function in a neural network. Here, we integrate a novel implementation of the *grouped Lasso* penalty function into the neural network's loss function so that the network gets rid of the less informative input features. After training, the neural network performs the classification task as usual.

We perform a comparative analysis of the performance of GLEm-Net versus different state-of-the-art methods (SOTA) for feature selection.

Our goal is threefold:

- We show the potential of the GLEm-Net framework for encoding, data reduction and classification of mixed data types.
- We compare our framework with SOTA approaches and identify best practices for handling mixed data types.
- We evaluate the performance of our framework and SOTA approaches in a real-world scenario.

To experiment with GLEm-Net we use data collected in a real industrial plant of BITRON SPA, a leading company in the development and manufacturing of mechatronic devices. In the manufacturing of electronic devices, production lines consist of multiple machines that collect large amounts of

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).

979-8-3503-2445-7/23/$31.00 ©2013 IEEE

mixed data. Our dataset describes the defectiveness monitored at the end of the first part of the production line. The dataset consists of more than 6 million records, each described by 19 numerical and 7 categorical features. The problem shows a strong class imbalance as there are only 0.58% defective records. In this scenario, reducing the dimensionality of the data is of paramount importance as collecting large amounts of data slows down the manufacturing process and helps domain experts to understand it.

Despite the difficulty of building a high-performance model, our results show that GLEm-Net outperforms the other SOTA methods both in selecting the most important features and in classification. Moreover, retraining the neural network after feature elimination improves performance by up to 3%. In contrast to the SOTA methods, which tend to select mainly numerical features, GLEm-Net is more balanced in selecting both numerical and categorical features.

To allow other researchers to experiment with GLEm-Net we release the code as open source [1].

The rest of the paper is organized as follows. Sec. II gives an overview of related work, while Sec. III dissects the GLEm-Net methodology. In Sec. IV we present the dataset, while in Sec. V we describe the experimental setup and our results. Finally, Sec. VI concludes the paper and presents possible future directions.

II. Related Work

Data reduction techniques such as sample reduction and feature selection are crucial for managing big datasets. Sample reduction helps reduce the size of the dataset by removing entire samples, while feature selection focuses on identifying the most informative features and retaining all samples. In both scenarios, the goal is to reduce the size of the dataset while preserving the information it contains [2]. In this paper, we focus on feature selection as it improves interpretability by focusing on the most relevant features while mitigating the risk of overfitting [3].

Feature selection methods can be divided into three main categories: filter, wrapper and embedded approaches [4]. Filter methods rank the features based on their correlation with the target variable. The authors of [5] present the mRMR method. mRMR is a method for selecting a subset of features that maximizes their relevance to the prediction of the target variable while minimizing their redundancy. This method has proven successful for both continuous [6] and categorical features [7]. For wrapper approaches, the performance of the classifier guides the selection of the best subset of features [8]. For mixed features, BORUTA [9] and RFE-SVM [10] are two valuable solutions. 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. The second method, RFE-SVM, focuses on identifying the features that significantly maximize the margin of an SVM classifier for class separation. This method uses a systematic sequential backward selection process in which one feature is progressively removed in each iteration until a certain number of features remain. Another notable study is presented by the authors in [11], in which they use a recursive backward feature elimination approach in conjunction with three different models: Catboost, Random Forest and Logistic Regressor. Their method starts with the full feature set and eliminates the least informative features in each iteration based on the feature importance scores obtained from these three models. For embedding solutions, the best subset of features is given as a by-product of the classification/regression process. The authors in [12] propose Elastic Net, a regression model that reconciles Lasso feature selection and ridge capability to handle multicollinearity.

With the increasing use of neural network methods, feature selection tailored to them becomes more and more important. In this direction, the authors in [13] propose LassoNet, a method for integrating feature selection mechanisms into a multi-layer 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. The authors in [14] propose an approach to feature selection 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 authors in [1] present another 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 regularization term based on the grouped Lasso penalty function. However, all these methods work either only with categorical features or only with numerical features and have limited applicability with mixed data types. In [15], the authors provide an overview of possible strategies for dealing with mixed data types: transforming categorical features, treating features individually, and applying a unified principle-based metric customized for each data type. However, each of these methods also has its drawbacks. The first approach may result in the original variable having no inherent ordering or spacing. The second strategy uses different metrics for different data types, making it difficult to compare results. The third strategy uses a uniform metric but adjusts its calculation for different data types, which creates similar difficulties when comparing features. Therefore, the development of methods to directly use mixed data remains an active challenge for the feature selection research community.

In contrast to them, here we develop a unified feature selection framework to handle mixed data types. We propose to integrate embedding layers [16] to encode categorical features and enable their analysis together with numerical features. This reduces the need for custom encoding. We then extend the use of the grouped Lasso penalty function to include categorical features. Finally, we validate it with a challenging dataset collected from a real-world scenario and characterized by categorical features with high cardinality, which typically

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

pose a challenge for neural networks, deep neural networks and tree-based models [17].

III. Methodology

Grouped Lasso is a regularization technique used in machine learning and statistics to select groups of features [1]. It effectively handles features that can be organized into smaller groups. Its application to neural networks enables the creation of predictive models while reducing feature dimensionality, thereby improving interpretability. Although this approach is promising, it cannot handle categorical features that are common in real-world tabular datasets. To overcome this limitation, we propose here GLEm-Net, a methodology that enables the effective processing of mixed-type features and adapts the Grouped Lasso regularization technique to categorical features.

A. Background

The Least Absolute Shrinkage and Selection Operator (*Lasso*) is a regularization 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's coefficients. This promotes sparsity and automatic feature selection and prevents overfitting of the model by preventing an excessive increase 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 described by this coefficient is considered useless for prediction.

Consider a simple neural network with p neurons in the input layer and 3 neurons in the hidden layer. Each neuron x_i in the input layer is connected to the first hidden layer through a series of weighted connections $w_{i,j}$. In this architecture, using the Lasso regularization term is not feasible because each input variable (i.e., each neuron) is described by multiple parameters, i.e., weights. To overcome this limitation, the *grouped Lasso* extends the Lasso regularization term by considering the grouped sets of coefficients as a penalty.

Consider an input neuron x_1 connected to the three neurons in the first hidden layer. The input variable becomes useless for prediction if the group of weights described by the vector $w_1 = (w_{11}\ w_{12}\ w_{13})^T$ has all components close to zero. Using the *grouped Lasso* regolarization term, we can modify the *Loss function* L used in training the neural network as follows to consider the input weights of the model as a group for each variable:

$$L = L_{std} + \underbrace{\lambda \sum_{i=1}^{p} ||w_i||_2}_{\textit{regularization term}}$$

$$L = L_{std} + \lambda \sum_{i=1}^{p} h(E_i[w])$$

The first term L_{std} represents a standard loss, such as the Cross-Entropy in the case of classification, which is used to reduce the error of the model for the task. The second term instead represents the penalty of the *grouped Lasso*, regularization term which is used to reduce the number of features [18]. We introduce a parameter $\lambda \geq 0$ that weights the importance of feature selection with respect to 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. To consider all weights of a feature as one 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 perform the sum of all norms and multiply them by the term λ to calculate the regularization term. If a feature x_i does not contribute to performance improvement during model training, the norm of its vector w_i is lowered to a value close to zero and this input neuron can be removed from the network. Unfortunately, using the norm as a regularization term resulted in a non-differentiable function at the origin due to a discontinuity. This discontinuity leads to possible numerical oscillations when $||w_i||_2$ is close to zero. Therefore, the *smoothed Group Lasso* introduces a function h that makes the regularization term differentiable at the origin and thus facilitates the convergence of the input vector norms [1].

$$L = L_{std} + \underbrace{\lambda \sum_{i=1}^{p} h(||w_i||_2)}_{\textit{regularization term}}$$

B. Smoothing function h

The authors of [1] propose smoothing functions that are characterized by the property of being differentiable at the origin. This property is fundamental to accelerate the convergence of the norms of the input vectors and to reduce the oscillations around the origin. Here we use this function to refine the feature selection process. The norm of a variable that is not notably beneficial to the neural network must be as close to zero as possible. This ensures that the feature is effectively eliminated without affecting performance when the same neural network is used for prediction tasks. To this end, we propose here a function that boosts the gradient of the regularization term as the norm approaches zero by using the following equation:

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

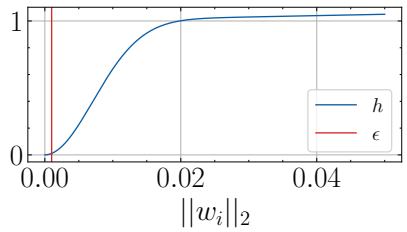

Fig. 1. h function with $\beta = 10^{-4}$, $\gamma = 1$ and $\epsilon = 10^{-3}$.

Fig. 1 shows the h function with $\beta = 10^{-4}$ and $\gamma = 1$. The function initially shows exponential behaviour, with β

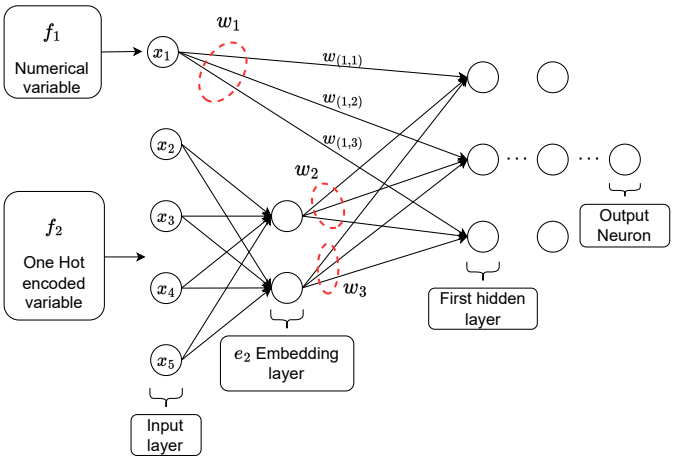

Fig. 2. Generic MLP with an embedding layer, having a categorical and a numerical variable in the input layer.

adjusting the size of this exponential segment. The function then switches to a linear phase in which the slope is equal to γ. The parameter γ provides an additional degree of freedom by allowing us to penalize the norms of those variables that the network considers less important. This function has the disadvantage that it is again not differentiable at the origin. Therefore, we freeze the weight vector w_i if its norm is smaller than a certain threshold ϵ. All in all, ϵ must be kept small enough to ensure that only the norms of a weight vector considered negligible are frozen during training. β and γ, on the other hand, cannot be fixed in advance and should be tuned during the grid search to find the optimal values. For β, it is important to consider the initialization of the neural network weights to ensure that all weights fall within the linear range after initialization. This allows all features to start from a neutral region characterized by a limited penalty gradient. According to our analysis, β could be in the range of 10^{-2} to 10^{-4}, while λ is in the range of 0.1 to 1.

C. 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 categorical data into continuous vector representations [19], which enables efficient processing and improved generalization 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 compresses the information better than is possible with a one-hot encoding method. Fig. 2 shows an example of a neural network architecture with two input features: f_1 numeric and f_2 categorical. While 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 $i \in [2,5]$ by using a one-hot encoding representation (4 neurons in this case). Then each of the categorical neurons is connected to the embedding layer e_2 of f_2, which has $|e_2| = 2$ neurons compressing the categorical feature information by 2. Then the embedding layer e_2 is fully connected to the first hidden layer of the network. For each categorical feature f_i, we have a different embedding layer e_i with a different number of neurons. In Sec. III-D we will describe how to set the number of neurons in this layer. To conclude, the neural network architecture on the right contains all the hidden layers of our classifier and the output neuron used for prediction.

For each categorical feature f_i, each neuron of the embedding layer e_i has n input connections with $n = |f_i|$, each with a different weight. The embedding neurons are used to initially map the categorical variable into coordinates in latent space. These coordinates are then used as input for the connections between the embedding and the hidden layer. Since they are part of the neural network, these weights can be learned during training along 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 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 compute the grouped Lasso penalty function. On the other hand, we cannot do this for the categorical feature since it is described by two vectors: $w_{(2,*)}$ and $w_{(3,*)}$. For a preliminary approach, one could use the Frobenius norm [20] on the submatrix $W_2 = [w_{(*,2)}, w_{(*,3)}]$. However, it is important to note that the resulting value will not be 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 regularization term.

In general, consider P 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 represents the generalized version of the loss function, that introduces the concept of grouped Lasso penalty into a network that can handle both numeric and categorical variables.

$$L = L_{std} + \lambda \sum_{i=1}^{P} h \left(\frac{1}{|l(i)|} \sum_{j \in l(i)} ||w_j||_2 \right)$$

$$L = L_{std} + \lambda \sum_{i=1}^{P} h \left(\frac{1}{|l(i)|} \sum_{j \in l(i)} ||w_j||_2 \right)$$

This equation is also applicable when the embedding layers are excluded and the regularization is applied directly to the connections between the one-hot-encoded variables and the first hidden layer. Note, however, that in the case of a categorical feature with high cardinality and low information, this approach could result in the neural network using an excessively large number of parameters.

D. Embedding size

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. In such scenarios, we expect the embeddings to significantly reduce the size of the vector used to represent these features without losing the amount of information available. It 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 $\frac{C \cdot H}{R}$.

IV. Dataset

To evaluate the results of our methodology, we use a dataset collected in a real industrial environment. It describes a production line for manufacturing electronic devices where printed circuit boards (PCBAs) are assembled. We use the term *board* to refer to a device. The dataset contains data from both the machines involved in the assembly processes and the environmental data within the facility. In our production line, the PCBAs are organised in panels, with each panel consisting of 8 boards. Each board consists of 43 components that are mounted and soldered to the board with multiple pins, for a total of 313 pins.

Fig. 3 shows our assembly line, where we collect environmental data and data from the manufacturing machines:

- Printing Machine: the laser prints a unique serial number on each board for identification, followed by the application of solder paste.

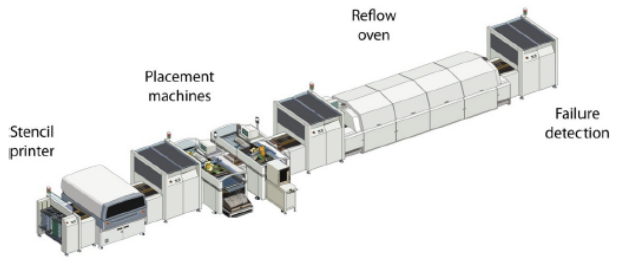

Fig. 3. Assembly line example.

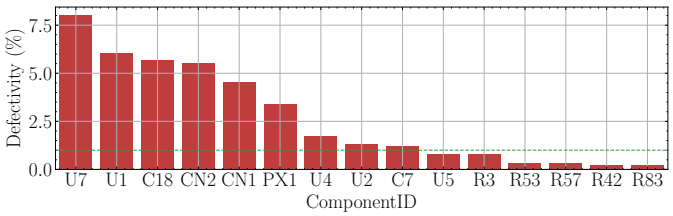

Fig. 4. Percentage of defectiveness per component for the top 15 components. The green line reports the threshold at 1%.

- Solder Paste Inspection (SPI) Station: inspects the solder paste applied by the printing machine.
- Pick and Place Station: places the electronic components on the boards.
- Reflow Oven: heats the solder paste until it is melted and forms permanent solder joints.
- Automatic Optical Inspection (AOI) Station: automatically inspects boards for defects, including missing components, fillet size/shape issues, component misalignment and more.

If the AOI detects a potential defect, the board is stopped and awaits visual inspection by the operator. The operator determines if it is a real defect, so the board undergoes a thorough inspection and possibly be reworked, or if it is a false defect the board can go back to the production line. If a defect is detected, the time to assemble the board and panel will be extended due to the operator's inspection. In addition, the machines slow down in collecting all this data because they collect a large amount of data for each board. Identifying the subset of data that contributes to defect detection can speed up the process by reducing the number of false defects and the amount of data to be captured by each machine, making the data collection process feasible during real production.

The dataset consists of $6,494,186$ rows representing one week of production totaling $2.5GB$. During this week, all machines were set to collect all data, slowing down the production line. Each row contains information about one pin in one board. For each pin, we collect information such as the area, height and volume of the solder paste, the position of the pin, the result of the Solder Paste Inspection, the component identifier, etc. Each row is labelled with the result of the Automatic Optical Inspection (AOI), which indicates

whether a pin has been classified as defective or not. Given the size of the dataset, we use our big data cluster and Apache Spark to analyse it.[2] As expected in real production lines, the dataset is very unbalanced with only 0.58% defective pins. In detail, Fig. IV shows the percentage of defective pins per component in descending order. Given the strong skewness of the distribution, with hundreds of components having no defectivity, we focus on the top components that have a defectivity of at least 1.00%.

As a result, we are left with 2,964,178 rows, each described by 7 categorical features and 19 numerical features, both from the machines involved in the assembly and from data sources in the environment.

V. Experiments

In this section, we present the pipelines we use as the baseline for feature selection and classification, and the GLEm-Net results.

A. Experimental Setup

We start by splitting our dataset into two parts: the training set with 80% of the samples and the remaining 20% intended for testing purposes. Given the significant class imbalance in the dataset and the different performance observed when choosing different classification thresholds, we use the Area Under Curve (AUC) metric to evaluate the classification performance.

First, we identify the architecture of the neural network used for the classification task. To do this, we use GLEm-Net without performing feature selection, i.e., we set the parameter $\lambda = 0$. Given the size of the dataset and the time required to train multiple models, we opt for a hold-out validation technique where we split the available training dataset into two parts; 85% is used to build the model, while the remaining 15% serves as the validation set to determine the best hyperparameters. Since the dataset is highly imbalanced, we use a batch size of 512 to ensure that there is at least one defective sample in each batch. To further counteract the problem of unbalanced data, we also assign a weight to each sample. Specifically, we assign a lower weight to working samples than samples with defective pins. We determine the weight of each class as follows: $\alpha_j = S/(2 \cdot S_j)$. Where S is the total number of samples in the training set and S_j is the number of samples belonging to class j within the training set.

TABLE I
Grid search space.

Hyperparameters	Values
First layer size	[16, 24, 32]
Second layer size	[0, 8, 16]
Hidden activation function	[ReLU, Tanh]
Output activation function	[Sigmoid]
R	[0 (No embeddings), 2, 3]

[2]https://smartdata.polito.it/computing-facilities/

To find the best architecture, we perform a grid search (Tab. I reports the grid space) using the Stochastic Gradient Descent optimization algorithm.

We use a Multi-layer Perceptron (MLP) consisting of two hidden layers. The architecture with the best performance in the validation set has 16 neurons in the first hidden layer, 8 neurons in the second hidden layer, Tanh as the activation function, and an embedding layer with a compression factor $R = 2$. This choice of R gives the same performance as using no embedding layer, i.e., a one-hot encoding approach. Once we have found the best MLP architecture for classification, we use it for all subsequent experiments.

B. GLEm-Net Framework

For GLEm-Net, we run the entire pipeline, enter all features and change the λ value to select different subsets of features. Note that when λ is set, the network automatically identifies the best subset of features without having to specify the number of desired features. We then calculate the performance on the test set. Note that after selecting the features, GLEm-Net already returns an MLP model trained with only the most important features. However, we consider the possibility of training the model from scratch using the only selected features as input. This is essential as highly informative features may be penalized by the regularization term, resulting in poorer performance of the model. Following the Lottery Ticket Hypothesis [21] we train the model with only the selected subset of features and set $\lambda = 0$.

C. Feature selection baselines

To compare the performance of GLEm-Net with different feature selection methods, we choose two state-of-the-art (SOTA) filtering methods and a wrapper method. For the filter methods we use:

- *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 [5] would introduce;
- Feature importance for *Catboost*: is a method within the Gradient Boosted Trees family specifically designed to manage categorical features. Specifically, we perform a grid search to find the model with the best performance on a 10-fold cross-validation. Then we use the *Loss Function Change* (LFC) to evaluate the features [22].

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

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

Finally, we use a wrapping approach where the best subset is identified as follows:

- *Catboost* based backward feature elimination [23]: is a wrapper approach that uses Catboost to identify the best subset of features. In contrast to the previous case where

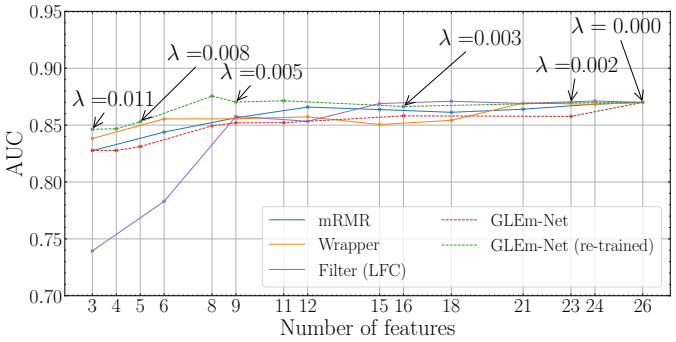

Fig. 5. Area Under Curve for the different feature selection methods.

we used a single ranking, here we compute multiple rankings by iteratively removing features by fitting multiple models for each iteration. Specifically, we start with all features and perform a grid search to determine the best Catboost [22] model using a 5-fold cross-validation of the training set. From this model, we use LFC to rank the features and remove the least a important features. We store the remaining features as a possible subset of features and then repeat the grid search with these features to identify the new a least important features. We continue this process until no feature is left.

After this process, we have a new subset of features $S_{R_i}(j)$.

Once we have all feature subsets, we evaluate the performance of each feature subset $S_{R_i}(j)$ using the MLP architecture described in Sec. V-A. For categorical features, we integrate embedding layers and train them together with the MLP architecture. This approach is important to enable a fair comparison among the different subsets. Specifically, we train the neural network with $S_{R_i}(j)$ as input features and evaluate the performance of each model on the test set.

Finally, we use a *learning curve* [24] to evaluate for each ranking R_i how the performance changes with increasing information, i.e., for each $S_{R_i}(j)$ $j \in (1, ..., |R_i|)$.

D. Results

Here we present the results of our experiments. Fig. 5 shows the classification performance for different subsets of features. To ease the representation we align the head and tail of the number of features selected by the SOTA methods with the feature selection results for GLEm-Net. For GLEm-Net, the red dashed line shows the performance on the test set without retraining the MLP network, while the green dashed line shows the performance after retraining the MLP network. Starting from the right, the performance without feature selection is equal to 0.87 for all methods since they use all features. When we start to reduce the features, we find that all feature selection methods perform similarly well. The GLEm-Net method shows lower performance without retraining. This is because the MLP network was trained with a regularization term that also penalizes highly informative features. This result underlines the need for retraining. The results become more promising when the number of features is significantly reduced. In the left part of the figure, we see that the retrained GLEm-Net delivers the best performance among all methods. For all SOTA methods, the performance is similar with an AUC of about 0.85. Only GLEm-Net achieves 0.87 when 8 out of 26 features are selected. Such a selection allows the production line to reduce the amount of data to be collected by a factor of 3, potentially allowing data collection during normal production.

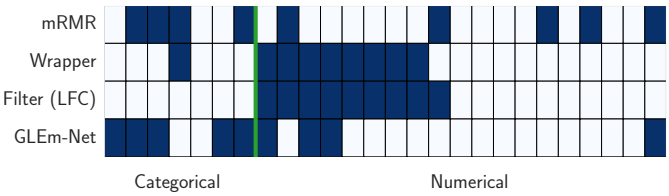

Fig. 6. Comparison of feature subset with 9 features. The green line separates categorical and numerical features.

Next, we examine the subset of features selected by the different methods. For this purpose, we analyze the different subsets consisting of 9 features. Fig. 6 shows with a blue square the features selected by the different methods. The green line separates categorical and numerical features. Note that no feature is selected by all methods. Interestingly, both Catboost-based methods select almost only numerical features, which are the same for both solutions. mRMR, on the other hand, is more balanced in the use of categorical and numerical features. Similarly, GLEm-Net shows a balanced selection of categorical and numerical features along with the best AUC performance.

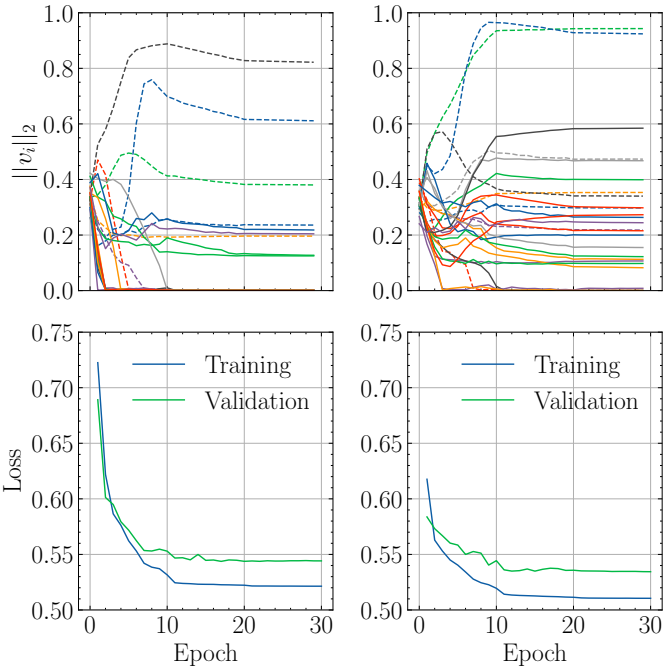

Fig. 7. Loss functions and norms evolution during training for $\lambda = 0.002$ (right column) and $\lambda = 0.005$ (left column).

To analyse how GLEm-Net works internally, we show in Fig. 7 the evolution of the loss function (bottom) during training and validation and the norms of the input features (top) during training for two values of λ. The solid lines indicate the norm of the numerical features, while the dashed lines indicate the norm of the categorical features. Focus on the left figures when $\lambda = 0.005$, where 9 features are selected at the end. Looking at the loss function, both training and validation converge after about 10 epochs. Looking at the evolution of the norms, most norms converge quickly to a stable value, with only 9 features having a non-negligible value, with the norms of the categorical features having the highest values. Note, however, that these norms cannot be used as a value for the importance of a feature. In fact, the norm value can only provide information on whether a feature should be included (higher than 0) or not. On the right, we show the case where $\lambda = 0.002$. Here 23 features are selected. Here, the low λ value leads to the model eliminating fewer features.

VI. Conclusions and Future Work

In this paper, we presented GLEm-Net a novel neural framework to reduce the dimensionality of data when mixed data consisting of both categorical and numerical features are available. GLEm-Net uses embedding layers to compress the amount of information of categorical features without losing information. By incorporating a revised version of the grouped Lasso penalty function, GLEm-Net efficiently reduces high-dimensional data consisting of mixed data types.

For this study, we use data collected in a real-world industry scenario with millions of records and both categorical and numerical features. We compare the performance of GLEm-Net with the performance of various state-of-the-art methods. Our results show that GLEm-Net has better classification performance than other state-of-the-art methods and selects a mixture of categorical and numerical features. Our results highlight the potential of the GLEm-Net framework for data reduction by applying feature selection to a mixture of data types, which ultimately improves model performance and the interpretability of neural networks.

Although the initial results of this study are promising, it is important to recognise that these are preliminary results. In the future, we plan to extend the evaluation of GLEm-Net to more open datasets. Expanding the experimental scope will strengthen the effectiveness of the framework in different data domains. In addition, we aim to integrate rule extraction methods into this framework to improve its interpretability.

References

[1] H. Zhang, J. Wang, Z. Sun, J. M. Zurada, and N. R. Pal, "Feature selection for neural networks using group lasso regularization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 4, pp. 659–673, 2020.

[2] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 1200–1205.

[3] D. M. Hawkins, "The problem of overfitting," *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[4] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[5] Z. Zhao, R. Anand, and M. Wang, "Maximum relevance and minimum redundancy feature selection methods for a marketing machine learning platform," in *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2019, pp. 442–452.

[6] D. Giordano, E. Pastor, F. Giobergia, T. Cerquitelli, E. Baralis, M. Mellia, A. Neri, and D. Tricarico, "Dissecting a data-driven prognostic pipeline: A powertrain use case," *Expert Systems with Applications*, vol. 180, p. 115109, 2021.

[7] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[8] D. Giordano, F. Giobergia, E. Pastor, A. La Macchia, T. Cerquitelli, E. Baralis, M. Mellia, and D. Tricarico, "Data-driven strategies for predictive maintenance: Lesson learned from an automotive use case," *Computers in Industry*, vol. 134, p. 103554, 2022.

[9] M. B. Kursa and W. R. Rudnicki, "Feature selection with the boruta package," *Journal of Statistical Software*, vol. 36, no. 11, p. 1–13, 2010.

[10] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1, pp. 389–422, 2002.

[11] F. G. F. Niquini, A. M. B. Branches, J. F. C. L. Costa, G. d. C. Moreira, C. L. Schneider, F. C. d. Araújo, and L. N. Capponi, "eliminating feature elimination and neural networks applied to the forecast of mass and metallurgical recoveries in a brazilian phosphate mine," *Minerals*, vol. 13, no. 6, 2023.

[12] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[13] I. Lemhadri, F. Ruan, L. Abraham, and R. Tibshirani, "Lassonet: A neural network with feature sparsity," *J. Mach. Learn. Res.*, vol. 22, pp. 127:1–127:29, 2019.

[14] E. Gasca, J. Sánchez, and R. Alonso, "Eliminating redundancy and irrelevance using a new mlp-based feature selection method," *Pattern Recognition*, vol. 39, no. 2, pp. 313–315, 2006.

[15] S. Solorio-Fernández, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "A survey on feature selection methods for mixed data," *Artificial Intelligence Review*, vol. 55, no. 4, pp. 2821–2846, 2022.

[16] C. Guo and F. Berkhahn, "Entity embeddings of categorical variables," *arXiv preprint arXiv:1604.06737*, 2016.

[17] F. Sigrist, "A comparison of machine learning methods for data with high-cardinality categorical variables," 2023.

[18] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[19] C. Guo and F. Berkhahn, "Entity embeddings of categorical variables," *CoRR*, vol. abs/1604.06737, 2016.

[20] P. Guo, "A frobenius norm regularization method for convolutional kernels to avoid unstable gradient problem," *CoRR*, vol. abs/1907.11235, 2019.

[21] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Training pruned neural networks," *CoRR*, vol. abs/1803.03635, 2018.

[22] A. V. Dorogush, V. Ershov, and A. Gulin, "Catboost: gradient boosting with categorical features support," 2018.

[23] K. Z. Mao, "Orthogonal forward selection and backward elimination algorithms for feature subset selection," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 629–634, 2004.

[24] C. Sammut and G. I. Webb, *Encyclopedia of machine learning*. New York: Springer Science & Business Media, 2011.