Machine Learning (2025) 114:246
https://doi.org/10.1007/s10994-025-06898-8

Semi-supervised learning from tabular data with autoencoders: when does it work?

Sintija Stevanoska[1,2] · **Jurica Levatić**[1] · **Sašo Džeroski**[1,2]

Received: 17 April 2025 / Revised: 4 August 2025 / Accepted: 17 September 2025

Abstract
Labeled data scarcity remains a significant challenge in machine learning. Semi-supervised learning (SSL) offers a promising solution to this problem by simultaneously leveraging both labeled and unlabeled examples during training. While SSL with neural networks has been successful on image classification tasks, its application to tabular data remains limited. In this work, we propose SSLAE, a lightweight yet effective autoencoder-based SSL architecture that integrates reconstruction and classification losses into a single composite objective. We conduct an extensive evaluation of the proposed approach across 90 tabular benchmark datasets, comparing SSLAE's performance to its supervised baseline and several other neural approaches for both supervised and semi-supervised learning, on varying amounts of labeled data. Our results show that SSLAE consistently outperforms its competitors, particularly in low-label regimes. To better understand when unlabeled data can improve performance, we perform a meta-analysis linking dataset characteristics to SSLAE's relative gains over its supervised baseline. This analysis reveals key properties—such as class imbalance, feature variability, and alignment between features and labels—that influence the success of SSL, contributing to a deeper understanding of when the inclusion of unlabeled data is beneficial in neural tabular learning.

Keywords Semi-supervised learning · Tabular data · Representation learning · Meta-analysis

Editors: Riccardo Guidotti, Anna Monreale, Dino Pedreschi.

✉ Sintija Stevanoska
sintija.stevanoska@ijs.si

[1] Department of Knowledge Technologies, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia

[2] Jožef Stefan International Postgraduate School, Jamova cesta 39, 1000 Ljubljana, Slovenia

Published online: 15 October 2025

1 Introduction

The current success of machine learning (ML) in the image and text domains is primarily driven by training deep models on large labeled datasets (Devlin et al., 2019; Krizhevsky et al., 2012; Vaswani et al., 2017; Yosinski et al., 2014). However, obtaining large quantities of labeled data is a laborious and expensive process, particularly for tasks that require domain expertise for correct annotation. On the other hand, a large amount of additional unlabeled data is often easily available.

Semi-supervised learning (SSL) has emerged as a promising solution to this issue. It uses both labeled and unlabeled data in the learning process, as opposed to only labeled data in supervised learning (SL) and only unlabeled data in unsupervised learning (Chapelle et al., 2006). SSL's unique advantage lies in using both kinds of data simultaneously. This one-step approach to the problem of insufficient labeled data typically improves upon the performance of models learned on labeled data only.

Recently, SSL research has focused on neural network (NN) based methods (Engelen & Hoos, 2020; Yang et al., 2023), which have proven to be especially powerful in processing image data (Chen et al., 2023; Zhang et al., 2024; Zheng et al., 2022). The SSL methods for images include regularization functions based on the data structure, often implemented using specific image-based augmentations, like rotation, cropping, and flipping (Sohn et al., 2020; Zheng et al., 2022). However, these augmentations do not have meaningful equivalents in tabular data. Additionally, NNs are very good at learning spatial dependencies and invariances in the data, properties not often found in tabular data, where features may be independent. These factors limit the translation of successful SSL NN-based methods to tabular data.

Despite these challenges, NNs remain an appealing option for tabular learning, as they offer the flexibility to learn smooth, nonlinear representations that capture dependencies across features. They are also well-suited for end-to-end optimization, support learning with multiple objectives (e.g., reconstruction + classification), and benefit from recent advances in representation learning and regularization techniques. Most of the recent NN-based SL and SSL approaches for tabular data rely on complex architectures or focus on pretraining methods (Arik & Pfister, 2021; Bahri et al., 2022; Chen et al., 2023; Darabi et al., 2021; Gorishniy et al., 2021; Hollmann et al., 2025; Huang et al., 2020; Somepalli et al., 2022; Yoon et al., 2020), which may not be effective with limited quantities of labeled data.

In an attempt to mitigate the aforementioned issues, in this work we propose SSLAE (Semi-Supervised Learning AutoEncoder), a simple yet effective autoencoder-based architecture for SSL from tabular data. Unlike existing SSL NN-based methods that rely on complex architectures or extensive pretraining, our approach leverages a global composite loss function that seamlessly integrates reconstruction and classification objectives, effectively utilizing both labeled and unlabeled data. Autoencoders are known in the ML community for their effectiveness in tasks such as dimensionality reduction and denoising (Vincent et al., 2008). Autoencoders consist of an encoder and a decoder. The encoder converts points in the input space to a compact and meaningful representation in a latent space, from which the original representation can be reconstructed by the decoder. The latent representation provides useful features by capturing important patterns in the data in an unsupervised manner. In our approach, we introduce a global composite loss function that combines the reconstruction and the classification losses as a weighted sum, enabling the simultaneous

processing of labeled and unlabeled data. For classification, we use a separate Multilayer Perceptron (MLP) with the latent space as input and the class labels as output. The labels predicted by the MLP are used to calculate the classification (supervised) loss on the labeled examples. The decoder provides the reconstructed features from the original space, on which the reconstruction (unsupervised) loss is calculated for all examples.

It is important to note that obtaining better performance by only incorporating unlabeled data is not always feasible or straightforward. The unlabeled data can only be beneficial if it conforms to the basic assumptions that SSL algorithms make about the relationship between the descriptive attributes and corresponding labels, such that a decision boundary should not lie in a high-density data region (Chapelle et al., 2006). A mismatch between the dataset structure and the algorithm's assumption can lead to predictive performance degradation (Fredriksson et al., 2025; Oliver et al., 2018). Due to this, and in contrast to supervised learning, there are no universally good semi-supervised methods, i.e., SSL methods tend to be domain or dataset dependent (Li & Liang, 2019). Despite the detrimental effects of these issues on SSL, it is generally still poorly understood under which conditions a particular semi-supervised algorithm is expected to perform well. To this end, we perform a meta-analysis to associate the characteristics of a given dataset with the performance of our proposed SSL algorithm on that dataset. This provides insights into why and when the proposed SSL approach performs better than its supervised counterpart, and more importantly, when it fails to do so. To the best of our knowledge, such meta-analysis is rarely performed in existing SSL research; therefore, our study can serve as a guide for future studies of this kind.

Main contributions: In this paper, we (1) adapt an autoencoder-based architecture with a global loss criterion that allows extension to the SSL domain, providing a simple and effective framework for learning from tabular data that outperforms state-of-the-art NN-based approaches for tabular data. While the use of autoencoders in SSL is not entirely novel in itself, we demonstrate that this simple architecture, when properly adapted, can be highly competitive in scenarios with limited labeled data. To support this claim, we (2) conduct an extensive empirical benchmark across 90 tabular datasets, evaluating the proposed approach against its supervised baseline and several NN competitors for SL and SSL from tabular data. Lastly, we (3) perform a meta-analysis of the results on the 90 datasets to identify which dataset characteristics influence the performance of our proposed SSL approach relative to its SL baseline, offering insights into when unlabeled data is beneficial.

The remainder of the paper is organized as follows: We begin by reviewing related work in Sect. 2 and discussing similar approaches that we use as competitors for our own. Next, in Sect. 3 we provide a formal definition of SSL and describe the details of our model and its workflow. The following Sect. 4 outlines our experimental setup, focusing on the data, metafeatures, and evaluation and implementation specifics. We then present our results and discuss our findings in Sect. 5. The paper concludes in Sect. 6 with a summary of our work, identifying its limitations and proposing directions for future research.

2 Related work

SSL has been a part of ML research for over two decades, with numerous methods proposed to address the scarcity of labeled data. The recent work in this field can be broadly categorized into two approaches: (1) augmenting the regularization function by combining supervised and unsupervised terms (Berthelot et al., 2019; Tarvainen & Valpola, 2017), and (2) using pseudo-labeling (Sohn et al., 2020; Xie et al., 2020). These approaches are particularly successful in the image classification domain.

2.1 SSL for image data

MixMatch (Berthelot et al., 2019) augments the supervised classification loss with an unsupervised consistency loss, encouraging the model to make consistent predictions for different augmentations of the same data point. MeanTeacher (Tarvainen & Valpola, 2017) augments the regularization function by using a teacher model, which is the exponential moving average of the student model. The supervised loss is combined with an unsupervised consistency loss to enforce the student model to predict similarly to the teacher model on the same inputs.

FixMatch (Sohn et al., 2020) combines pseudo-labeling with consistency regularization, using the most confident predictions of the model as pseudo-labels. It also applies strong data augmentations to enforce consistency between the original and augmented versions of the unlabeled data. Self Training with Noisy Student (Xie et al., 2020) is a method that uses a teacher model to generate pseudo-labels, and a student model is trained using these pseudo-labels with added noise.

All of the above SSL methods were developed for image data. Conversely, in tabular data, the features are typically independent and heterogeneous, and domain-specific transformations like those used in images are not applicable. This creates a fundamental limitation for applying image-based NN SSL techniques to tabular data.

2.2 SSL for tabular data

Several NN-based architectures have been developed specifically for tabular data in the SL context, with some extensions to semi- or self-supervised modes. Most recent developments for learning from tabular data rely on the transformer architecture and attention mechanisms.

TabNet (Arik & Pfister, 2021) is one such approach that uses a sequential attention mechanism to select relevant features for each decision step, offering interpretability. It does not rely on the positional encodings, instead using a soft feature selection process that allows the model to learn which features to focus on in the decision steps.

TabTransformer (Huang et al., 2020) focuses on the categorical features, using self-attention mechanisms to learn contextual embeddings for the categorical variables. The transformer layers learn relationships between categories, allowing the model to handle high cardinality categorical features. The TabTransformer and TabNet papers also explore the option to pretrain the model on all training data in an unsupervised (or rather, self-supervised) manner, and then fine-tune with the labeled part of the training data. FTTransformer (Gorishniy et al., 2021) uses the self-attention mechanism to process both continuous and

categorical data. It does not rely on any feature engineering. It directly applies transformers for processing tabular data, treating each feature as a token.

Gille et al. (2023) propose an SL autoencoder for Biomedical applications. Their approach learns from the labeled data incorporating both supervised and unsupervised losses in the loss criterion. It then uses the learned network to classify unlabeled examples by applying softmax directly to the latent space.

VIME (Yoon et al., 2020) consists of two components. Self-supervised learning, used for pretraining, reconstructs corrupted versions of the input to learn meaningful feature representations. Its semi-supervised phase is intended to improve the model's ability to extract meaningful representations from the unlabeled data by combining the supervised and consistency losses.

SCARF (Bahri et al., 2022) applies contrastive learning for tabular data in a self-supervised learning setting, generating augmented views of the tabular data by perturbing it and training the model to differentiate between similar (positive) and dissimilar (negative) samples. Its goal is to maximize the agreement between the augmented views, learning robust feature representations. SAINT (Somepalli et al., 2022) is a transformer-based model that introduces two types of attention: self-attention for feature relationships within a sample and intersample attention for learning the interactions between different samples. It can work in supervised mode, and it can also be pretrained and fine-tuned when there are few labeled examples.

ReConTab (Chen et al., 2023) is a framework designed to learn robust representations for tabular data, combining reconstruction loss with contrastive loss to improve feature extraction. ContrastiveMixup (Darabi et al., 2021) extends contrastive learning by mixing data points, creating interpolations of positive and negative samples. This forces the model to learn more refined decision boundaries by encouraging similarity between mixed representations.

In this paper, we propose an extension of autoencoders to a semi-supervised setting. Autoencoders are particularly suitable for SSL because they are easily adaptable to exploit both labeled and unlabeled data during training. They use the underlying structure of the unlabeled data to learn meaningful representations, which can improve the performance on the classification task when few labeled data points are available. By combining an unsupervised autoencoder with a supervised classification head, our model is able to simultaneously learn from both labeled and unlabeled data during training. Unlike VIME or TabNet, which rely on data corruption or feature selection mechanisms, our model directly learns from the latent representations of both labeled and unlabeled data, providing a simple solution for handling the complex and heterogeneous nature of tabular data, particularly in a small labeled data size scenario, which is common in real-life tabular datasets.

3 Semi-supervised autoencoder

In this section, we first describe the general task of Semi-Supervised Learning (SSL), and then describe the proposed semi-supervised autoencoder architecture.

3.1 General formulation of SSL

Consider a descriptive space $\mathcal{X}$ consisting of K descriptive variables, $\mathcal{X} = (x_1, ..., x_K)$, and a target space $\mathcal{Y}$, which can be binary, discrete or continuous. We are provided with two sets of examples. The first set, D_l, is the labeled data, where each example is a pair of one element from $\mathcal{X}$ and one element from $\mathcal{Y}$ (the example's label), i.e., $D_l = \{(x_i, y_i) : x_i \in \mathcal{X}, y_i \in \mathcal{Y}, 1 \leq i \leq N_l\}$, where N_l is the number of labeled examples. The second set, D_u, is the unlabeled data, which contains only examples from $\mathcal{X}$, given as $D_u = \{x_i : x_i \in \mathcal{X}, 1 \leq i \leq N_u\}$, where N_u is the number of unlabeled examples.

The task is to find a function f: $\mathcal{X} \rightarrow \mathcal{Y}$, by using both D_l and D_u, such that f minimizes a loss function. In this work, we use a weighted composite loss function defined in Sect. 3.3.

3.2 Motivation

The goal of our method is to learn a model that predicts the labels of examples in a tabular dataset using both labeled and unlabeled data. At the core of our approach is an autoencoder, which excels at learning compressed representations of data. We extend this to a semi-supervised framework by introducing a composite loss function that balances the unsupervised reconstruction loss (which helps in understanding the overall data structure) and a supervised classification loss (which helps with predicting the correct labels for the data).

The encoder learns meaningful representations from both labeled and unlabeled data in a latent space, capturing important patterns even when labels are not available. A Multilayer Perceptron (MLP) further processes these representations to predict the labels. The decoder attempts to reconstruct the input data, ensuring that the proposed approach retains general information about the distribution of the data.

By using this combined approach, we achieve two important objectives. First, we utilize the unlabeled data: the reconstruction loss helps our model learn general representations from unlabeled data. Second, we improve the predictive accuracy: the classification loss uses the labeled data to guide the proposed model toward learning patterns that are predictive of the target labels. With this high-level understanding of our approach, we now proceed to describe the architectural details of SSLAE.

3.3 Architecture

Our proposed SSLAE (Semi-Supervised Learning AutoEncoder) consists of three primary components: an encoder, a decoder, and an MLP classifier. The overall workflow is illustrated in Fig. 1. The method uses two data sources: an unlabeled dataset D_u and a labeled dataset D_l. These datasets are combined into training batches according to the labeled-to-unlabeled ratio in the training dataset.

The input features x are processed by the encoder, producing a latent space z. For each dataset, the dimension of the latent space is estimated using the TwoNN algorithm (Facco et al., 2017), which approximates the local intrinsic dimension (ID) by analyzing the distances to the nearest neighbors, generating a cumulative distribution. The latent representation is used in two separate processing pipelines.

In the first pipeline, depicted in the lower right part of Fig. 1, the latent features z are passed through an MLP, followed by a softmax layer to convert the logits into the predicted

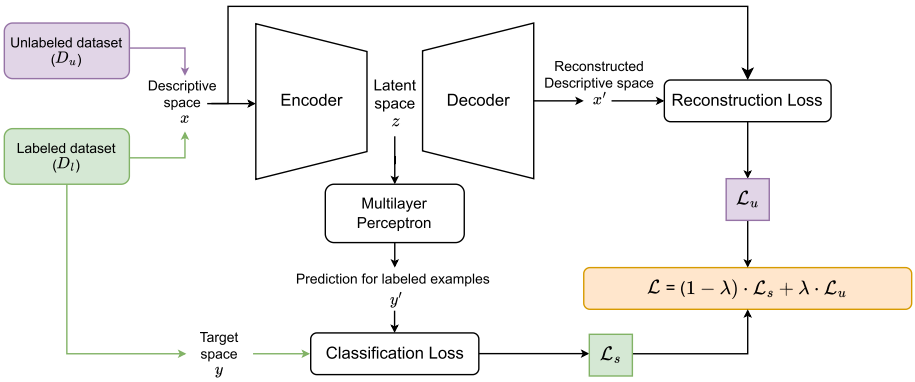

Fig. 1 The architecture of the semi-supervised autoencoder

labels y' for the labeled data that comes from D_l. The supervised loss $\mathcal{L}_s$ is calculated as the cross-entropy between the true labels y and the predicted labels y', only propagating the loss from the labeled examples. Once all labeled examples are exhausted, the remaining batches consist only of unlabeled examples.

In the second pipeline, depicted in the upper right part of Fig. 1, the latent features are passed through the decoder, which reconstructs the original feature space, producing x'. The unsupervised loss $\mathcal{L}_u$ is computed as the reconstruction loss, namely the Smooth L_1 loss function. This criterion combines an L_2 loss for element-wise errors below 1 with L_1 loss for larger errors, measured as the absolute difference between the original input x and the output of the decoder x'.

Composite loss function. This brings us to the definition of the composite loss $\mathcal{L}$ used to train the model proposed in this paper, defined as:

$$\mathcal{L} = (1-\lambda)\cdot\mathcal{L}_s(y,y') + \lambda\cdot\mathcal{L}_u(x,x') \tag{1}$$

where $\mathcal{L}_s$ is the supervised loss, $\mathcal{L}_u$ is the unsupervised loss, and λ is the parameter that determines the amount of supervision. Theoretically, when $\lambda = 1$, we rely only on the reconstruction loss $\mathcal{L}_u$ for training the proposed model, making the learning process unsupervised. The opposite is true when $\lambda = 0$, meaning we disregard $\mathcal{L}_u$ and only use the supervised loss $\mathcal{L}_s$ for training, performing supervised learning.

We use the supervised version of the proposed approach as a baseline and refer to it as SLAE (Supervised Learning AutoEncoder). It is trained only on examples from D_l. Using exactly the same architecture to perform supervised learning is instrumental in providing a fair comparison between the proposed SSL approach and its supervised baseline, as suggested by Oliver et al. (Oliver et al., 2018). The primary goal of any SSL algorithm is to outperform, or at least perform as well as, its SL counterpart by using the information present in the unlabeled data.

For the semi-supervised version, we set λ to 0.5. It is important to note that during the initial development phase of our approach, we optimized this parameter for each dataset, but the performance gain of optimizing λ was marginal compared to the time that was needed to obtain the optimal parameter value. We found that the fixed value of λ=0.5 already delivers

competitive performance. We refer the reader to Appendix A for an ablation study of the influence of the λ parameter on the performance.

Encoder and decoder. In our implementation, the encoder and decoder are symmetrical. They consist of two fully connected linear layers with a ReLU activation function. The input layer of the encoder and the output layer of the decoder correspond to the number of features of the dataset. The first layer of the encoder projects from the input dimension to the hidden dimension, whose size is the mean of the input and output layer dimensions, gradually lowering the layer size before reaching the latent space with a dimension corresponding to the estimated intrinsic dimension.

Multilayer perceptron. The MLP is the block that handles the classification. Its input layer size corresponds to the estimated intrinsic dimensionality and the output layer size corresponds to the number of classes in the dataset. Similar to the encoder and decoder, the first layer projects from the input to the hidden dimension, whose size this time corresponds to the sum of the input and output layer dimensions. The second layer projects from the hidden dimension to the output dimension, followed by softmax to obtain class probabilities.

In sum, the proposed SSLAE approach offers a unified framework that uses both labeled and unlabeled data to enhance predictive performance. By integrating an encoder-decoder architecture with a classification MLP, SSLAE simultaneously captures the underlying data structure and optimizes for classification accuracy. The use of a composite loss function, balanced by the parameter λ, allows for flexible adjustment between supervised and unsupervised learning objectives. Setting λ to 0.5 provides an effective trade-off between the two objectives, harnessing the benefits of both learning paradigms without extensive hyperparameter tuning. By deriving the sizes of the layers of the encoder, decoder, and MLP directly from the data characteristics, specifically by using the dataset's feature count, estimated intrinsic dimension, and number of classes, we eliminate the need for manual tuning of these parameters.

4 Experimental setting

The experimental study is designed to answer the following research questions:

1. *Performance comparison: How does the proposed SSLAE perform as compared to its SL baseline and other NN-based algorithms for SL and SSL from tabular data?*
   Understanding the practical benefits and limitations of the proposed SSLAE model requires a thorough evaluation against its SL baseline and other state-of-the-art NN-based algorithms for both SL and SSL from tabular data. To this end, we conduct an extensive empirical comparison on 90 benchmark datasets. This comparison is essential to find out whether incorporating unlabeled data through SSLAE leads to tangible improvements over existing NN tabular approaches.
2. *Contextual efficacy: What are the dataset characteristics that influence the relative performance of our SSL approach as compared to its SL version?*
   By pinpointing these factors, we can understand the contexts in which SSL is most effective and using the unlabeled data helps, providing guidance for practitioners on when to choose SSL techniques over standard SL methods when considering NNs. To

this end, we extract metafeatures describing the 90 datasets and relate them to the relative performance of SSLAE (to its competitors) on these datasets.

In this section, we detail the experimental setup designed to address these questions. We first describe the datasets used, the preprocessing steps taken, and the SSL examination scenario where we vary the amount of labeled data. We then describe the evaluation strategy used, including the statistical methods for comparing algorithm performance. We finally describe the meta-analysis performed, including the extraction of metafeatures, the measure of (relative) performance used, and the meta-analysis methodology itself.

4.1 Data

We evaluated the proposed approach on 90 benchmark datasets for single-target classification, both binary and multi-class. The datasets were mainly sourced from the OpenML (Vanschoren et al., 2013), UCI (Kelly et al., 2019), and PMLB (Olson et al., 2017) repositories. We selected datasets from different domains, with no (or minimal) number of missing values in the feature space. We focused on datasets that have at least 900 examples so we can simulate the relevant SSL scenario where the unlabeled data size is large relative to the labeled data size. The datasets and their characteristics are listed and visualized in Appendix B. Most of the datasets have between 1000 and 2000 examples, with the smallest dataset containing 932 samples, and the largest dataset 28,056 samples. Most of the datasets have fewer than 20 features, and 55 out of 90 datasets concern binary classification (35 concern multi-class classification).

While we attempted to compose a diverse collection of datasets, biases in our dataset selection may exist, limiting our conclusions concerning other domains not represented in the selected datasets. Despite these potential biases, we believe our results remain applicable in the specific scenario we are exploring with the particular range of dataset sizes.

We split the datasets into training, validation, and testing sets in a stratified manner, preserving the overall proportions of the class values. One-third (33%) of each dataset is reserved as the testing set, and the remaining two-thirds (67%) comprise the training set. We further set aside 15% of the training set as a validation set.

Data preprocessing. We applied a very simple data preprocessing pipeline. First, we one-hot encoded the categorical variables to represent the discrete categories as binary vectors. Missing data was imputed with the mean value for the numerical features and the mode for the categorical features, so the datasets can be used without discarding rows or columns with missing data. The numerical features were then standardized by subtracting the mean and scaling to unit variance, to prevent bias towards features with higher magnitudes in the learning process.

Creating unlabeled and labeled datasets. To simulate an SSL scenario, we created labeled and unlabeled subsets of the training data, focusing on settings with very few labeled examples. For each dataset, we created five labeled subsets, with 50, 100, 200, 350, and 500 labeled examples. To maintain a realistic scenario, the labeled subsets were created incrementally: each larger subset contained all the examples from the previous, smaller subsets. For example, the 100 labeled examples set contains the 50 labeled examples from the first subset, adding 50 new examples to reach the 100 total. The remaining data points in the training set for each increment had their labels removed and were treated as unlabeled

examples. The selection of labeled and unlabeled examples was done in a stratified manner, ensuring that the class distributions in both the labeled and unlabeled sets match the training data class distribution.

4.2 Evaluation strategy

To address the first research question, the proposed algorithm and its competitors were executed using 10 different random seeds (for each of the varying amounts of labeled data), which influenced both the initial weights in the network (where possible) and the division of data into training, validation and testing sets. The results from all 10 trials for each algorithm on each dataset for each amount of labeled examples were averaged and recorded for analysis. We report the macro-averaged (across different class values) Area Under the Precision Recall Curve (AUPRC Macro) as the primary performance metric, which gives equal weight to (the AUPRC of) each class in the calculation of the average.

We compared our proposed approach to 7 other NN-based algorithms designed for learning from tabular data. These include: the SL baseline of our approach; the VIME (Yoon et al., 2020) algorithm in self- and semi-supervised modes (VIME-SelfSL and VIME-SSL, respectively); TabNet (Arik & Pfister, 2021) in both supervised and self-supervised modes (TabNet and TabNet-SelfSL respectively); and TabTransformer (Huang et al., 2020) and FT-Transformer (Gorishniy et al., 2021), both in supervised mode. We used the default hyperparameters for each of the competing algorithms. Please refer to Appendix C for a hyperparameter optimization study, where the effect of tuning the models is explored. In terms of preprocessing, our approach, its SL baseline, and VIME work with the data preprocessed as described in Sect. 4.1. For TabNet, TabNet-SelfSL, TabTransformer, as well as FT-Transformer, following the implementation and the provided examples in the respective code repositories, the numerical features were not standardized and the categorical features were label encoded.

Please refer to Appendix D.1 for further implementation details.

Statistical Analysis. For the statistical performance comparison of the proposed SSLAE with its competitors, we used the Bayesian t-test (Benavoli et al., 2017). This approach offers a probabilistic interpretation of the data, estimating the probability distributions of the differences in performance between models. This allows us to determine the likelihood that one method will outperform another on a new dataset similar to those used in our study.

An important parameter in the Bayesian t-test is the Region of Practical Equivalence (ROPE), which is defined as the range of performance differences where the compared approaches are considered practically equivalent. In our case, we set ROPE to 0.01, meaning that the algorithms are considered to perform equally well if the difference in their AUPRC Macro is below 1%. The Bayesian t-test computes the posterior that the difference in performance falls within the ROPE.

4.3 Meta-analysis

To address our second research question, we have conducted a meta-analysis of the obtained results. This analysis aims to identify which properties of the datasets are most predictive of SSLAE outperforming its SL counterpart. By understanding these characteristics, we

can provide insights into the contexts where SSLAE (and in general SSL), offers the most benefit.

Metafeature extraction. The metafeatures were extracted using the Python library PyMFE (Alcobaça et al., 2020), which is designed specifically for tabular datasets. The metafeatures are grouped into the following categories: (1) General dataset metafeatures, such as the number of instances, features, and classes, (2) Statistical measures that describe the numerical properties of the dataset, (3) Landmarking measures that assess the performance of simple learning algorithms on the dataset, (4) Clustering measures that provide insights into cluster performance, (5) Concept measures that estimate the variability of the class labels among instances, and (6) Complexity measures that estimate the difficulty in separating the data points into their expected classes. Some of these measures are defined as per-example or per-feature measures, so they can return more than one value. In these cases, the package provides options to aggregate the values and provide a single number per dataset. In our analysis, we used the minimum, maximum, and mean values for aggregation. All metafeatures that had at least one missing value across datasets, had the same value across datasets, or contained outliers, were not considered for further analysis, which left 36 metafeatures for exploration. It is important to note that before extracting the metafeatures, the numerical features in the datasets were scaled. The extracted metafeatures provide the feature set for the meta-analysis.

Relative performance measure. To quantify the performance of SSLAE relative to SLAE, we defined the *Performance Area* metric as the difference between the areas under the learning curves of SSLAE and SLAE. Formally:

$$PerformanceArea = \int_{50}^{500} SSLAE(n)dx - \int_{50}^{500} SLAE(n)dx \tag{2}$$

where *SSLAE*(*n*) and *SLAE*(*n*) are the AUPRC Macro values for the respective models, at a given number of labeled examples *n*. A positive *Performance Area* means that the SSLAE model outperforms its SL baseline, while a negative *Performance Area* means that the SSLAE performs worse, overall, for that dataset. The metric contributes towards obtaining a robust estimation of the performance difference comparable across datasets. This metric is the meta-label of our analysis.

Metafeature importance analysis. To construct the meta-analysis dataset, we concatenated the extracted metafeatures with the corresponding *Performance Area* for each dataset. We formulated the meta-analysis task as a binary classification problem. The datasets with *Performance Area* > 0 were labeled as 1 (indicating SSLAE outperforms SLAE), while those where *Performance Area* ≤ 0 were labeled as 0. For the classification task, we applied the Random Forest (Breiman, 2001) approach. We trained the Random Forest classifier on the meta-analysis training set.

We used the Permutation Importance (PI) score (Breiman, 2001) to estimate the metafeature importance. This score quantifies the impact of permuting a feature on the performance of a model, with a larger deterioration in performance indicating a more important feature. To ensure the robustness of the feature importance results, we repeated this methodology for 10 random seeds, affecting both the data splits and the initializations of the Random Forest and the PI score.

For the PI-based feature importance estimates to be considered reliable, it is important to remove correlated metafeatures before training the Random Forest classifier used to estimate the PI (Gregorutti et al., 2017). When features are highly correlated, permuting one may have minimal impact on model performance. This is because the model can still get the information from the correlated features, leading to potentially underestimating the importance of individual features. One way to address this issue is by applying hierarchical clustering to group correlated features into clusters and then selecting a single representative from each cluster. In our case, hierarchical clustering resulted in six distinct clusters, from which we retained the feature with the highest variance within each cluster.

To gain deeper insights into the relationships between selected metafeatures, we trained a Decision Tree classifier on the chosen metafeatures, predicting the *Performance Area*. The Decision Tree visualizes the interaction between the most influential metafeatures and their impact on the relative performance of SSLAE versus SLAE.

Further implementation details on the metafeature extraction and selection processes can be found in Appendix D.4.

5 Results and discussion

In this section, we discuss the results, addressing both questions proposed in Sect. 4.

5.1 Performance comparison

To see how the proposed SSLAE compares to its SL baseline and other NN-based SL and SSL algorithms for tabular data, we analyze the average AUPRC Macro scores, for each algorithm across all datasets, per number of labeled examples, as outlined in Sect. 4.2. Additionally, we compute the average rank of each algorithm across all datasets. Finally, we conduct a Bayesian t-test for statistical performance comparison.

5.1.1 AUPRC macro scores and average ranks

As shown in Table 1, SSLAE achieves the highest average AUPRC Macro scores in all cases, i.e., for 50, 100, 200, 350, and 500 labeled examples, indicating its advantage when labeled data is scarce. In terms of ranking, SSLAE consistently maintains the best average rank across the different numbers of labeled examples.

SLAE, the SL baseline of our proposed method, shows competitive performance. As the number of labeled examples increases, the gap in average AUPRC Macro scores between SLAE and SSLAE narrows. Note that, while competitive, the performance of SLAE is never better than the performance of SSLAE. The average ranks for SLAE also reflect this trend, with ranks improving slightly as the number of labeled examples increases, but never surpassing SSLAE's rank.

The two *VIME variants*, VIME and VIME-SelfSL, show moderate performance across all labeled data sizes. Despite improvements with the increase in the number of labeled examples, both VIME variants do not surpass SSLAE in terms of average AUPRC Macro scores or ranks. One explanation for this lies in how each model learns from the unlabeled data. SSLAE uses a composite loss function that equally balances the classification and

Table 1 The average value of the AUPRC Macro score across all datasets for the algorithms is given in the Algorithm column, which can be either SL or SSL (as specified in the Type column), for each number of labeled examples

		AUPRC Macro					Average ranks					
Type	Algorithm	50	100	200	350	500	50	100	200	350	500	Avg
SL	SLAE	$66.3_{5.0}$	$71.3_{4.1}$	$77.4_{3.1}$	$80.3_{2.9}$	$81.4_{2.0}$	3.58	3.66	3.37	3.50	3.51	3.52
	TabNet	$44.3_{6.0}$	$56.0_{7.8}$	$68.8_{7.6}$	$76.7_{5.0}$	$80.5_{3.3}$	7.29	6.78	6.08	5.27	5.01	6.08
	TabTransformer	$63.6_{3.9}$	$65.8_{3.6}$	$70.3_{3.0}$	$73.0_{2.6}$	$74.2_{2.3}$	4.22	4.77	4.82	5.04	5.08	4.79
	FT-Transformer	$63.8_{4.9}$	$67.7_{4.2}$	$73.0_{3.3}$	$76.6_{2.9}$	$78.4_{2.7}$	4.06	4.37	4.44	4.41	4.51	4.36
SSL	TabNet-SelfSL	$47.0_{6.6}$	$58.7_{8.0}$	$69.4_{7.6}$	$77.5_{5.2}$	$80.6_{3.8}$	6.71	6.30	5.84	5.33	5.03	5.84
	VIME-SelfSL	$65.7_{6.8}$	$70.5_{4.5}$	$74.6_{3.6}$	$77.9_{3.0}$	$79.8_{2.7}$	4.10	4.19	4.74	5.07	5.24	4.67
	VIME-SSL	$66.7_{5.6}$	$71.5_{4.3}$	$75.7_{3.3}$	$78.5_{2.7}$	$79.9_{2.8}$	3.59	3.62	4.04	4.38	4.68	4.06
	SSLAE (ours)	$\mathbf{69.3}_{4.6}$	$\mathbf{74.5}_{3.6}$	$\mathbf{78.7}_{2.7}$	$\mathbf{81.0}_{2.6}$	$\mathbf{81.9}_{2.0}$	**2.46**	**2.32**	**2.65**	**3.00**	**2.94**	**2.67**

The values in the subscript show the standard deviation. Average ranks are shown separately on the right. Best performances are in bold—higher is better for AUPRC Macro, lower is better for ranks

reconstruction losses. In contrast, VIME relies on masked feature reconstruction, data augmentations, and consistency loss, which may not capture the relationships in such small numbers of labeled examples.

TabNet and TabNet-SelfSL show significant improvements as the number of labeled examples increases. At smaller data sizes (50 and 100 labeled examples), their performance is considerably lower than that of SSLAE and all other baselines. This suggests that TabNet and its self-supervised variant benefit substantially from larger amounts of labeled data. Their average ranks improve correspondingly at higher data sizes, although they remain worse than those of SSLAE overall. TabNet's architecture incorporates feature selection through sequential attention. As more labeled data become available, this possibly allows TabNet to focus on the most informative features, improving its performance.

TabTransformer and FT-Transformer achieve competitive average AUPRC Macro scores across all labeled data regimes. Their performance improves gradually with more labeled examples but remains behind SSLAE, SLAE, and the VIME variants. Their average ranks are on par with both VIME variants across all data sizes, but never higher than SLAE or SSLAE. Transformer-based models like TabTransformer and FT-Transformer have high capacity and are prone to overfit when the labeled datasets are small. This can lead to poorer generalization on unseen data, especially for scarce-label regimes.

These results highlight the effectiveness of SSLAE in using unlabeled data to enhance performance, particularly when labeled data size is very limited. The diminishing performance gap between SSLAE and SLAE as the number of labeled examples increases suggests that the advantage of SSL decreases when more labeled data points are available. This observation aligns with the expectation that semi-supervised approaches are most beneficial in scarce-label regimes. To contextualize the performance of SSLAE beyond neural baselines, we additionally compared it to non-neural models, specifically Random Forests (RF), XGBoost, LightGBM, and CatBoost, in both supervised and semi-supervised learning, under the same scarce-label settings. Out of the 90 datasets, SSLAE outperforms RF on 28 datasets, XGBoost on 44, LightGBM on 40, and CatBoost on 13 datasets. The full comparison of SSLAE to the tree-based methods is provided in Appendix F.1.

5.1.2 Bayesian t-test for performance comparison

To further substantiate our claims, we estimated the statistical significance of the difference in performance between the proposed approach and the competing algorithms. We conducted a Bayesian t-test, as outlined in Sect. 4.2. The results are summarized in Table 2, showing the significance of the difference between the algorithms when considering the AUPRC Macro score. For each competing algorithm listed in the *Algorithm* column, the table presents two probabilities across different numbers of labeled examples: *p(algo)*, the probability that the competing algorithm performs better than SSLAE, and *p(rope)*, the probability that the performance difference between the competing algorithm and SSLAE is within the 1% ROPE.

The Bayesian t-test results provide further statistical evidence that SSLAE outperforms its competitors, especially when labeled data are scarce. The only notable exception is SLAE, our SL baseline, which shows increasing *p(rope)* values as the labeled data size grows, converging towards the 1% ROPE with SSLAE. VIME (both versions), TabNet

Table 2 Bayesian t-test results comparing the performance (in terms of AUPRC Macro) of SSLAE against competing algorithms across different numbers of labeled examples

Algorithm	Number of labeled examples									
	50		100		200		350		500	
	p(algo)	p(rope)	p(algo)	p(rope)	p(algo)	p(rope)	p(algo)	p(rope)	p(algo)	p(rope)
SLAE	0.00	0.00	0.00	0.00	0.00	0.6279	0.00	0.9992	0.00	0.9998
VIME-SelfSL	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0001
VIME-SSL	0.0005	0.00	0.0001	0.00	0.0001	0.00	0.0004	0.00	0.00	0.0006
TabNet	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0001	0.00
TabNet-SelfSL	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0002
TabTransformer	0.0006	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
FT-Transformer	0.00	0.00	0.00	0.00	0.00	0.00	0.0008	0.00	0.0004	0.00

The results are rounded to two decimals when the result is exactly 0 or 1; four decimal places are provided otherwise

(both versions), TabTransformer, and FT-Transformer consistently show low *p(algo)* relative to SSLAE, with no evidence of practical equivalence.

5.2 Contextual efficacy

To identify the key factors that influence the performance of SSLAE compared to SLAE, we conducted a metafeatures analysis using permutation importance (PI), as detailed in Sect. 4.3. To provide deeper insights, Fig. 2a–f shows the scatter plots of the datasets across the original *Performance Area* (shown on the *y*-axis) and the metafeature in question (shown on the *x*-axis).

5.2.1 The importance of metafeatures for the relative performance

The key metafeatures are summarized below, with the metafeature aggregation type shown in brackets and their PI score noted. Please consult Appendix G for exact details on how these metafeatures are calculated.

Canonical correlation (min), PI=0.161

This metafeature evaluates the strength of linear relationships between the feature set and the target labels through Canonical Correlation (CC) Analysis. It identifies linear combinations of the features that are most correlated with linear combinations of the one-hot encoded labels. The minimum CC captures the weakest (worst-case) relationship between linear combinations of the features and linear combinations of the one-hot encoded target labels. A high minimum value indicates that all canonical correlations in the dataset are strong, meaning the target space is well aligned with every linear projection of the feature space.

Figure 2a shows a negative slope, with *p-value* = 0.0002, suggesting a statistically significant insight: when the relationships between features and the target are strongly linear and aligned (high CC), SLAE can learn more effectively from the labeled data alone, without there being a significant benefit from including unlabeled data. In contrast, when the minimum CC is low (i.e., at least one linear projection shows a weaker or more complex relationship to the target), SSLAE can use the unlabeled data to learn richer representations and improve performance.

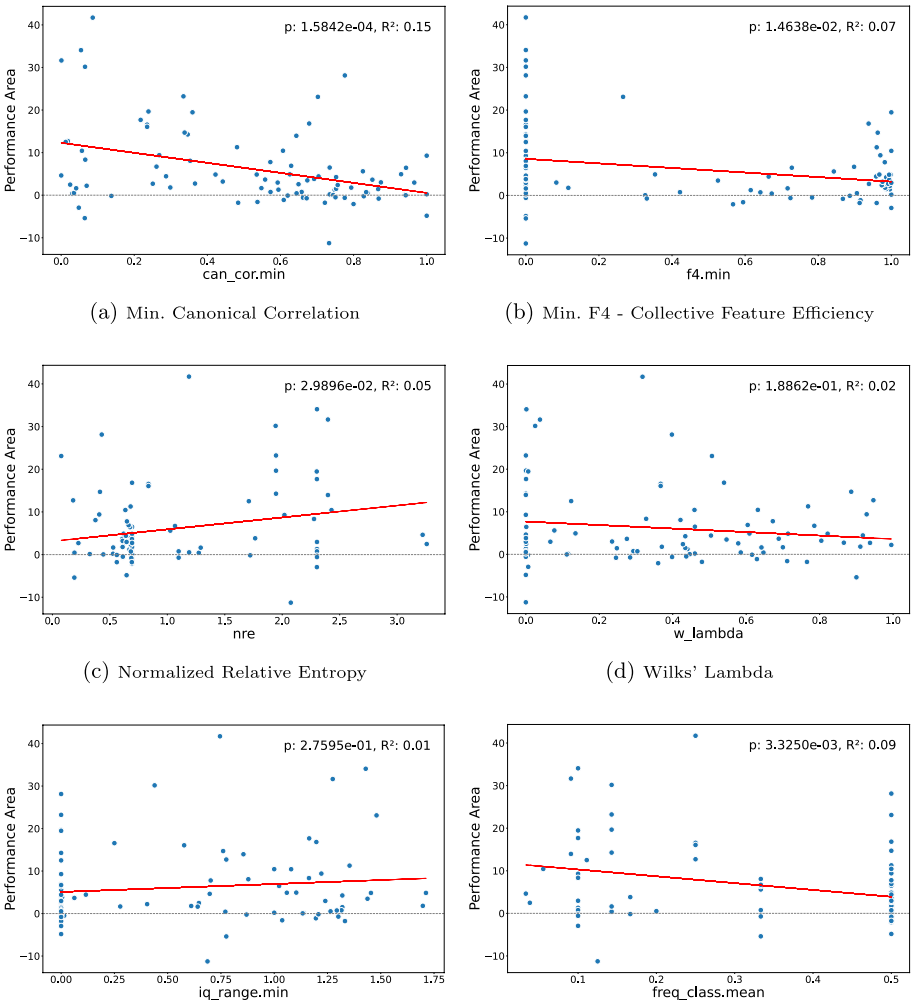

Fig. 2 Scatter plots of the key metafeatures depicting the relationship between the metafeature (*x*-axis) and *Performance Area* (*y*-axis). The *red regression line* shows the slope of the correlation. The statistical significance of the slope in terms of *p-value* and R^2, as estimated by the t-test, is reported at the top right corner of each figure. The *dashed horizontal line* shows where SSLAE begins to outperform SLAE (values above 0 indicate a positive *Performance Area*, translating to SSLAE performing better than its baseline)

F4 - Collective feature efficiency (min), PI=0.158

The Collective Feature Efficiency metafeature assesses how well the ensemble of features in a dataset works together to separate the classes. It iteratively measures class overlap in the feature space, progressively eliminating the least overlapping features and the instances that don't overlap under those features, until no features or instances remain. The F4 score for each pair of classes is defined as the fraction of remaining overlapping instances relative to the entire dataset. A low F4 indicates that most instances are well-distinguished by at least

one feature, while a high F4 points to extensive overlap, implying that no subset of features can clearly separate the instances.

The scatter plot in Fig. 2b, with *p-value* = 0.01, reveals a statistically significant negative correlation between the minimum value of F4 and the performance difference between SSLAE and SLAE. This indicates that, for datasets with high feature overlap (high F4), the gain of using SSLAE over SLAE diminishes. In other words, there is limited advantage from incorporating unlabeled data when there is at least one pair of classes the features cannot discriminate between. Conversely, for datasets with low overlap and effective feature spaces (low F4), using unlabeled data is beneficial.

Normalized relative entropy, PI=0.085

The Normalized Relative Entropy (NRE) is a metafeature that quantifies the uniformity of the class distribution within a dataset. It first calculates the relative frequency of each class, providing a probability distribution of the instances across classes. Then, it calculates the entropy of this probability distribution to measure how far the class distribution is from a uniform one, where each class would have an equal number of instances. A higher NRE indicates a less uniform class distribution, suggesting the presence of dominant and underrepresented classes.

The positive correlation in Fig. 2c, with *p-value* = 0.03, shows a statistically significant trend: SSL helps more for datasets with imbalanced class distributions (higher NRE). When the labeled set is small, minority classes may be represented by very few labeled points, making it difficult for SLAE to learn accurate decision boundaries. In contrast, SSLAE can incorporate unlabeled instances into the latent representation through the autoencoder's reconstruction loss, helping to shape the latent space in a way that allows the classifier to refine decision boundaries, particularly around minority classes.

Wilks' Lambda, PI=0.084

Wilks' Lambda (WL) is a statistical metafeature that quantifies the proportion of the total variance in the data not explained by class separation. WL is based on CC, computing the canonical eigenvalues associated with each canonical correlation, and then aggregating them through a product-based transformation. It differs from the CC (min) metafeature, which focuses on the weakest linear alignment between features and labels, by providing a single summary statistic that captures the overall strength of all canonical correlations in the dataset. Consequently, a dataset may have a fairly high minimum CC (implying no dimension is extremely weak), yet still not achieve an exceptionally low WL if its remaining dimensions are only moderately strong. WL thus provides a compact way to measure class separability: if all canonical correlations are high, the eigenvalues are large, resulting in low WL, which corresponds to strong linear class separability. In contrast, higher WL values suggest greater class overlap.

In Fig. 2d, we observe a negative correlation between WL and the performance gain of SSLAE over SLAE, although the slope is not statistically significant (*p-value* = 0.19). Nevertheless, the plot shows that the datasets that have the highest *Performance Area* tend to fall within the WL range of 0.0 to 0.5. This suggests that the largest gains from using unlabeled data occur when WL is low, and the datasets have clearer class separability. In such cases, unlabeled examples are more likely to be near their true classes in the latent space, enabling SSL to refine decision boundaries. Conversely, when class boundaries are less well-defined, including unlabeled data may introduce noise and diminish the effectiveness of SSL.

Interquartile range (min), PI=0.047

The Interquartile Range (IQR) is a statistical measure that quantifies the dispersion of numerical features by calculating the difference between the 75th and 25th percentiles. It reflects the variability of feature values within the dataset, with higher IQR indicating richer patterns and greater variability in the data.

In Fig. 2e, we observe a positive correlation between the performance gain of SSLAE over SLAE and the minimum IQR across features, although the relationship is not statistically significant (*p-value* = 0.28). The plot shows that the datasets that benefit most from SSLAE have higher IQR, implying that SSL methods can better use the unlabeled data when even the least variable feature exhibits sufficient spread, allowing the model to capture the rich patterns in the latent space.

Relative frequency of each distinct class (mean), PI=0.023

This metafeature calculates the average relative frequency of the classes within a dataset, quantifying class balance. Figure 2f shows that the maximum value of average relative frequency metafeature is 0.5. Upon closer inspection of the datasets, we find that all of the binary classification datasets have an average relative frequency of 0.5, revealing that in this particular case, this metafeature differentiates between the *Performance Area* of binary and multiclass classification datasets.

From the statistically significant slope on Fig. 2f (*p-value* = 0.003), we observe that there is a negative correlation between this metafeature and the performance gain of using unlabeled data. This suggests that SSLAE can extract more information from the unlabeled data when multiple classes are present in the dataset, where learning richer representations becomes increasingly important due to more complex decision boundaries.

5.2.2 Interplay of metafeatures

When considering metafeatures independently, we observed that SSLAE is more likely to outperform SLAE when: 1) the dataset exhibits a complex relationship between the feature and target spaces in at least one linear projection (low minimum CC); 2) the overall separability of the feature space remains strong (low WL); 3) class overlap is not extreme (low F4 values); 4) class distribution is non-uniform (high NRE); 5) the dataset contains multiple classes rather than just two (high IQR).

To further understand how these metafeatures interact, we trained a Decision Tree on the meta-analysis dataset, shown in Fig. 3. This tree provides a visual representation of the decision-making process, illustrating how combinations of metafeatures contribute to distinguishing between cases where SSLAE outperforms SLAE and vice versa. The meta-analysis of the relative performance of SSLAE vs. all other NN-based baselines is presented in Appendix E, while the meta-analysis for SSLAE vs. the tree-based baselines is presented in Appendix F.2.

The root node splits on minimum CC, reaffirming that this metafeature is the first factor in determining which model performs better, as shown previously with the PI score. The left child node, representing datasets with low minimum CC, contains 32 out of 71 positive examples, indicating that SSLAE benefits from including unlabeled data when relationships between features and target are more complex.

The second split occurs on the F4 metafeature, further classifying 12 of the remaining 40 positive instances into the right child node, where the F4 is very high. This indicates that even when a dataset shows strong global linear separability (high minimum CC), SSLAE

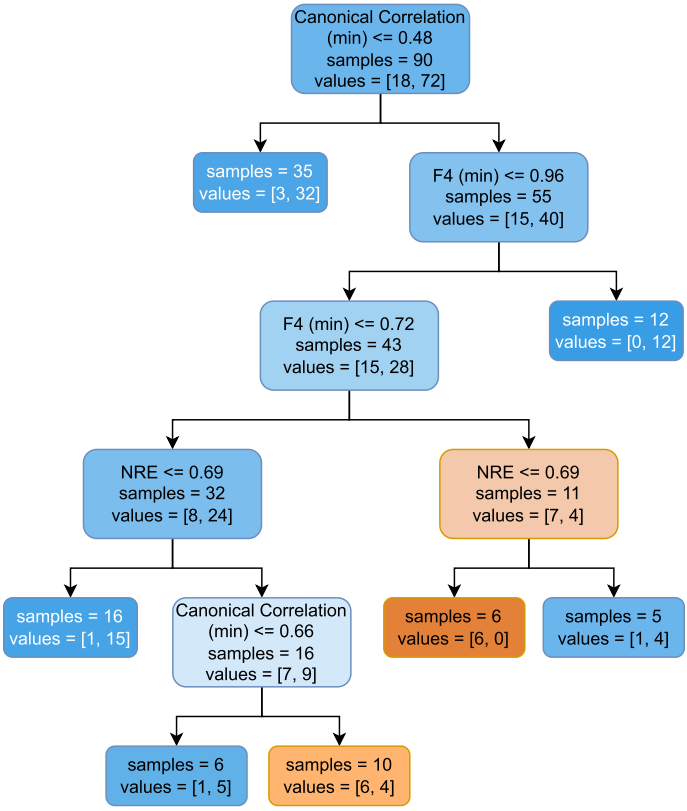

Fig. 3 Decision tree trained on the key metafeatures from the meta-analysis. Each node represents a split based on one of the metafeatures. The left child node corresponds to samples with lower values than the condition in the parent node, and the right child node to samples with higher values. Each internal node displays the metafeature used for splitting, the threshold value, the total number of samples reaching that node, and the distribution of negative and positive instances, shown in *square brackets*

can still outperform SLAE when class-specific overlaps can't be resolved using individual features (high F4). In such cases, the reconstruction loss encourages the model to learn global representations from unlabeled data, helping the classifier handle ambiguous regions better. The third split also occurs on F4, followed by splits on NRE in both child branches.

Following the right child node, we find that when datasets have good linear class separation (high CC), contain at least some features that can discriminate between the classes (moderately high F4, between 0.73 and 0.964) and uniform, balanced class distributions (low NRE), SSLAE offers no advantage over SLAE, providing a very pure leaf with 6 negative examples where SL alone is sufficient. Conversely, in the left subtree, where the F4 values are lower (fewer overlapping instances), and class balance remains uniform (low NRE), incorporating unlabeled data does not introduce noise. Instead, the minimal class overlap reduces the risk of dense regions forming around decision boundaries, and the uniformity of the class distribution provides enough context, leading to improved performance over SLAE.

The final split again occurs on minimum CC to differentiate cases where datasets have good class separation with few overlapping instances and, this time, imbalanced class distribution (high NRE). In these cases, when there are signs of complex relationships being present in the dataset (CC being between 0.483 and 0.66), using the unlabeled data provides additional context for correct classification. Conversely, when the CC exceeds 0.66, the model classifies an additional 6 negative examples, indicating that in such strongly aligned datasets, the benefit of SSL diminishes.

6 Conclusion

In this paper, we proposed SSLAE, an autoencoder-based architecture to address the task of SSL from tabular data. Unlike more complex models that rely on attention mechanisms or specialized loss functions for the unlabeled data, our method employs a straightforward architecture and a composite loss function that balances supervised classification and unsupervised reconstruction. Through extensive empirical evaluation on a diverse collection of 90 tabular benchmark datasets, we demonstrated that SSLAE consistently outperforms its supervised baseline (SLAE) and other state-of-the-art neural network-based approaches for both SSL and SL from tabular data, particularly in scenarios with very limited (labeled) data.

Our results demonstrate that a well-designed autoencoder-based model can effectively leverage unlabeled data to enhance performance. This points to the potential of simple architectures in meeting the classification challenges of SSL for tabular data, where the nature of the data differs significantly from data in image or text domains. For images and text, it is more complex neural approaches to SSL that benefit most in scenarios where both labeled and unlabeled data are available.

Our meta-analysis provides further insights into the dataset characteristics that influence the effectiveness of our approach for SSL. SSLAE tends to outperform its SL counterpart for datasets with good linear separability, limited class overlap, non-uniform class distribution, and multiple rather than just two classes. In contrast, datasets with strong linear relationships between a feature and the target, higher class overlap, and well-balanced class distributions benefit less from incorporating unlabeled data, for the proposed architecture. To the best of our knowledge, such meta-analysis is rarely performed. While our extensive empirical benchmark confirms that SSLAE performs well across a wide range of datasets, the meta-analysis goes a step further, providing insight into when and why it performs well.

Despite the promising results, our approach has some limitations. Future work will explore more complex strategies for preprocessing categorical features by using categorical feature-specific embeddings and different encoder architectures to obtain more informative representations for more complex datasets. We will also explore the use of a more advanced global loss function with an additional regularization term. Further studies on datasets with a wider range of sizes (in terms of examples and features) will be performed to evaluate the generalization capability of our approach more comprehensively.

Our work underscores that simplicity does not exclude effectiveness in SSL for tabular data. Relating dataset characteristics to the performance of our approach, we provide practitioners with guidelines on when SSL can be most beneficial. We believe that our findings will stimulate further research in developing robust and general SSL methods for tabular

Table 3 Ablation study results for different λ values, ranging from 0 to 1 with increments of 0.1

Weight	Number of labeled examples				
	50	100	200	350	500
$\lambda = 0.0$	$69.2_{\pm 4.7}$	$74.3_{\pm 3.6}$	$78.5_{\pm 2.9}$	$80.6_{\pm 2.9}$	$81.5_{\pm 2.1}$
$\lambda = 0.1$	$69.4_{\pm 4.4}$	$74.4_{\pm 3.4}$	$78.7_{\pm 2.7}$	$80.7_{\pm 2.8}$	$81.6_{\pm 2.0}$
$\lambda = 0.2$	$69.4_{\pm 4.3}$	$74.6_{\pm 3.3}$	$78.7_{\pm 2.8}$	$80.8_{\pm 2.7}$	$81.7_{\pm 2.0}$
$\lambda = 0.3$	$69.4_{\pm 4.5}$	$74.6_{\pm 3.5}$	$78.7_{\pm 2.8}$	$80.8_{\pm 2.9}$	$81.8_{\pm 2.0}$
$\lambda = 0.4$	$69.3_{\pm 4.5}$	$74.4_{\pm 3.5}$	$78.9_{\pm 2.7}$	$80.8_{\pm 2.8}$	$81.9_{\pm 2.0}$
$\lambda = 0.5$	$69.3_{\pm 4.6}$	$74.5_{\pm 3.6}$	$78.7_{\pm 2.7}$	$81.0_{\pm 2.6}$	$81.9_{\pm 2.0}$
$\lambda = 0.6$	$69.3_{\pm 4.6}$	$74.4_{\pm 3.7}$	$78.8_{\pm 2.8}$	$80.9_{\pm 2.6}$	$81.9_{\pm 1.9}$
$\lambda = 0.7$	$69.1_{\pm 4.8}$	$74.2_{\pm 3.8}$	$78.8_{\pm 2.8}$	$80.8_{\pm 2.5}$	$81.7_{\pm 1.9}$
$\lambda = 0.8$	$69.1_{\pm 4.8}$	$74.1_{\pm 3.9}$	$78.6_{\pm 2.9}$	$80.5_{\pm 2.7}$	$81.4_{\pm 1.9}$
$\lambda = 0.9$	$68.8_{\pm 4.6}$	$73.5_{\pm 4.0}$	$77.9_{\pm 3.2}$	$79.7_{\pm 2.9}$	$80.7_{\pm 2.0}$
$\lambda = 1.0$	$37.3_{\pm 8.1}$	$37.5_{\pm 8.2}$	$37.2_{\pm 7.6}$	$37.1_{\pm 7.9}$	$36.4_{\pm 7.6}$

The results in the table show the average value of the AUPRC Macro score across all datasets for SSLAE, with the exact λ value denoted in the rows. The values in subscript show the standard deviation

data, and encourage the exploration of simple yet practical models that can integrate unlabeled data, paving the way for more accessible and efficient SSL approaches.

Appendix A: Weight parameter (λ) influence study

We conducted an ablation study for the supervised part of the loss, reducing its role and eventually removing it completely ($\lambda = 1$). We vary the weight parameter λ in Eq. 1 from 0.0 to 1.0 in increments of 0.1, to investigate how the balance between supervised and unsupervised losses affects performance (Table 3). The results show that SSLAE's performance, measured by AUPRC Macro, is stable for λ between 0.1 and 0.6, and it only begins to decline once λ exceeds 0.7. In particular, values in the middle range (0.3–0.6) yield highly similar average scores and standard deviations, indicating that a moderate trade-off between classification and reconstruction losses is sufficient to leverage unlabeled data without sacrificing supervised performance. Lower values (e.g., $\lambda = 0.0$, which means fully supervised learning) yield slightly lower scores, likely due to the limited number of labeled examples in limited-label settings. On the other end, performance gradually declines beyond $\lambda = 0.7$, and collapses completely at $\lambda = 1.0$, where the model is trained solely on reconstruction loss. In this purely unsupervised setting where $\lambda = 1.0$, the network is only minimizing reconstruction error. As a result, the encoder and decoder learn to compress and reconstruct the input, but the classifier sees virtually no label-driven updates. The classification head never receives the training signal it needs, so the model ends up with poor classification accuracy.

Selecting the midpoint $\lambda = 0.5$ provides strong and consistent performance across different amounts of labeled data, without the need for additional, per-dataset, tuning of λ.

Appendix B: Datasets

The datasets used are described in Table 4. The table lists the values of three properties of the datasets: number of examples, number of features, and number of classes, per dataset. A histogram summarizing the distribution of the values of their three properties is given in Fig. 4.

Appendix C: Hyperparameter optimization study

In this section, we present a limited hyperparameter optimization (HPO) study to assess the impact of tuning SSLAE and its NN-based baselines. The experiment was conducted on six representative datasets of different sizes (*abalone*, *bank32nh*, *car*, *elevators*, *krkopt*, and *pgp*), spanning 3 labeled data regimes (50, 200, and 500), and the results were averaged over 10 seeds.

C.1 Optimization details

We used the Optuna library (Akiba et al., 2019) to perform Bayesian optimization. For SLAE and SSLAE, we optimized the dropout rate in the classifier, learning rate, and weight decay. For SSLAE we additionally optimized the λ parameter. For each of the competing algorithms, we optimized the hyperparameters as specified in the respective papers, along with the learning rate and weight decay. Table 5 shows the explored HPO space for each algorithm. We set the budget for each optimization run to a maximum of 30 trials or 15 min, whichever occurred first. For each algorithm, HPO was done using solely the training dataset, with threefold cross-validation used to split it into HPO training and HPO validation sets. Early stopping was enabled via Optuna's default pruning mechanism to discard unpromising trials.

C.2 Results

Table 6 summarizes the results in terms of macro-averaged AUPRC (top half) and average ranks (bottom half) for all methods under default (DEF) and hyperparameter-optimized (HPO) settings. Results are averaged across six datasets and 10 seeds per labeled data regime (50, 200, and 500 examples).

While tuning causes slight variations in performance, it does not change the top-performing model in any labeled regime: SSLAE remains best at 50 and 200 labels, and SLAE at 500. Ranks are largely stable across HPO and DEF settings, especially for SSLAE, SLAE, and TabNet. Some models exhibit minor shifts. TabTransformer shows consistently lower performance when optimized. The self-supervised version of TabNet shows an increase of 2 ranks in the 500 labels regime, moving from rank 5 in DEF to rank 3 in HPO. FT-Transformer and the semi-supervised version of VIME show improvement by 1 rank in the 50 labeled examples regime, and drop by 1 rank at 200 and 500 labels. The self-supervised version of VIME has the opposite trend of VIME-SSL and FT-Transformer: losing a rank in the case of 50 labeled examples, and gaining one in the 200 and 500 labeled examples regimes.

Table 4 The datasets and their properties: number of samples, number of features, and number of classes

Dataset name	N examples	N features	N classes
car	1728	6	4
cardiotocogramy10	2126	35	10
dna	3186	180	3
imagesegment	2310	18	7
wall-robot-navigation	5456	4	4
abalone	4177	8	2
banknote-authentication	1372	4	2
kr-vs-kp	3196	36	2
mofn-3-7-10	1324	10	2
mfeat-fourier	2000	76	2
Satellite	5100	36	2
socmob	1156	5	2
space_ga	3107	6	2
visualizing_soil	8641	4	2
allrep	3772	29	4
artificial-characters	10,218	7	10
baseball	1340	16	3
collins	1000	23	30
segmentation	2310	19	7
MiceProtein	1080	77	8
obesity	2111	16	7
texture	5500	40	11
winequality	6497	11	7
1StudentPerfromance	1000	7	2
ada_prior	4562	14	2
delta_ailerons	7129	5	2
ringnorm	7400	20	2
satimage	6430	36	6
splice	3190	60	2
spambase	4601	57	2
steel-plates-fault	1941	33	2
students_scores	1000	7	2
sylvine	5124	20	2
allbp	3772	29	3
cardiotocogramy3	2126	35	3
digits	1797	64	10
optdigits	5620	64	10
waveform_21	5000	21	3
churn	5000	20	2
diatoma_vulgare-t14200	1060	9	2
hypo	3163	25	2
JapaneseVowels	9961	14	2
quake	2178	3	2
rmftsa_sleepdata	1024	2	2
puma32H	8192	32	2
cpu_act	8192	21	2
cpu_small	8192	12	2
fri_c2_1000_10	1000	10	2

Table 4 (continued)

Dataset name	N examples	N features	N classes
ada	4147	48	2
national-longitudinal-survey-binary	4908	16	2
ibm-employee-performance	1470	33	2
letter	20,000	16	26
MagicTelescope	19,020	10	2
pendigits	10,992	16	10
segment	2310	16	7
elevators	16,599	18	2
bank32nh	8192	32	2
house_16H	22,784	16	2
ailerons	13,750	40	2
colleges_aaup	1161	15	2
led24	3200	24	10
USPS	1424	256	2
xd6	973	9	2
tokyo1	959	44	2
vowel	990	13	11
contraceptive	1473	9	3
yeast	1479	8	9
mfeat_factors	2000	216	10
mfeat_morphological	2000	6	10
mfeat_karhunen	2000	64	10
mfeat_pixel	2000	240	10
german	1000	20	2
kin8nm	8192	8	2
mushroom	8124	22	2
pgp	932	183	2
PhishingWebsites	11,055	30	2
rice	3810	7	2
UCI_churn	3333	20	2
wind	6574	14	2
gasdrift	13,910	128	6
qsar-biodeg	1055	41	2
led7	3200	7	10
waveform_40	5000	40	3
nursery	12,960	8	5
wine_quality_white	4898	11	7
car_evaluation	1728	21	4
krkopt	28,056	6	18
coil2000	9822	85	2
compas-two-years	4966	11	2
tic-tac-toe	958	9	2

The Bayesian t-test results in Table 7 indicate an increase in the probability of some tuned baselines outperforming SSLAE. For instance, with 500 labeled examples, there is a 0.79 probability that SSLAE and SLAE will perform within 1% AUPRC of each other, and a probability of 0.21 (which was 0.14 in the default parameters case) that SLAE will

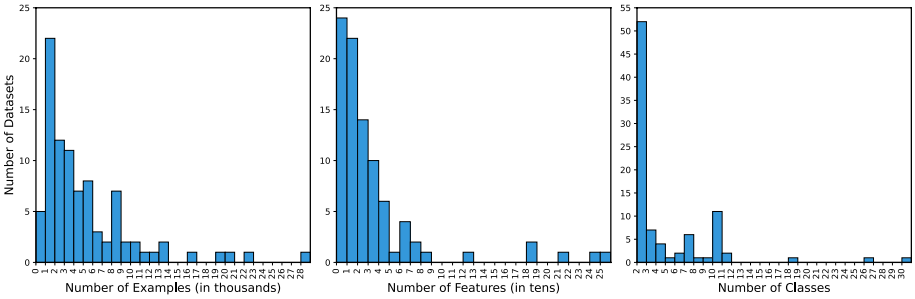

Fig. 4 Histograms showing the distributions of the number of examples (*left*), number of features (*middle*), and number of classes (*right*). Note that the number of examples is shown in thousands, and the number of features is shown in tens

perform better. Under the same conditions, the probability that TabTransformer outperforms SSLAE by more than 1% AUPRC rises from 21% (default) to 32% (HPO).

Hyperparameter tuning also introduces considerable computational overhead. For example, running the default version of SSLAE on 50 labeled examples for the *abalone* dataset takes on average 14 s. The hyperparameter optimization time for the same dataset, for the same number of labeled examples, would put an additional overhead of (on average) 873 s, which is a 60-fold increase.

These results indicate that while hyperparameter tuning can introduce minor performance shifts, it does not substantially alter the relative ordering of algorithms, nor does it affect the conclusions of our main study. The best-performing methods remain the same across label regimes, and improvements from tuning are often marginal. The high computational cost and minimal change in performance lead to the conclusion that using the default hyperparameters provides a fair and pragmatic basis for large-scale benchmarking.

Appendix D: Implementation details

D.1 Architecture

In order to maintain consistency, we applied the following global hyperparameters for all datasets: (1) batch size set to 32, (2) learning rate set to 0.001, using Adam as the optimizer, and (3) the maximum number of training epochs set to 100, with early stopping implemented when no improvement on the validation set is observed for 3 consecutive epochs. The estimation of the intrinsic dimension for each dataset is an exception to this rule of no hyperparameter optimization, due to its importance in determining the appropriate size of the latent space in the autoencoder architecture.

The total number of trainable parameters in SSLAE scales linearly with the number of features and classes of the dataset, and latent dimensions. For a dataset with 20 features, 5 latent dimensions, and 4 classes, SSLAE has 743 parameters:

Table 5 Hyperparameter search spaces used for each algorithm

Algorithm	Hyperparameter	Search space/values
SLAE/SSLAE	λ	$\{0.0, 0.1, 0.2, \ldots, 1.0\}$*(SSLAE only)*
	Learning rate	Log-uniform $[10^{-6}, 10^{-3}]$
	Weight decay	Log-uniform $[10^{-6}, 10^{-3}]$
	Dropout rate (Classifier)	$\{0.0, 0.2, 0.4, 0.6, 0.8, 0.9\}$
VIME	p_m	$\{0.1, 0.2, \ldots, 0.9\}$
	α	$\{0.1, 0.2, \ldots, 10.0\}$*(VIME-SelfSL only)*
	β	$\{0.1, 0.2, \ldots, 10.0\}$*(VIME-SSL only)*
	K	$\{2, 3, 5, 10, 15, 20\}$*(VIME-SSL only)*
TabNet	n_d, n_a	$\{8, 16, 32, 64, 128\}$
	n_{steps}	Integer: [3, 10]
	γ	$\{1.0, 1.5, 2.0\}$
	λ_{sparse}	$\{0, 10^{-6}, 10^{-4}, 10^{-3}, 10^{-2}, 0.1\}$
	Momentum	$\{0.6, 0.7, 0.8, 0.9, 0.95, 0.98\}$
	Learning rate	$\{0.005, 0.01, 0.02, 0.025\}$
	Weight decay	Log-uniform: $[10^{-6}, 10^{-3}]$
	n_{shared}	Integer: [1, 3]
	$n_{\text{independent}}$	Integer: [1, 3]
	Pretraining ratio	$\{0.2, 0.4, 0.5, 0.7, 0.8\}$*(TabNet-SelfSL only)*
TabTransformer	Hidden dimension	$\{32, 64, 128, 256\}$
	Number of layers	$\{1, 2, 3, 6, 12\}$
	Attention heads	$\{2, 4, 8\}$
	Attention dropout	Uniform: [0.0, 0.5]
	FF dropout	Uniform: [0.0, 0.5]
	Learning rate	Log-uniform: $[10^{-5}, 10^{-3}]$
	Weight decay	Log-uniform: $[10^{-6}, 10^{-3}]$
FT-Transformer	Feature Embedding Size	Integer: [32, 512]
	Number of layers	Integer [1, 6]
	Attention heads	$\{2, 4, 8\}$
	Attention dropout	Uniform: [0.0, 0.5]
	FF dropout	Uniform: [0.0, 0.5]
	Learning rate	Log-uniform: $[10^{-5}, 10^{-3}]$
	Weight decay	Log-uniform: $[10^{-6}, 10^{-3}]$

1. Encoder:

   (a) Linear(20 → 12): weights = 20 12, bias = 12 → 240 + 12 = 252
   (b) Linear(12 → 5): weights = 12 5, bias = 5 → 60 + 5 = 65
   (c) Encoder total: 252 + 65 = 317

2. Decoder:

   (a) Linear(5 → 12): 5 12 + 12 → 60 + 12 = 72
   (b) Linear(12 → 20): 12 20 + 20 → 240 + 20 = 260
   (c) Decoder total: 72 + 260 = 332

Table 6 Comparison of macro-averaged AUPRC scores and average ranks for different algorithms under hyperparameter-optimized (HPO) and default (DEF) settings, for 50, 200, and 500 labeled data points on 6 datasets

Algorithm	AUPRC (50)		AUPRC (200)		AUPRC (500)	
	HPO	DEF	HPO	DEF	HPO	DEF
SLAE	$62.7_{6.6}$	$66.3_{4.1}$	$76.4_{3.3}$	$77.6_{3.2}$	$\mathbf{83.7}_{2.5}$	$\mathbf{84.9}_{2.7}$
TabNet	$50.8_{5.8}$	$47.4_{5.3}$	$65.8_{7.9}$	$61.7_{7.9}$	$75.5_{4.1}$	$72.6_{5.5}$
TabTransformer	$62.0_{4.1}$	$63.1_{3.2}$	$69.7_{2.8}$	$70.3_{3.3}$	$74.8_{2.8}$	$74.9_{2.6}$
FT-Transformer	$58.3_{4.8}$	$57.7_{4.4}$	$69.8_{4.9}$	$70.3_{3.9}$	$78.1_{3.7}$	$79.0_{2.4}$
TabNet-SelfSL	$48.9_{6.1}$	$48.6_{6.3}$	$64.8_{8.9}$	$62.9_{7.0}$	$80.7_{4.8}$	$77.7_{7.8}$
VIME-SelfSL	$57.7_{9.6}$	$58.9_{6.2}$	$69.1_{5.7}$	$68.5_{5.0}$	$76.0_{4.0}$	$74.0_{4.0}$
VIME-SSL	$58.5_{5.8}$	$57.3_{8.2}$	$67.7_{4.8}$	$68.5_{4.3}$	$74.9_{3.3}$	$73.9_{3.5}$
SSLAE (ours)	$\mathbf{65.1}_{5.4}$	$\mathbf{68.1}_{4.4}$	$\mathbf{77.5}_{3.1}$	$\mathbf{78.7}_{2.9}$	$82.1_{3.1}$	$84.5_{3.0}$
	Avg Rank (50)		Avg Rank (200)		Avg Rank (500)	
	HPO	DEF	HPO	DEF	HPO	DEF
SLAE	2.40	2.40	2.80	2.80	**2.40**	**1.80**
TabNet	5.60	6.20	5.40	5.60	5.00	5.40
TabTransformer	5.00	4.60	5.00	5.00	4.80	4.60
FT-Transformer	4.20	4.60	4.80	4.60	4.40	4.40
TabNet-SelfSL	7.00	6.40	6.60	6.40	4.00	5.00
VIME-SelfSL	4.80	4.40	4.60	5.00	5.80	6.40
VIME-SSL	5.00	5.60	5.00	5.00	6.00	6.00
SSLAE (ours)	**2.00**	**1.80**	**1.80**	**1.60**	3.60	2.40

Best performances are in bold (higher is better for AUPRC, lower is better for rank)

Table 7 Bayesian t-test results comparing the performance (in terms of AUPRC Macro) of SSLAE against competing algorithms across different numbers of labeled examples after HPO

Algorithm	Number of labeled examples					
	50		200		500	
	p(algo)	p(rope)	p(algo)	p(rope)	p(algo)	p(rope)
SLAE	0.0095	0.6885	0.00	0.7866	0.2070	0.7930
VIME-SelfSL	0.00	0.0007	0.00	0.0161	0.00	0.1143
VIME-SSL	0.00	0.0159	0.00	0.0596	0.00	0.0573
TabNet	0.00	0.0161	0.00	0.0031	0.1409	0.1200
TabNet-SelfSL	0.00	0.0007	0.00	0.0007	0.2708	0.1091
TabTransformer	0.3703	0.0035	0.2763	0.0038	0.3266	0.0457
FT-Transformer	0.1357	0.0017	0.0992	0.0026	0.1653	0.0297

The results are rounded to two decimals when the result is exactly 0 or 1; four decimal places are provided otherwise

3. Classifier:

   (a) Linear($5 \rightarrow 9$): $5\ 9 + 9 \rightarrow 45 + 9 = 54$
   (b) Linear($9 \rightarrow 4$): $9\ 4 + 4 \rightarrow 36 + 4 = 40$
   (c) Classifier total: $54 + 40 = 94$

So, the final number of trainable parameters is $317 + 332 + 94 = 743$.

D.2 Software details

We implemented our code using Python 3.10. The proposed model is implemented in PyTorch Lightning. For the competing algorithms, we used and adapted implementations available on GitHub for VIME,[1] TabNet,[2] and TabTransformer/FT-Transformer.[3]

D.3 Hardware details

We evaluated all of our experiments on a commodity PC.

Listing 1 System Description

```
################ CPU ###############
CPU op-mode(s):          32-bit, 64-bit
Address sizes:           64 bits physical, 64 bits virtual
Byte Order:              Little Endian
CPU(s):                  16
Vendor ID:               AuthenticAMD
Model name:              AMD Ryzen 7 5700G with Radeon Graphics
CPU family:              107
Model:                   178BFBFF00A50F00
Thread(s) per core:      2
Core(s) per socket:      8
Socket(s):               1
Stepping:                0
CPU max MHz:             3801
CPU current MHz:         3801
Caches (sum of all):
  L3:                    512 KiB
  L4:                    4 KiB
  L5:                    16 KiB
################ Memory ###############
Total Memory             63 GiB
Used Memory              13 GiB
Free Memory              50 GiB
Swap total               4 GiB
```

D.4 Metafeature extraction and selection

Metafeature extraction. When extracting metafeatures using the pyMFE library, we handled categorical and numerical features separately. Specifically, we excluded numerical features when computing categorical-specific metafeatures and vice versa. For the nine datasets that only contain categorical features, we applied one-hot encoding, omitting the intercept term to ensure that no category was dropped.

[1] https://github.com/jsyoon0823/VIME/.

[2] https://github.com/dreamquark-ai/tabnet.

[3] https://github.com/lucidrains/tab-transformer-pytorch.

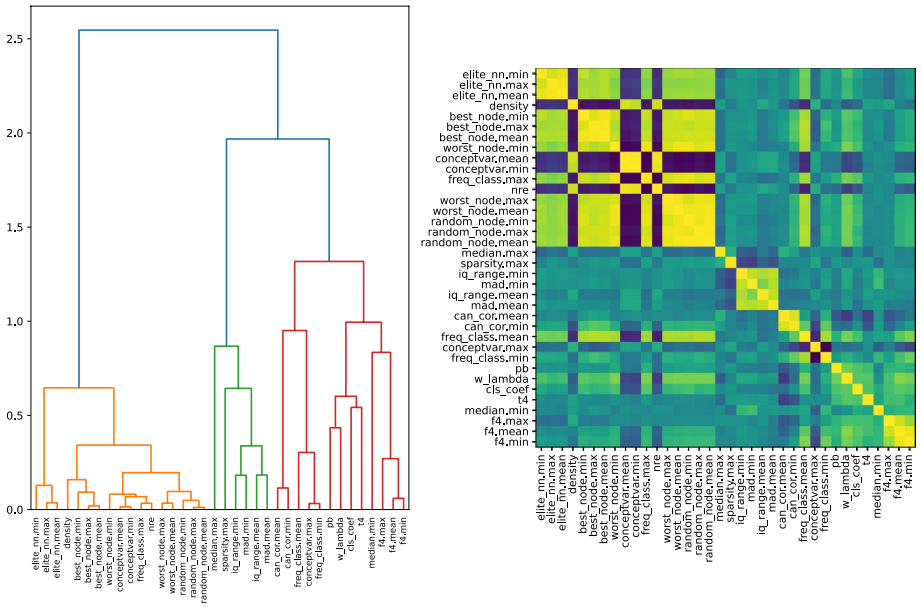

Fig. 5 Dendrogram and correlation heatmap showing the feature relationships

Hierarchical clustering for metafeature selection. Before computing the Permutation Importance (PI) scores, we applied hierarchical clustering to group correlated metafeatures and select the most representative ones. First, we computed pairwise correlations between metafeatures and performed hierarchical clustering using Ward's linkage method, producing the dendrogram and heatmap shown in Fig. 5. To determine an appropriate clustering threshold, we evaluated different distance cutoffs and selected the one that maximized the silhouette score while ensuring that each cluster contained at least two metafeatures. The optimal threshold was found to be 0.8677, resulting in 6 distinct clusters.

From each of these clusters, we selected a single representative metafeature for subsequent PI score estimation. This was done by using an unsupervised selection strategy, choosing the metafeature with the highest variance in each cluster. This approach ensures that we retain the most informative feature from each group while avoiding redundancy.

After selecting the six metafeatures, we trained a Random Forest classifier,[4] with balanced subsampling and a minimum of five instances per leaf. The PI score[5] is calculated based on the balanced accuracy metric. For robustness, we estimate the PI score across 10 different random seeds.

Finally, to analyze the interaction between metafeatures, we trained a pruned Decision Tree[6] using the selected metafeatures, imposing a minimum of five samples per leaf to prevent overfitting while maintaining interpretability.

[4] https://scikit-learn.org/1.3/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[5] https://scikit-learn.org/1.3/modules/permutation_importance.html.

[6] https://scikit-learn.org/1.3/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

D.5 Training time

The time needed to train SSLAE and its NN-based baselines is presented in Table 8. The time for each labeled data regime and method is averaged across datasets and seeds, where lower is better for the Average Execution Time and higher is better for the Average Rank. The table shows that, even though SSLAE on average takes more time than the supervised algorithms (with the exception of FT-Transformer), it still takes less time than the other algorithms that use unlabeled data (VIME-SSL and TabNet-SSL), while the self-supervised version of VIME takes the least time to train.

Appendix E: Meta-analysis of SSLAE vs. NN baselines

In this section, we extend the meta-analysis from Sect. 4.3 to analyze the relationship between the dataset metafeatures and the *Performance Area* between SSLAE and the other NN-based baselines from this paper: VIME-SelfSL, VIME-SSL, TabNet, TabNet-SelfSL, TabTransformer, and FT-Transformer.

We follow the same methodology described in Sects. 4.3 and 5.2, with the difference being that we now formulate a multi-label classification (MLC) task, where each of the six binary targets corresponds to whether SSLAE outperforms a given baseline method on a particular dataset.

We used the `CLUS` software package (Petković et al., 2023), which supports multi-target prediction with both single and ensembles of predictive clustering trees (PCTs). Feature importance is computed using a Random Forest-based score, equivalent to the PI score used in the single-target analysis in Sect. 5.2.1. The six most important metafeatures in this extended setting were, in order of importance: NRE (Normalized Relative Entropy, class imbalance), WL (Wilks' Lambda, overall linear feature-target alignment), CC min (Canonical Correlation, minimum linear feature-target alignment), F4 min (Collective Feature Efficiency, overlap between a pair of classes), IQR min (Interquartile Range, feature dispersion), and RFC mean (Relative Frequency of Each Distinct Class, binary/multiclass problem).

To visualize how these metafeatures interact to affect SSLAE's performance against its baselines, we trained a multi-target PCT for MLC, shown in Fig. 6. Each internal node

Table 8 Average execution time and rank for SSLAE and the NN baselines across labeled sample sizes

Method	Average execution time (s)					Average rank				
	50	100	200	350	500	50	100	200	350	500
SSLAE	14.03	15.18	19.61	25.81	32.53	3.28	3.60	3.80	3.82	4.08
SLAE	8.73	8.28	8.97	10.31	11.59	4.46	5.06	5.61	5.76	5.80
VIME-Self	1.27	1.25	1.22	1.18	1.12	7.58	7.68	7.78	7.82	7.88
VIME-SSL	47.18	51.41	57.45	72.47	84.96	1.61	1.58	1.64	1.71	1.68
TabNet	3.43	5.68	8.99	12.57	16.06	6.24	5.69	5.16	5.04	5.00
TabNet-SelfSL	22.33	24.90	28.80	34.29	38.54	3.03	2.74	2.66	2.71	2.71
TabTransformer	4.94	7.46	8.39	11.42	14.40	6.31	6.37	6.37	6.39	6.27
FTTransformer	14.72	20.62	33.01	49.09	66.26	3.49	3.29	2.99	2.74	2.59

Lower is better for the average execution time, and higher is better for the average rank

corresponds to a metafeature-based split, while leaf nodes display the number of datasets falling into that region, along with the predicted binary vector indicating whether SSLAE outperforms each baseline method. The vector ordering is: [VIME-SelfSL, VIME-SSL, TabNet, TabNet-SelfSL, TabTransformer, FT-Transformer]. For reference, SSLAE is outperformed by these methods on the following number of datasets, respectively: [16, 30, 7, 9, 34, 25]. The leaf nodes also feature the support for the predictions, i.e., how many samples in the leaf actually have the label that was predicted.

E.1 SSLAE vs. VIME

The self-supervised version of VIME relies solely on masked feature recovery, while the semi-supervised variant adds a consistency regularization loss on label predictions across augmented views. From the PCT we identify several dataset regimes where VIME has an advantage over SSLAE:

Very high F4. VIME in both variants outperforms SSLAE in datasets where F4 is extremely high, indicating the presence of at least one pair of classes that are not separable by any feature subset, pointing to extensive class overlap. This is a case where SSLAE's

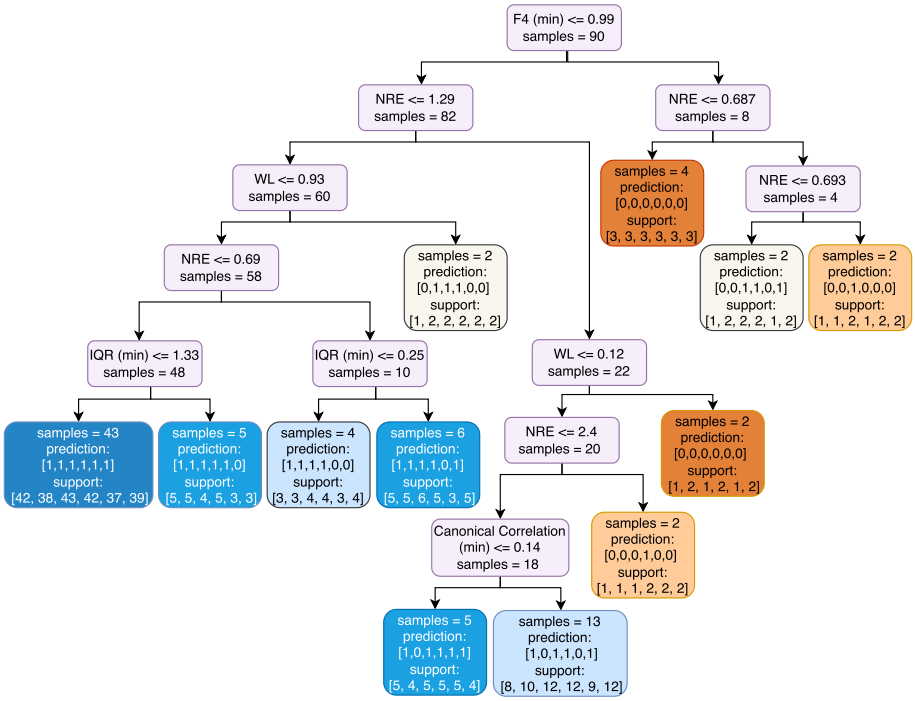

Fig. 6 Decision tree trained on the key metafeatures from the meta-analysis. Each node represents a split based on one of the metafeatures. *Blue* leaf nodes indicate that SSLAE performs better (more positive examples), while *orange* leaf nodes indicate that SSLAE is outperformed (more negative examples). Predictions in the leaf nodes are for the NN baselines in the order: [VIME-SelfSL, VIME-SSL, TabNet, TabNet-SelfSL, TabTransformer, FT-Transformer]. The intensity of the color reflects the proportion of positive or negative examples in the leaf node. The prediction for each leaf is given along with the support (how many of the examples in the leaf node have the predicted value)

reliance on the global reconstruction loss may be misled by the overlapping class structure, while VIME's feature masking is more successful at learning meaningful conditional dependencies even when the global separability is poor.

Moderate-to-low F4, high NRE. In datasets where there is at least one feature that can distinguish between the classes (moderate-to-low F4), but imbalanced classes (high NRE) and high class overlap (high WL), both versions of VIME outperform SSLAE. VIME also has an edge in highly imbalanced classes (very high NRE), which are compact and well clustered (very low WL). VIME, especially in its semi-supervised form, benefits from label-aware regularization that stabilizes learning even under imbalance and variance. On the other hand, SSLAE may flatten or misrepresent minority-class structure in the latent space when there are dominant classes.

Moderate-to-low F4, low NRE, high WL (self-supervised version only). The self-supervised variant of VIME outperforms SSLAE when there is no extreme class imbalance, but there is some degree of overlap in the class space (high WL). In the absence of labels, VIME's corruption task manages to learn more stable, structure-preserving embeddings, which produce a latent space that is separable across classes.

Moderate-to-low F4, moderate NRE, low WL (semi-supervised version only). The semi-supervised version of VIME outperforms SSLAE when the imbalance is moderate ($1.29 < \text{NRE} \leq 2.4$), and the classes are compact (low WL), uniquely being the only one to do so in the case when CC is below 0.14. Here, its consistency regularization, combined with label supervision, helps extract sharper decision boundaries that are more adapted to the class geometry.

E.2 SSLAE vs. TabNet

TabNet uses instance-wise, sparse feature selection through sequential attention. Each decision step focuses on a subset of informative features, allowing efficient learning and interpretability. The supervised version uses this structure to refine discriminative splits, while the self-supervised variant pretrains the encoder by predicting masked features from partial input as the first step. There are several regions where TabNet can outperform SSLAE:

Very high F4 and low NRE. TabNet in both variants excels when there is significant overlap between at least two pairs of classes (high F4), but the classes are balanced (NRE). Its localized, instance-wise selection helps isolate discriminative signals and avoids wasting capacity on irrelevant features.

Moderate-to-low F4, high NRE, high WL. On datasets where features can distinguish well between the classes (moderate-to-low F4), but there is high imbalance (high NRE) and low overall linear separability of the classes (high WL), both TabNet variants outperform SSLAE. TabNet's sparse attention mechanism for feature selection gives it an edge over SSLAE when the features explain only a small fraction of the variance in the target (high WL).

Moderate-to-low F4, very high NRE, low WL (supervised version only). The supervised TabNet variant is advantageous in datasets with clean class clusters (low WL) and effective feature spaces (moderate-to-low F4), but with extreme class imbalance (very high NRE). It benefits from its supervised attention-based feature selection, which focuses directly on the high-signal features even in the case of extreme imbalance.

Very high F4, high NRE (self-supervised version only). The self-supervised variant of TabNet shows an advantage in datasets with significant class overlap (very high F4) and imbalance (high NRE). While the supervised version of TabNet can also outperform SSLAE in cases with very high F4, its condition is that NRE is low. Pretraining with masked feature reconstruction allows TabNet to learn additional structure from the data which SSLAE can't exploit, succeeding even in imbalanced settings.

E.3 SSLAE vs. TabTransformer

TabTransformer introduces feature-wise attention and contextual tokenization that present a strong inductive bias for capturing feature interactions. This model outperforms SSLAE on the highest number of datasets among all neural baselines in this meta-analysis (34 out of 90), and several distinct dataset regimes highlight its architectural strengths:

Very high F4. TabTransformer outperforms SSLAE on datasets with very high F4, where all features fail to uniquely distinguish between at least one pair of classes. In these cases, TabTransformer uses feature-wise attention to modulate how each input dimension is interpreted in the context of others, suppressing redundant dimensions, enabling it to capture the discriminative structure even when the feature space is not effective.

Moderate-to-low F4, moderate NRE, low WL. TabTransformer outperforms SSLAE in datasets with clearer class separability (moderate-to-low F4), moderate imbalance ($1.29 < \text{NRE} \leq 2.4$), and strong linear separability between classes (low WL, high CC). Another region of exceptional success, where only TabTransformer manages to outperform SSLAE, features separable classes (moderate-to-low F4, low WL), moderate imbalance ($0.69 < \text{NRE} \leq 1.29$), and high feature dispersion (high IQR). This suggests that when there is non-trivial alignment between the features and the targets, and there is some signal from the class imbalance and/or from the feature dispersion, TabTransformer's contextualized embeddings are more effective than SSLAE's shared latent space.

Moderate-to-low F4, high NRE. In datasets that are both imbalanced (high NRE) and have overall weak class separability (high WL), TabTransformer outperforms SSLAE, likely due to its attention-based feature weighting that helps emphasize relevant signals from minority classes. In the case when the imbalance is very high ($\text{NRE} > 2.4$), there has to be some alignment between the classes and the target (low WL) in order for TabTransformer to outperform SSLAE.

Moderate-to-low F4, low NRE, high WL. Datasets where class distributions are balanced (low NRE) but the classes are not easily separable (high WL) present another region of success for TabTransformer, which remains sensitive to local variability and is more successful at preserving separable structures even when the features fail to explain the variance in the target space.

E.4 SSLAE vs. FT-Transformer

FT-Transformer's distinction is that it treats all features uniformly as tokens. It avoids the specialized modules or pretraining tasks that TabTransformer has, relying on residual connections, learnable feature embeddings, and Transformer blocks with layer normalization and dropout. FT-Transformer outperforms SSLAE on 25 out of 90 datasets, particularly in regimes with:

Very high F4, NRE not between 0.687 and 0.693. FT-Transformer outperforms SSLAE on datasets exhibiting class overlap across features (very high F4), except when NRE falls in a narrow window between 0.687 and 0.693. This suggests that FT-Transformer is robust to class overlap as long as imbalance does not hit a critical, edge-case range.

Moderate-to-low F4, high NRE, high WL. In datasets with separable classes (moderate-to-low F4), but strong imbalance (high NRE) and weak overall feature-target relationships (high WL), FT-Transformer outperforms SSLAE.

Moderate-to-low F4, very high NRE, low WL. Regions where the dominant classes are compact (low WL) but very overrepresented (very high NRE) are beneficial for FT-Transformer. Its regularized attention layers help FT-Transformer avoid overfitting to patterns from the majority classes.

Moderate-to-low F4, low NRE, high WL. FT-Transformer is dominant also in balanced datasets (low NRE) with weak overall feature-target alignment (high WL). This model is able to model local dependencies between the features, even when the class signal is noisy.

Moderate-to-low F4, moderate-to-low NRE, low WL, low IQR. FT-Transformer outperforms SSLAE in datasets with lower class overlap (moderate-to-low F4), moderate-to-low class imbalance ($0.69 < \text{NRE} \leq 1.29$), good overall alignment between the features and the target (low WL), and low feature dispersion (low IQR). These are datasets where unlabeled data provides limited additional signal.

Moderate-to-low F4, very low NRE, low WL, high IQR. A unique regime where only FT-Transformer outperforms SSLAE is on balanced (very low NRE) datasets with efficient feature space (low F4) that is well aligned with the target space (low WL), where the feature dispersion is higher (high IQR). In these cases, FT-Transformer's embeddings are more beneficial than SSLAE's autoencoder structure, as it embeds features independently and applies attention across them, allowing for a more selective focus.

Across these baselines, the meta-analysis reveals that SSLAE is less beneficial when the classes are not balanced (high NRE) and overlap (high F4), or are not well aligned with the feature space (high WL). VIME's localized corruption task and consistency loss allow the model to learn structure from overlapping or imbalanced data without relying on global reconstruction. TabNet benefits from its instance-wise attention that isolates predictive features even when the global signal is noisy or biased. TabTransformer uses the contextualized embeddings to recover feature dependencies missed by SSLAE, especially in feature spaces that are moderately aligned with the target or are dispersed. FT-Transformer's uniform feature handling makes it robust to both highly variable and low complexity regimes.

In contrast, SSLAE demonstrates advantage in scenarios where there is low F4, low NRE, low WL, and low IQR, indicating that SSLAE is superior on datasets with at least one feature that discriminates well between the classes (low F4), that have uniformly balanced classes (low NRE), and a compact feature space (low IQR) that is overall well-aligned with the target (low WL). In such datasets, where the SSL assumptions hold, its combined reconstruction and classification loss enables SSLAE to learn useful information from the unlabeled data.

Appendix F: Comparison to tree-based methods

The SSLAE model proposed in this paper is an NN-based approach for SSL from tabular data. Accordingly, the core experimental comparisons throughout this paper have focused on other neural models, both supervised and semi-supervised. In this section, we broaden the evaluation by comparing SSLAE against four widely used tree-based methods: the Random Forest (RF) method, XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018), each applied in both supervised and semi-supervised settings, with default hyperparameters, on datasets preprocessed as described in Sect. 4.1. For the semi-supervised variants, we wrapped each tree-based model in a self-training framework[7]: the model is first trained on the labeled portion of the data, then used to generate pseudo-labels for the unlabeled examples. Confident predictions are iteratively added to the training set until a stopping condition is met.

F.1 Performance comparison

Tables 9 and 10 summarize the performance comparison between SSLAE and tree-based baselines. Table 9 reports the average AUPRC Macro scores across all datasets for varying amounts of labeled data, along with the averaged ranks of each method. Table 10 presents the Bayesian t-test comparisons between SSLAE and each tree-based method, including the probability that a method outperforms SSLAE (*p(algo)*), and the probability of practical equivalence (*p(rope)*), defined as difference in AUPRC Macro below 1%.

The results indicate that although SSLAE outperforms all contemporary NN-based methods (as shown in the main text), the supervised tree-based models achieve better overall performance. CatBoost, in particular, delivers the best results across all labeled data regimes. In contrast, the SSL versions of these models do not outperform their supervised counterparts and are generally ranked below SSLAE, particularly in low-label settings. The Bayesian comparisons reinforce this trend. While SSLAE's relative performance improves slightly as the number of labeled examples increases, this improvement is not enough to close the performance gap. In particular, CatBoost remains the most difficult model to outperform.

These findings suggest that while SSLAE offers a competitive way to include unlabeled data within NN-based approaches, the effectiveness of SSL in tabular data remains very dependent on the dataset characteristics, and tree-based methods continue to be strong contenders, even if considering labeled data only.

F.2 Meta-analysis

To gain a deeper understanding of when SSLAE outperforms the tree-based methods, and vice versa, we performed a meta-analysis following the methodology described in Sect. 4.3. For each tree-based baseline (RF, XGBoost, LightGBM, and CatBoost, along with their SSL variants), we computed a binary target per dataset: a value of 1 if SSLAE outperforms the baseline in terms of *Performance Area*, and 0 otherwise. This resulted in eight binary

[7] https://scikit-learn.org/1.3/modules/generated/sklearn.semi_supervised.SelfTrainingClassifier.html.

Table 9 The average value of the AUPRC Macro score across all datasets for the tree-based and SSLAE algorithms (Algorithm column), for different numbers of labeled examples

Algorithm	AUPRC macro					Average ranks				
	50	100	200	350	500	50	100	200	350	500
SSLAE (ours)	$69.3_{4.6}$	$74.5_{3.6}$	$78.7_{2.7}$	$81.0_{2.6}$	$81.9_{2.0}$	5.21	5.57	6.00	6.24	6.33
RandomForest	$75.0_{2.7}$	$79.0_{2.1}$	$82.3_{1.7}$	$84.4_{1.2}$	$85.6_{1.2}$	3.03	3.42	3.93	4.46	4.69
SSL RandomForest	$68.4_{3.8}$	$73.9_{2.8}$	$78.6_{2.0}$	$81.9_{1.5}$	$83.5_{1.4}$	6.86	7.18	7.16	7.11	6.93
XGBoost	$71.5_{3.6}$	$77.1_{2.5}$	$81.4_{1.8}$	$84.1_{1.3}$	$85.4_{1.2}$	5.04	5.40	5.66	5.75	5.83
SSL XGBoost	$70.4_{4.0}$	$76.1_{2.9}$	$80.8_{2.1}$	$83.6_{1.4}$	$85.1_{1.4}$	6.10	6.30	6.26	6.43	6.43
LightGBM	$67.4_{3.5}$	$75.8_{2.4}$	$81.1_{1.8}$	$84.3_{1.4}$	$85.7_{1.3}$	6.45	5.34	4.92	4.54	4.51
SSL LightGBM	$68.0_{4.6}$	$75.2_{2.9}$	$80.7_{2.2}$	$83.7_{1.5}$	$85.1_{1.4}$	6.77	6.37	5.93	5.65	5.49
CatBoost	$\mathbf{76.3}_{2.8}$	$\mathbf{80.6}_{2.1}$	$\mathbf{84.1}_{1.5}$	$\mathbf{86.1}_{1.1}$	$\mathbf{87.1}_{1.0}$	**1.88**	**1.96**	**1.83**	**1.78**	**1.82**
SSL CatBoost	$73.7_{3.7}$	$78.7_{2.5}$	$82.8_{1.7}$	$85.0_{1.4}$	$86.4_{1.2}$	3.65	3.46	3.31	3.04	2.97

The values in the subscript show the standard deviation. Average ranks are shown separately on the right. Higher value is better for AUPRC Macro, lower value is better for ranks. The best results are shown in bold

Table 10 Bayesian t-test results comparing the performance of SSLAE against tree-based methods across different numbers of labeled examples in terms of AUPRC Macro

Algorithm	Number of labeled examples									
	50		100		200		350		500	
	p(algo)	p(rope)	p(algo)	p(rope)	p(algo)	p(rope)	p(algo)	p(rope)	p(algo)	p(rope)
RandomForest	1.00	0.00	1.00	0.00	1.00	0.00	0.9989	0.0010	0.9905	0.0093
SSL RandomForest	0.0634	0.00	0.2457	0.00	0.3793	0.0001	0.5271	0.0024	0.5555	0.0218
XGBoost	0.8679	0.00	0.9546	0.00	0.9967	0.0022	0.9421	0.0575	0.8856	0.1140
SSL XGBoost	0.4979	0.00	0.7761	0.00	0.9747	0.0214	0.9352	0.0628	0.8506	0.1477
LightGBM	0.0410	0.00	0.9216	0.0018	0.9952	0.0008	0.9409	0.0591	0.8590	0.1410
SSL LightGBM	0.0657	0.00	0.7588	0.00	0.9845	0.0046	0.9245	0.0753	0.6340	0.3659
CatBoost	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00	1.00	0.00
SSL CatBoost	1.00	0.00	1.00	0.00	1.00	0.00	0.9999	0.0001	0.9993	0.0007

targets, one per baseline. We used the same unsupervised metafeature selection procedure as in Sect. 5.2 to construct the feature space.

Similarly to Sect. E, we used the `CLUS` framework to train a multi-target PCT for MLC and to estimate the feature importance across all targets. The order of importance of the metafeatures over all 8 targets was: WL, F4 (min), NRE, CC (min), IQR (min), and Relative Frequency of Each Distinct Class (mean) as last.

Figure 7 displays the PCT trained to model the relative performance of SSLAE against all eight tree-based baselines. Each internal node represents a split on one of the metafeatures, while each leaf node contains an eight-dimensional binary prediction vector, indicating whether SSLAE outperforms the corresponding baseline. Leaf nodes are colored blue when the majority of the predictions are positive (SSLAE performs better), and orange when the majority are negative (tree-based models perform better), with color intensity reflecting the strength of the majority.

The vector ordering of the prediction targets is as follows: [Random Forest, SSL Random Forest, XGBoost, SSL XGBoost, LightGBM, SSL LightGBM, CatBoost, SSL CatBoost].

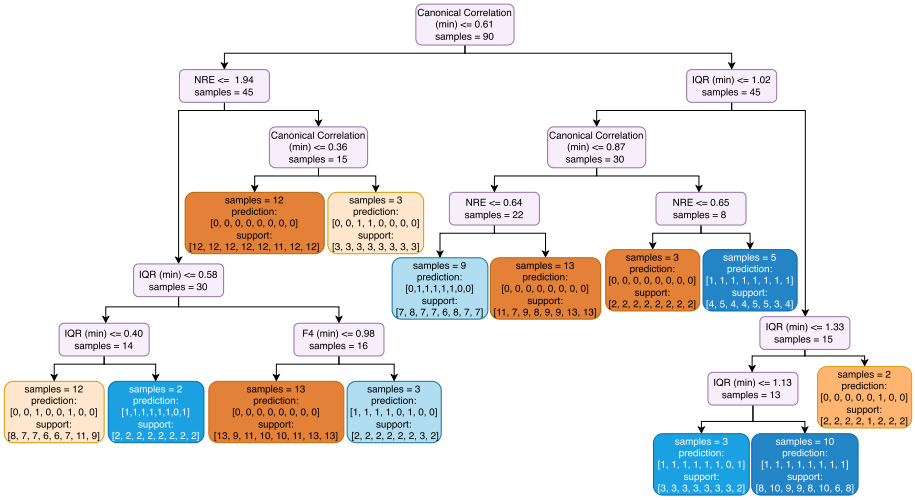

Fig. 7 Decision tree trained on the key metafeatures from the meta-analysis. Each internal node represents a split based on one of the metafeatures. *Blue* nodes indicate where SSLAE outperforms the tree-based methods (more positive examples), while *orange* nodes indicate where tree-based methods outperform SSLAE (more negative examples). The intensity of the color reflects the proportion of positive/negative examples in the node. The order of the predictions in the leaves is: [Random Forest, SSL Random Forest, XGBoost, SSL XGBoost, LightGBM, SSL LightGBM, CatBoost, SSL CatBoost]

SSLAE outperforms these methods on [28, 46, 44, 45, 40, 47, 13, 23] datasets, respectively, showing notably stronger performance over the semi-supervised tree variants and relatively weaker results against CatBoost in both supervised and SSL settings.

F.2.1 Where the tree-based methods outperform SSLAE

To identify the conditions under which the tree-based methods outperform SSLAE, we examine the branches in the PCT (Fig. 7) that lead to leaves dominated by zero vectors, indicating that SSLAE is outperformed by all eight tree-based baselines under those dataset characteristics. Three patterns emerge:

Low CC and high NRE. These are datasets that have weak alignment between features and the class labels (low minimum CC), and class imbalance is present (high NRE). This combination proves particularly problematic for SSLAE. As a neural architecture trained with both reconstruction and classification objectives, SSLAE relies on the assumption that the structure captured from unlabeled data is representative and helpful for the supervised task. When class imbalance is extreme, unlabeled data can distort the learned manifold, especially if minority classes are poorly represented and boundaries between classes are not well defined. In contrast, tree-based methods are more successful in partitioning the space even in imbalanced settings, by recursively splitting on high-importance features. Surprisingly, even simple self-training versions of tree models outperform SSLAE in this setting, likely because their base classifiers are already good at inducing strong partitions, leading to reliable pseudo-labels.

Low CC, low NRE, high IQR, and lower F4. Another challenging region for SSLAE involves datasets with complex feature-target relationships (low CC), balanced class dis-

tributions (low NRE), high variability in the feature space (high minimum IQR), and no extreme overlap in the feature space (F4 $\leq$ 0.98). Despite the availability of informative features, SSLAE struggles to find the correct decision boundaries. The autoencoder attempts to generalize across highly variable input dimensions, but with poor label alignment and no class imbalance to act as a signal, the reconstruction objective may dominate the learning without helping the classifier.

Moderate-to-high CC and low IQR. The third pattern in which tree-based methods perform better than SSLAE is when datasets have strong linear alignment between features and targets (high CC) and compact or moderately variable features (low IQR). The PCT identifies two distinct branches within this region, dependent on CC. When CC is moderately high ($0.61 < CC \leq 0.87$), there needs to be some class imbalance present (high NRE) for the tree-based models to retain advantage over SSLAE. When CC is very high ($CC > 0.87$), the class distribution needs to be balanced (low NRE) for the tree baselines to outperform SSLAE. These characteristics point to simple, low-noise datasets where class clusters are tight and can be easily separated. Tree-based models excel in such settings, as their decision paths can capture discriminative partitions early, even with minimal labeled data. SSLAE's autoencoding structure offers no significant advantage over tree-based methods in such datasets.

F.2.2 Where SSLAE outperforms the tree-based methods

Despite the strong performance of tree-based models, several regions in the PCT in Fig. 7 reveal where SSLAE outperforms all tree-based baselines. These regions correspond to leaf nodes where the prediction vector contains only ones. Two additional paths reveal settings where SSLAE outperforms every baseline model except CatBoost in its supervised form.

Very high CC, low IQR and high NRE. SSLAE is dominant on datasets with very high CC (> 0.87), compact features (low IQR) and pronounced class imbalance (high NRE). While datasets with very high alignment between the class labels and the feature space, and low IQR are more favorable for tree models, in cases where there is class imbalance, SSLAE's reconstruction loss encourages the model to retain structure across all data points, including underrepresented classes, leading to generalizable latent representations.

High CC and moderate IQR. Even when the feature space is more variable, SSLAE remains effective. In datasets with high CC and moderate IQR ($1.13 < IQR \leq 1.33$), where even the most compact feature has some dispersion, the encoder can learn the global structure well, producing features that enable the classifier to separate the classes better.

High CC and low IQR (except CatBoost). In another region, SSLAE outperforms all methods except for CatBoost on datasets with high CC (> 0.61) and low IQR. These are dense, linearly aligned problems, and generally present "easy" datasets for classification. CatBoost's edge likely comes from the properties that set it apart from the other tree baselines—its symmetric tree structure and ordered boosting.

Low CC, low NRE and moderate IQR (except CatBoost). In datasets with complex or non-linear class boundaries (low CC), balanced classes (low NRE), and some feature richness (moderate IQR), SSLAE effectively uses the unsupervised signal, building latent representations that help in the absence of strong alignment, while the lack of extreme imbalance ensures the reconstruction loss does not bias toward overrepresented classes. However, this advantage is not enough to outperform CatBoost, which avoids the local overfitting and is good at capturing subtle decision boundaries in moderately complex, balanced datasets.

This meta-analysis highlights that the relative strengths of SSLAE and tree-based models depend heavily on specific dataset characteristics, especially alignment (CC), variability (IQR), and class imbalance (NRE). Tree-based models have a strong advantage when the datasets have complex feature-target relationships (low CC) and: (1) are highly imbalanced (high NRE), or (2) have uniformly balanced classes (low NRE) and features that have a lot of variation (high IQR), but still retain some separability of classes (lower F4). In such cases, SSLAE's reliance on global representations and reconstruction objectives can mislead learning. Tree-based models also excel in datasets with densely-packed feature spaces (lower IQR), and: (1) extremely well aligned feature-target spaces (very high CC) and balanced classes (low NRE), or (2) there is moderately-high linear alignment between the features and targets ($0.61 < CC \leq 0.87$) and there are dominant classes (high NRE). Notably, in these regimes, even basic SSL on trees can outperform SSLAE, underscoring the importance of inductive bias in tabular learning.

In contrast, SSLAE shows clear advantages on datasets where the features and labels are extremely well aligned (very high CC), the variation of the feature space is low (low IQR), and there are dominant classes (moderate-to-high NRE), benefiting from the unlabeled data that augments the latent representations. SSLAE also performs better in datasets that have moderately aligned feature and target spaces (high CC) and moderate variability (moderate IQR). The cases where only CatBoost outperforms SSLAE, the datasets have moderate IQR and: (1) favor high CC (good feature-target alignment), or (2) lower CC (weak feature-target alignment) and low NRE (balanced classes).

F.2.3 SSLAE vs. CatBoost

To further investigate the relationship between SSLAE and the top-performing tree-based method, we conducted a dedicated meta-analysis comparing SSLAE to CatBoost. This tree-based model has several architectural properties that remain advantageous even when its categorical processing is not used. CatBoost's ensemble consists of symmetrical (oblivious) trees, where each level applies the same split across all branches. It also features an ordered boosting scheme that helps prevent overfit by ensuring that information leakage is minimized. CatBoost's effective default regularization settings help maintain generalization.

Figure 8 shows the decision tree trained to model the binary outcome of whether SSLAE outperforms CatBoost on a given dataset. The metafeatures used for training are the same as in previous meta-analyses (see Sect. 5.2). Each internal node represents a split on a metafeature, while leaf nodes indicate whether SSLAE or CatBoost wins. Blue leaves indicate SSLAE dominance, and orange leaves CatBoost dominance.

The tree highlights that most of the negative examples (50 out of 77) are classified following the path of lower CC, lower F4 and low IQR - indicating that CatBoost is superior on datasets with low-variance feature spaces (low IQR) that have at least one feature that can discriminate between the classes to a moderate degree (lower F4), but they also have limited feature-target alignment (lower CC). In this regime, SSLAE struggles to extract useful structure from unlabeled data. The reconstruction objective cannot compensate for poor feature-target alignment, and the compact feature space does not help in capturing intricate patterns in the latent space under these conditions. CatBoost excels by identify-

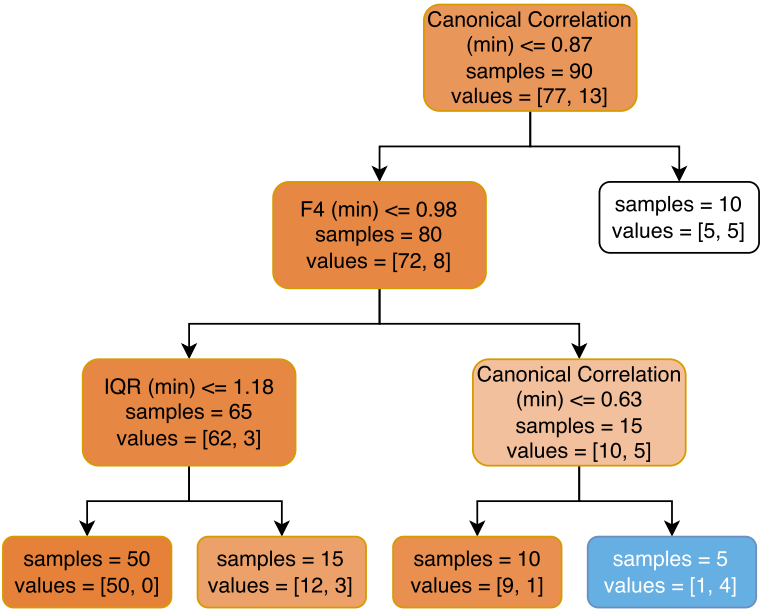

Fig. 8 Decision tree trained on the key metafeatures from the meta-analysis. Each internal node represents a split based on one of the metafeatures. *Blue* nodes indicate where SSLAE outperforms CatBoost (more positive examples), while *orange* nodes indicate where CatBoost outperforms SSLAE (more negative examples). The intensity of the color reflects the proportion of positive or negative examples in the node

ing informative splits even in dense feature spaces, ignoring uninformative features in the learning process.

As for the positive examples, the clearest success region for SSLAE emerges along the path of moderate CC ($0.63 < \text{CC} \leq 0.87$) and very high F4. This combination implies reasonably strong linear feature-target alignment and a very efficient feature space with few overlapping instances.

In the case of very high CC (>0.87), the tree indicates a tie between CatBoost and SSLAE, suggesting that when the linear alignment between the target and the features is near perfect, either approach is a viable choice.

The results of this meta-analysis confirm CatBoost's advantage in datasets with concentrated feature spaces and overall low alignment between the target and the features, where the unlabeled data used in SSLAE may introduce more noise than useful information. In contrast, SSLAE demonstrates its strength on datasets where the classes are easily discriminable by the features and there is moderate-to-high feature-target alignment.

Appendix G: Metafeature definition and calculation

In this section, we will provide details about how each of the metafeatures used in the meta-analyses of this paper is calculated.

G.1 Canonical correlation

Canonical Correlation Analysis (CCA) (Anderson, 1984) explores the relationships between two sets of variables. Unlike traditional correlation measures that assess relationships between variables in their original form, CCA seeks linear combinations of the variables in each set that are maximally correlated. These linear combinations, known as canonical variates, define new coordinate axes along which the two sets are most strongly related. The process proceeds iteratively: it first finds the pair of linear combinations with the highest correlation, then finds subsequent pairs that are uncorrelated with all previously found pairs but still maximally correlated with each other, continuing until no further dimensions remain. Mathematically, for two sets of variables X and Y, CCA solves for the weight vectors w_x and w_y that maximize:

$$\rho = \frac{\mathrm{Cov}(Xw_x, Yw_y)}{\sqrt{\mathrm{Var}(Xw_x)\mathrm{Var}(Yw_y)}} \tag{3}$$

In our setting, X represents the features, and Y (the one-hot encoded version of the labels) represents the target space. The number of canonical components is limited by $\min$(number of features, number of classes -1). For example, if there are 5 features and 3 classes, we obtain 2 canonical correlations. The CCA model is then fitted, deriving the weight vectors w_x and w_y. These weights are used to transform X and Y, projecting the data onto new axes that extract the most correlated components. As a final step, each canonical dimension is associated with a correlation coefficient ρ, calculated as the Pearson correlation between Xw_x and Yw_y (Eq. 3). These canonical correlations range from 0 to 1, where higher values indicate a stronger linear alignment between the feature and target spaces.

For the meta-analysis in this paper, we focus specifically on the minimum (i.e., worst-case) CC across all dimensions. A high minimum value indicates that every canonical dimension is strongly predictive of the target. A low minimum correlation suggests that at least one dimension has a weaker or more complex relationship with the target space.

G.2 Collective feature efficiency (F4)

The F4 measure, introduced by Orriols-Puig et al. (2010), evaluates feature efficiency in a classification task by estimating how many dataset instances can be effectively separated by hyperplanes perpendicular to individual feature axes.

For each pair of classes, the procedure iterates over the dataset one feature at a time, selecting the most discriminative feature (i.e., the one with the fewest overlapping instances) at each step. The instances that can be separated using this feature are removed from the dataset, and the process continues in the same manner with the remaining features and the reduced subset of instances. This continues until no features, or alternatively, no instances, remain in the dataset.

The final F4 score is the proportion of examples left unclassified after l iterations over the dataset, where $l \in [1, m]$ and m is the number of features. In the case where a single feature is able to classify all instances, l=1; otherwise, the process may continue up to $l = m$. Mathematically, F4 is defined as:

$$F4 = \frac{N_o(f_{\min}(T_l))}{N} \tag{4}$$

where $N_o(f_{\min}(T_l))$ is the number of instances in the overlapping region of the most discriminative feature $f_{\min}$ at iteration l, and N is the total number of instances in the dataset.

A lower F4 value indicates that a large number of instances can be correctly classified using individual features, which hints at an easier classification task. In contrast, a higher F4 value reflects significant class overlap, suggesting that the features provide limited discriminatory power.

In Sect. 5.2.1 we focus on the minimum collective efficiency across class pairs. A high minimum F4 value suggests that none of the class pairs can be efficiently separated by the features, implying substantial class overlap. A low minimum F4 value indicates that there is at least one pair of classes for which the features are collectively efficient.

G.3 Normalized relative entropy

The Normalized Relative Entropy (NRE) measures how close a dataset's class distribution is to uniform, quantifying the divergence from a perfectly balanced class scenario (Nascimento et al., 2009). It is computed using the Kullback–Leibler divergence, normalized by $2 \log k$, where k is the total number of classes. The empirical probability of class c_j is given by:

$$P(c_j) = \frac{c_j}{N} \tag{5}$$

where c_j is the number of instances in class j and N is the total number of instances in the dataset. The NRE is then calculated as:

$$NRE = \frac{\sum_{j=1}^{k} P(c_j) \log \left(\frac{P(c_j)}{1/k} \right)}{2 \log k} \tag{6}$$

Higher values of NRE indicate a greater divergence from a uniform class distribution, suggesting that some classes dominate while others are underrepresented. Lower values imply a more balanced distribution, where classes are represented more uniformly.

G.4 Wilks' Lambda

Wilks' Lambda (WL), or U-statistic, was proposed as a metafeature by Lindner and Studer (1999). It is a statistical measure used in multivariate analysis of variance (MANOVA) to test whether there are differences in the means between two sets of variables. Mathematically, it is defined as:

$$\Lambda = \prod_{j=i} \frac{1}{1 + \lambda_i} \tag{7}$$

where λ_i is the eigenvalue associated with the i-th canonical correlation. For numerical stability, Eq. 7 can be equivalently rewritten as:

$$\Lambda = e^{-\sum \log(1+\lambda_i)} \tag{8}$$

The method first calculates the canonical correlations by transforming the feature space X and the one-hot encoded target space Y, as described in Sect. G.1. Each canonical correlation ρ_i is then converted into an eigenvalue λ_i using the transformation:

$$\lambda_i = \frac{\rho_i^2}{1-\rho^2} \tag{9}$$

Finally, WL is calculated using Eq. 8, which aggregates all canonical correlations into a single summary statistic. A lower WL value indicates a stronger overall linear relationship between features and labels, meaning that the features collectively explain a greater proportion of variance in the target space. In contrast, a higher WL suggests weaker canonical correlations and implies that the feature space explains only a small fraction of the variance in the target.

G.5 Interquartile range

The Interquartile Range (IQR) measures the dispersion of a numerical distribution by computing the difference between the 75th (Q3) and 25th (Q1) percentiles. For each feature in a dataset, the IQR reflects how spread out the central 50% of the values are.

A high IQR indicates greater variability or richer structure within a feature, while a low IQR suggests a tightly clustered distribution with limited variation.

In our study, we focused on the minimum IQR across all features in a dataset, highlighting the least variable dimension. When the minimum IQR is high, even the feature with the smallest spread still shows notable variation. Conversely, if the minimum IQR is low, at least one feature has a narrow range.

G.6 Relative frequency of each distinct class

The Relative Frequency of Each Distinct Class (RFC) captures how instances in the dataset are distributed across different classes. Let c_j be the number of instances in a class j, and N the total number of instances in the dataset. The relative frequency for class j is defined as:

$$P(c_j) = \frac{c_j}{N} \tag{10}$$

In this study, we focus on the mean RFC for each dataset, which reflects the overall class balance. When this mean is close to $\frac{1}{k}$, where k is the number of classes, the class distribution is approximately uniform. Larger deviations from $\frac{1}{k}$ indicate the presence of majority or minority classes, suggesting an imbalance in the dataset's class distribution.

Acknowledgements Weacknowledge the financial support of the Slovenian Research and Innovation Agency (ARIS) via: the Research Program Knowledge Tech- nologies (grant P2-0103), including the grant for young

researchers to the first author; the Gravity project AI for Science (grant GC-0001); and the projects J1-3033, J2- 2505, J2-4452, J2-4460, J3-3070, J4-3095, J5-4575, J7-4636, and J7-4637. We were also supported by the EC via the projects ASSAS (grant number 101059682) and ELIAS (grant 101120237). We truly appreciate the valuable comments and discussions with Michelangelo Ceci, Dragi Kocev, Katharina Dost, Boshko Koloski, and Marjan Stoimchev.

Author contributions Conceptualization: Sintija Stevanoska, Jurica Levatić, Sašo Džeroski; Methodology: Sintija Stevanoska, Jurica Levatić, Sašo Džeroski; Writing - original draft preparation: Sintija Stevanoska; Writing - review and editing: Sintija Stevanoska, Jurica Levatić, Sašo Džeroski; Funding acquisition: Sašo Džeroski; Resources: Sašo Džeroski; Supervision: Sašo Džeroski; Software: Sintija Stevanoska; Visualization: Sintija Stevanoska; Data curation: Sintija Stevanoska; Investigation: Sintija Stevanoska; Validation: Sintija Stevanoska.

Funding The authors acknowledge the financial support of the Slovenian Research and Innovation Agency (ARIS) via: the Research Program Knowledge Technologies (grant P2-0103), including the grant for young researchers to the first author; the Gravity project AI for Science (grant GC-0001); and the projects J1-3033, J2-2505, J2-4452, J2-4460, J3-3070, J4-3095, J5-4575, J7-4636, and J7-4637. We were also supported by the EC via the projects ASSAS (grant number 101059682) and ELIAS (grant 101120237).

Data availability The datasets used in the experiments were sourced from the publicly available OpenML (Vanschoren et al., 2013), UCI (Kelly et al., 2019), and PMLB (Olson et al., 2017) dataset repositories.

Code availability The implementation of SSLAE is available at the following GitHub link: https://github.com/sintija-s/sslae.

Declarations

Competing interests The authors declare no competing interests.

Consent to participate Not applicable.

Consent for publication Not applicable.

Ethical approval Not applicable.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2623–2631). New York, NY: Association for Computing Machinery.

Alcobaça, E., Siqueira, F., Rivolli, A., Garcia, L. P. F., Oliva, J. T., & Carvalho, A. C. P. L. F. (2020). MFE: Towards reproducible meta-feature extraction. *Journal of Machine Learning Research, 21*(111), 1–5.

Anderson, T. W. (1984). Canonical correlations and canonical variables. In *An Introduction to Multivariate Statistical Analysis* (2nd ed., pp. 480–520). New York: Wiley.

Arik, S., & Pfister, T. (2021). TabNet: Attentive interpretable tabular learning. In *AAAI Conference on Artificial Intelligence (AAAI 2021)* (pp. 6679–6687).
Bahri, D., Jiang, H., Tay, Y., & Metzler, D. (2022). SCARF: Self-supervised contrastive learning using random feature corruption. In *International Conference on Learning Representations (ICLR 2022)*.
Benavoli, A., Corani, G., Demšar, J., & Zaffalon, M. (2017). Time for a change: A tutorial for comparing multiple classifiers through Bayesian analysis. *Journal of Machine Learning Research, 18*(77), 1–36.
Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., & Raffel, C. A. (2019). MixMatch: A holistic approach to semi-supervised learning. *Advances in Neural Information Processing Systems, 32*, 1–11.
Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.
Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. Cambridge, MA: MIT Press.
Chen, H., Tao, R., Fan, Y., Wang, Y., Wang, J., Schiele, B., & Savvides, M. (2023). SoftMatch: Addressing the quantity-quality tradeoff in semi-supervised Learning. In *International Conference on Learning Representations (ICLR 2023)*.
Chen, S., Wu, J., Hovakimyan, N., & Yao, H. (2023). ReConTab: Regularized contrastive representation learning for tabular data. https://doi.org/10.48550/arXiv.2310.18541.
Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). New York, NY: Association for Computing Machinery.
Darabi, S., Fazeli, S., Pazoki, A., Sankararaman, S., & Sarrafzadeh, M. (2021). Contrastive mixup: Self- and semi-supervised learning for tabular domain. https://doi.org/10.48550/arXiv.2108.12296.
Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*.
Engelen, J. E. V., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine Learning, 109*(2), 373–440.
Facco, E., d'Errico, M., Rodriguez, A., & Laio, A. (2017). Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific Reports, 7*(1), 12140.
Fredriksson, T., Bosch, J., & Olsson, H. H. (2025). An empirical evaluation of deep semi-supervised learning. *International Journal of Data Science and Analytics*. https://doi.org/10.1007/s41060-024-00713-8
Gille, C., Guyard, F., & Barlaud, M. (2023). A new semi-supervised classification method using a supervised autoencoder for biomedical applications. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2023)*.
Gorishniy, Y., Rubachev, I., Khrulkov, V., & Babenko, A. (2021). Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems, 34*, 18932–18943.
Gregorutti, B., Michel, B., & Saint-Pierre, P. (2017). Correlation and variable importance in random forests. *Statistics and Computing, 27*(3), 659–678.
Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., & Hutter, F. (2025). Accurate predictions on small data with a tabular foundation model. *Nature, 637*(8045), 319–326.
Huang, X., Khetan, A., Cvitkovic, M., & Karnin, Z. (2020). TabTransformer: Tabular data modeling using contextual embeddings. https://doi.org/10.48550/arXiv.2012.06678.
Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems (NIPS 2017)* (Vol. 30). Curran Associates, Inc.
Kelly, M., Longjohn, R., & Nottingham, K. (2019). *The UCI Machine Learning Repository*. https://archive.ics.uci.edu.
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems, 25*, 1–9.
Li, Y.-F., & Liang, D.-M. (2019). Safe semi-supervised learning: A brief introduction. *Frontiers of Computer Science, 13*(4), 669–676.
Lindner, G., & Studer, R. (1999). AST support for algorithm selection with a CBR approach. In *Recent Advances in Meta Learning and Future Work: Workshop Proceedings of the International Conference on Machine Learning (ICML 1999)*.
Nascimento, A. C. A., Prudêncio, R. B. C., de Souto, M. C. P., & Costa, I. G. (2009). Mining rules for the automatic selection process of clustering methods applied to cancer gene expression data. In *International Conference on Artificial Neural Networks (ICANN 2009)*.
Oliver, A., Odena, A., Raffel, C., Cubuk, E. D., & Goodfellow, I. J. (2018). Realistic evaluation of deep semi-supervised learning algorithms. *Advances in Neural Information Processing Systems, 31*, 1–12.
Olson, R. S., Cava, W., Orzechowski, P., Urbanowicz, R. J., & Moore, J. H. (2017). Pmlb: A large benchmark suite for machine learning evaluation and comparison. *BioData Mining, 10*(1), 36.

Orriols-Puig, A., Macià, N., & Ho, T. K. (2010). *Documentation for the Data Complexity Library in C++* [Technical Report]. La Salle - Universitat Ramon Llul.

Petković, M., Levatić, J., Kocev, D., Breskvar, M., & Džeroski, S. (2023). Clusplus: A decision tree-based framework for predicting structured outputs. *SoftwareX, 24*, 101526.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). Catboost: Unbiased boosting with categorical features. In *International Conference on Neural Information Processing Systems (NeurIPS 2018)* (pp. 6639–6649). Red Hook, NY: Curran Associates Inc.

Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C. A., & Li, C.-L. (2020). Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems, 33*, 596–608.

Somepalli, G., Schwarzschild, A., Goldblum, M., Bruss, C. B., & Goldstein, T. (2022). SAINT: Improved neural networks for tabular data via row attention and contrastive pre-training. https://doi.org/10.48550/arXiv.2106.01342.

Tarvainen, A., & Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in Neural Information Processing Systems, 30*, 1–10.

Vanschoren, J., Rijn, J. N., Bischl, B., & Torgo, L. (2013). Openml: Networked science in machine learning. *SIGKDD Explorations, 15*(2), 49–60.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*, 1–11.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning (ICML 2008)*.

Xie, Q., Luong, M.-T., Hovy, E., & Le, Q. V. (2020). Self-training with noisy student improves imageNet classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*.

Yang, X., Song, Z., King, I., & Xu, Z. (2023). A survey on deep semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering, 35*(9), 8934–8954.

Yoon, J., Zhang, Y., Jordon, J., & Schaar, M. (2020). Vime: Extending the success of self- and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems, 33*, 11033–11043.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems, 27*, 1–9.

Zhang, B., Wang, Y., Hou, W., Wu, H., Wang, J., Okumura, M., & Shinozaki, T. (2024). Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Advances in Neural Information Processing Systems, 34*(2021), 18408–18419.

Zheng, M., You, S., Huang, L., Wang, F., Qian, C., & Xu, C. (2022). SimMatch: Semi-supervised learning with similarity matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2022)*.