FSL-LFMG: Few-Shot Learning with Augmented Latent Features and Multitasking Generation for Enhancing Multiclass Classification on Tabular Data

Aviv A. Nur[1], Chun-Kit Ngan[1] and Rolf Bardeli[2]
[1]*Data Science Program, Worcester Polytechnic Institute, 100 Institute Rd., Worcester, MA, U.S.A.*
[2]*thyssenkrupp Materials Services GmbH, Essen, Germany*
{*aanur, cngan*}*@wpi.edu, rolf.bardeli@thyssenkrupp-materials.com*

Keywords: Few-shot Learning, Machine Learning, Deep Learning, Multiclass Classification, Autoencoders, Random Forest, CatBoost, One vs Rest Classifier, STUNT, Prototypical Network, Tabular Data.

Abstract: In this work, we propose advancing ProtoNet that employs augmented latent features (LF) by an autoencoder and multitasking generation (MG) by STUNT in the few-shot learning (FSL) mechanism. Specifically, the achieved contributions to this work are threefold. First, we propose an FSL-LFMG framework to develop an end-to-end few-shot multiclass classification workflow on tabular data. This framework is composed of three main stages that include (i) data augmentation at the sample level utilizing autoencoders to generate augmented LF, (ii) data augmentation at the task level involving self-generating multitasks using the STUNT approach, and (iii) the learning process taking place on ProtoNet, followed by various model evaluations in our FSL mechanism. Second, due to the outlier and noise sensitivity of K-means clustering and the curse of dimensionality of Euclidean distance, we enhance and customize the STUNT approach by using K-medoids clustering that is less sensitive to noisy outliers and Manhattan distance that is the most preferable for high-dimensional data. Finally, we conduct an extensive experimental study on four diverse domain datasets—Net Promoter Score segmentation, Dry Bean type, Wine type, and Forest Cover type—to prove that our FSL-LFMG approach on the multiclass classification outperforms the Tree Ensemble models and the One-vs-the-rest classifiers by 7.8% in 1-shot and 2.5% in 5-shot learning.

1 INTRODUCTION

The increasing volume of data across various sectors, such as telecommunications, agriculture, and finance, has led to a pressing demand for effective multiclass classification (Hollmann et al., 2022). For instance, the telecom industry has utilized many machine-learning (ML) models, such as random forest, decision trees, and discriminant feature analysis, to forecast customer attrition and enhance investment optimization. These techniques strive to predict customer behavior and enhance investment choices (Sikri et al., 2024; Şahin, 2023; Abdulsalam et al., 2022; Loukili et al., 2022). In the field of agriculture, ML and deep learning (DL) improve crop monitoring, yield estimation, and productivity, showcasing their essential impact on improving farm management and productivity (Attri et al., 2024; Khan et al., 2023; Adebiyi et al., 2020). In the finance industry, ML and DL address tasks such as risk assessment, pricing, and the development of optimal insurance packages through various methodologies such as artificial neural networks and clustering algorithms. These approaches underscore the crucial role of data and the necessity to adapt to changing financial patterns for improved decision-making and efficiency (Matloob et al., 2021; Blier-Wong et al., 2020).

The widespread use of data in multiple industries typically involves the utilization of tabular data that has been demonstrated by a 2023 Kaggle survey of 14,000 data scientists. The poll indicated that a substantial portion of professionals within those industries, ranging from 50% to 90%, relied on tabular data in their work environments (Tunguz et al., 2023; Sun et al., 2019). The inclination towards tabular data presents distinct challenges such as high dimensionality, heterogeneity, and critical interdependencies among features, which are not found in images or other data modalities (Borisov et al., 2022). Despite these challenges, the adoption of innovative multiclass classification methods is still growing demonstrating the importance of those methods in enhancing

Nur, A., Ngan, C. and Bardeli, R.
FSL-LFMG: Few-Shot Learning with Augmented Latent Features and Multitasking Generation for Enhancing Multiclass Classification on Tabular Data.
DOI: 10.5220/0012934200003837
In *Proceedings of the 16th International Joint Conference on Computational Intelligence (IJCCI 2024)*, pages 531-542
ISBN: 978-989-758-721-4; ISSN: 2184-3236

decision-making and operational efficiency across industries.

Presently, the existing approaches for multiclass classification on tabular data can be broadly divided into two categories: DL Models and Tree Ensemble (TE) Models (Shwartz-Ziv and Armon, 2022). Recent advances in DL models, such as TabNet, Neural Oblivious Decision Ensembles (NODE), and Disjunctive Normal Formulas (DNF-Net), have demonstrated exceptional outcomes across diverse domain datasets (Arik and Pfister, 2021; Katzir et al., 2020; Popov et al., 2019). These models possess the ability to delve into intricate connections among features, resulting in heightened efficiency and performance for tasks involving high-dimensional, structured data. Each model utilizes distinct mechanisms for processing feature selection, which further improves their overall effectiveness. However, these models present challenges in terms of complexity and computation, as well as interpretability. On the other hand, the TE models, including Random Forest and Gradient Boosting, offer enhanced interpretability and reduced computational complexity. In particular, Gradient Boosting, such as XGBoost, exhibits significantly better performance in tabular data compared to DL models (Borisov et al., 2022; Shwartz-Ziv and Armon, 2022). However, the remarkable performance of these models is highly reliant on the utilization of copious amounts of training data, which are inadequate in some domains and require substantial storage space (Wang et al., 2021; Tian et al., 2020). Additionally, if the amount of training data is insufficient, it results in an overfitted model that lacks generalizability.

Few-shot learning (FSL) is an ML technique that trains on a small number of labeled samples, typically one to five samples per class, providing a potential solution to the aforementioned issues (Li et al., 2023; Wang et al., 2020). This technique enables efficient learning of multiclass classification tasks with only a limited amount of data (Parnami and Lee, 2022). Although FSL has achieved noteworthy success in the domain of image classification, research on these techniques on tabular data has been widely underexplored (Nam et al., 2023). Furthermore, the application of FSL in conjunction with TE models on tabular data is very challenging because of the models' limitations in generalizing on a few data samples per class.

An effort to address the limitations of TE models led to the implementation of the One-vs-the-rest (OvR) multiclass technique (sklearn, 2024). This method is specifically designed for multiclass classification and involves dividing the tasks into a series of binary tasks. The OvR classifier strategy can be integrated into various existing ML conventional models, including TE models, as the base estimators. This technique is expected to enhance the classification capabilities of tree-based models by splitting tasks into binary tasks. However, there is an opportunity that this technique provides suboptimal results due to the potential loss of significant data characteristics, such as complex inter-class correlation and interaction, which could result in unsatisfactory performance in classification tasks.

To improve and generalize the ability of models is to augment the data, thereby increasing data variability. Data augmentation can be conducted either at the sample or task level (Zhang and Liu, 2023). At the sample level, typical methods for image data involve modifying pixel properties through actions, such as rotation, scaling, cropping, and other similar approaches. These actions are performed to increase the variety of data. On the contrary, when it comes to tabular data, there is currently no recognized approach that can complete this data augmentation task. In order to tackle this issue, we investigate the use of autoencoders to extract significant latent features. Two methods have been experimented in this context: one is to directly apply the extracted latent features to a classifier, and the other is to concatenate these encoded latent features with the original data in order to enhance the number of features. The main contributions in this work is to utilize the knowledge found in large datasets to improve the accuracy of multiclass classification that leads to higher levels of accuracy. Despite its potential merits, this methodology is not reliable for comprehending new tasks, as it primarily concentrates on a single task, i.e., the process of learning to predict a single outcome, including binary, multiclass, or continuous values, respectively, from a labeled dataset. Hence, in addition to latent features, it is essential to employ task-level data augmentation methods that can improve the precision of classification, while also facilitating the model to efficiently learn new tasks. The task-level augmentation entails generating new tasks to offer the model a broader range of learning experience.

The incorporation of Self-generated Tasks from unlabeled Tables (STUNT) is recognized as a prominent task-level data augmentation strategy (Nam et al., 2023). Through the treatment of data as unlabeled, this technique has the potential to generate various tasks for a single dataset. This outcome is attained by applying the K-means clustering method to create new labels. It is anticipated that the generation of self-tasks leads to effective generalization, as the model acquires knowledge from multiple tasks, i.e., the process of jointly learning to predict multiple outcomes on inputs of the same dataset, simultaneously. In order to capture the generalized knowledge, a meta-learning scheme called Prototypical Networks (ProtoNet) is utilized as a classifier (Snell et al., 2017). In contrast to ProtoNet, tree-based models lack capabilities to perform generalization on small datasets because their complicated structures tend to overfit specific training samples instead of capturing broader patterns. A lack of data also makes methods, such as bagging, less useful, resulting in trees that are not varied and less-than-ideal decisions at splits (Biau and Scornet, 2016). ProtoNet has proven to be highly accurate and effective across various types of data (Nam et al., 2023; Yu et al., 2022; Snell et al., 2017). This approach successfully generates representative prototypes or mean embeddings for each class by utilizing Euclidean distance to determine the proximity of a target task to its prototype.

To incorporate the advantages of the above methods into our approach, we propose advancing ProtoNet that employs augmented latent features (LF) by an autoencoder and multitasking generation (MG) by STUNT in the few-shot learning mechanism. Specifically, the achieved contributions to this work are threefold. First, we propose an FSL-LFMG framework to develop an end-to-end few-shot multiclass classification workflow on tabular data. This framework is composed of three main stages that include (i) data augmentation at the sample level utilizing autoencoders to generate augmented LF, (ii) data augmentation at the task level involving self-generating multitasks using the STUNT approach, and (iii) the learning process taking place on ProtoNet, followed by various model evaluations in our FSL mechanism. Second, due to the outlier and noise sensitivity of K-means clustering (Arora et al., 2016) and the curse of dimensionality of Euclidean distance (Yu et al., 2022), we enhance and customize the STUNT approach by using K-medoids clustering that is less sensitive to noisy outliers and Manhattan distance that is the most preferable for high-dimensional data. Finally, we conduct an extensive experimental study on four diverse domain datasets—Net Promoter Score (NPS) segmentation, Dry Bean type, Wine type, and Forest Cover type—to prove that our FSL-LFMG approach on the multiclass classification outperforms the TE models and the OvR classifiers by 7.8% in 1-shot and 2.5% in 5-shot learning.

The remainder of this paper is organized as follows: Section 2 introduces our proposed FSL-LFMG framework. Section 3 describes the process that learns the LF by using autoencoders. Section 4 explains the MG approach by using the STUNT. Section 5 explains the meta learning process using ProtoNet. Section 6 details the experimental results, analyses, and discussion. Finally, in Section 7, we provide a conclusion and outline our future work for this project.

2 FSL-LFMG FRAMEWORK

In this section, we describe and explain our proposed FSL-LFMG framework that is an end-to-end pipeline consisting of four main modules shown in Figure 1. The modules include Data Preprocessing (DP), Latent Features Augmentation (LFA), Multitasking Generation (MG), and Prototypical Network (PN). First, raw tabular data is passed into the DP module that processes and cleans the data in three separate steps in sequence. In STEP 1, the data is divided into three parts, i.e., training set, validation set, and test set. The ratio among them is 64:16:20. For instance, in the NPS telecom dataset, which comprises 100,000 samples, the dataset is divided into 64,000 samples for the training set, 16,000 samples for the validation set, and 20,000 samples for the test set. In STEP 2, to deal with numerical features in the dataset, we apply the Min-Max scaler to ensure that all those features are on the same scale, for instance, between 0 and 1. The Min-Max scaling is defined by the following equation:

$$x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \tag{1}$$

where x_{scaled} is the new scaled value, $x_{\min}$ is the minimum value of the feature, $x_{\max}$ is the maximum value of the feature, and x is the original value. This approach helps minimize the influence of varying scales and measurements among different numerical features. For example, the 'data usage' feature of our NPS dataset has a range of 100 to 90,000, while the 'upload' feature has a range of 1 to 1,250. By using the Min-Max scaler, we transform these features into the same range of 0 to 1. In STEP 3, to manage categorical features, we employ the one-hot encoding technique to encode categorical data into numerical ones suitable for ML models to understand. For instance, the 'tariff' feature has three unique categorical values, i.e., Level 1, Level 2, and Level 3. By performing the one-hot encoding technique on this feature, we convert the values in the categorical variable into a numeric form, i.e., the binary variable (1/0), which can be understood by the model while maintaining its categorical nature.

After the data is cleaned, they are passed into the LFA module that augments the existing features with the latent features learned from the autoencoder. This

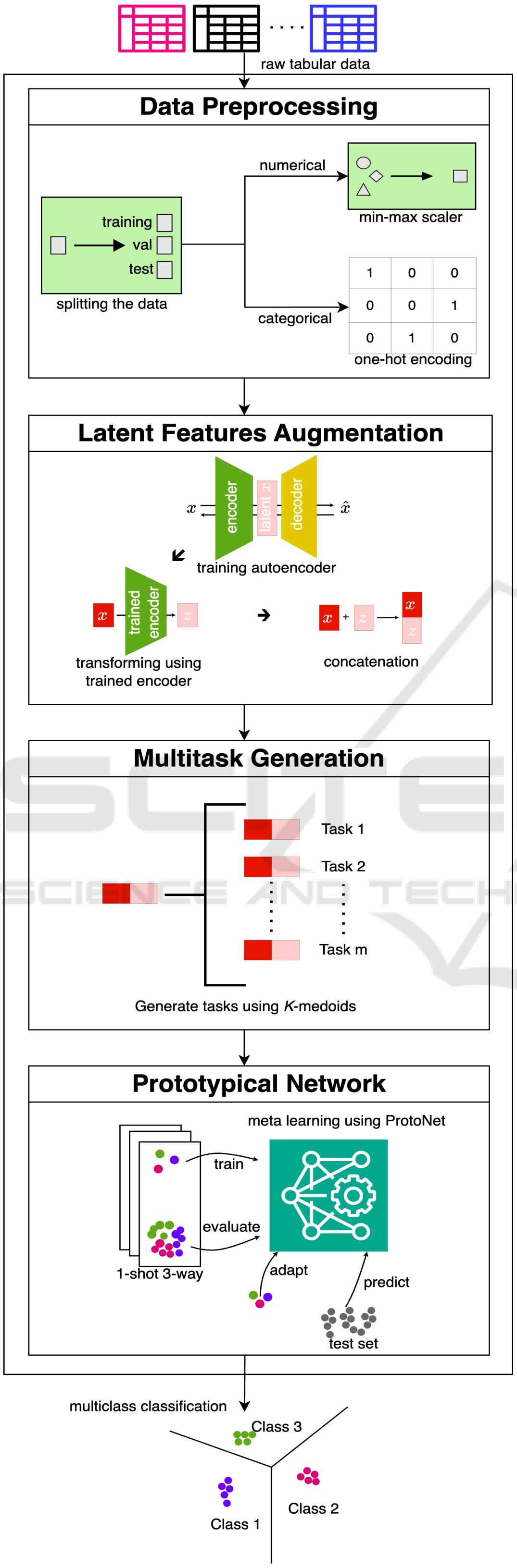


Figure 1: FSL-LFMG Framework.

module aims to increase the data variation at the sample level by increasing the number of input features. Specifically, the core of the LFA module involves training the autoencoder and then utilizing the trained encoder to extract the most important and compact latent features, which are then concatenated with the existing features to obtain a larger number of features. This process is further detailed in Section 3.

After the features are augmented, they are fed into the MG module, where the STUNT methodology (Nam et al., 2023) is implemented to facilitate the multitasking generation process. This approach is designed to address the challenges of FSL and aims to enhance the diversity of data at the task level by generating a range of diverse tasks. In each task, a random selection process is conducted with a specified number of samples according to the designated support and query sets. The support set consists of examples that are used for training, while the query set contains examples that are used for testing. For instance, in Task i-th, where $i = 1, \ldots, m$, in the 1-shot setting with three classes, one sample is randomly selected from each class, so the number of support set is three. Then the number of queries is set at fifteen samples per class to evaluate the performance of the model in Task i-th. This process is replicated in the predetermined number of tasks. In our work, we employ K-medoids rather than K-means for the task generation process, as K-medoids is a more robust method for overcoming the influence of noisy outliers in the dataset. In contrast to the original method, which utilizes K-means for segmentation and produces pseudo labels that resemble existing labels, our work employs K-medoids to improve the accuracy and reliability of the results. This process is thoroughly explained in Section 4.

Finally, the tasks corresponding to the selected data and features are passed into our meta-learning paradigm of the PN module, which effectively generalize from minimal examples by shaping a metric space conducive to distance-based classification which is explained in detail in Section 5. This holistic framework not only addresses the complexities inherent in FSL but also sets a new benchmark for processing tabular datasets more efficiently and accurately.

3 LATENT FEATURES LEARNING AND AUGMENTATION

The process of latent features learning and augmentation consists of three steps, as follows:

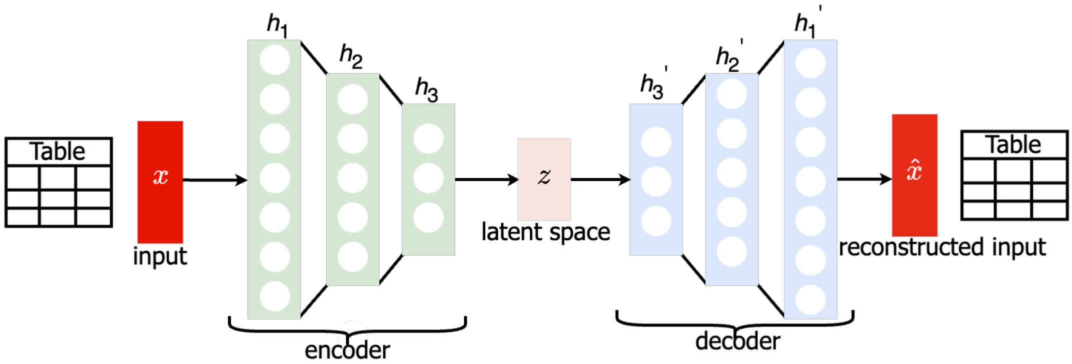


Figure 2: A high-level architecture of autoencoders adapted from Ye and Wang (2023).

Step 1: Encoding. Autoencoders are employed in high-dimensional data for feature extraction to generate compact representations that accurately reflect the original data (Ye and Wang, 2023). This technique is particularly advantageous for image and video data, as it minimizes storage requirements. For tabular data, autoencoders extract critical features that aim to replicate the original dataset's characteristics fully.

The training of autoencoders shown in Figure 2 utilizes a substantial portion of data to capture prevalent attributes, yielding the encoder, depicted by green layers that consist of three hidden layers (i.e., h_1, h_2, and h_3) to map the input features x to a latent representation z that extracts the significant features. The decoder, shown in the blue layers that consist of three symmetrical hidden layers with the encoder's hidden layers (i.e., h_1', h_2', and h_3'), reconstructs the input features $\hat{x}$ from z, aiming to minimize the difference between x and $\hat{x}$. The trained encoder can then transform the new input data into the latent representations z that is useful for the downstream tasks, including data augmentation and classification, respectively.

Mathematically, the encoder E and decoder D can be formulated in the following transformations:

$$\textbf{Encoder}: \begin{cases} h_1 = \sigma(W_1^{(E)}x + b_1^{(E)}) \\ h_2 = \sigma(W_2^{(E)}h_1 + b_2^{(E)}) \\ h_3 = \sigma(W_3^{(E)}h_2 + b_3^{(E)}) \\ z = \sigma(W_4^{(E)}h_3 + b_4^{(E)}) \end{cases}, \tag{2}$$

$$\textbf{Decoder}: \begin{cases} h_3' = \sigma(W_4^{(D)}z + b_4^{(D)}) \\ h_2' = \sigma(W_3^{(D)}h_3' + b_3^{(D)}) \\ h_1' = \sigma(W_2^{(D)}h_2' + b_2^{(D)}) \\ \hat{x} = \sigma(W_1^{(D)}h_1' + b_1^{(D)}) \end{cases}, \tag{3}$$

where x is a set of input features, z is a set of latent features, $\hat{x}$ is a set of reconstructed input features, W_i is a weight matrix, b_i is a bias, h_j is a hidden layer, and σ is the ReLU activation function shown in Equation (4), for $i = 1,2,3,4$ and $j = 1,2,3$,

$$\sigma = \mathrm{ReLU}(x) = \max(0,x). \tag{4}$$

In this work, we develop a three-layer symmetric autoencoder architecture with the ReLU activation functions to regularize the process. During the learning process, we utilize the mean squared error (MSE) loss criterion to optimize the architecture by using the Adam optimizer with a learning rate set at 10^{-3}. To prevent overfitting, the early stopping is implemented with a patience parameter of 5 that not only ensures the optimal model performance but also reduces the likelihood of training divergence. The optimization of autoencoder quantified using MSE between the original inputs x and the reconstructed outputs $\hat{x}$ can be defined as follows:

$$MSE = \mathcal{L}(x,\hat{x}) = \frac{1}{N}\sum_{i=1}^{N} \|x_i - \hat{x}_i\|^2. \tag{5}$$

This loss function *MSE* shown in Equation (5) guides the training process that encourages the model to find the most representative latent features, where N is the total number of data instances in the training set. The Adam optimizer, which adaptively adjusts the learning rate for each parameter based on the estimations of first and second moments of the gradients and the learning rate $\eta = 10^{-3}$, can be defined as follows:

$$\mathbf{W}^{(E)}, \mathbf{b}^{(E)}, \mathbf{W}^{(D)}, \mathbf{b}^{(D)} \leftarrow \mathrm{Adam}(\nabla\mathcal{L}, \eta). \tag{6}$$

Once the encoder is trained, it transforms the original data x to the latent features z shown in Figure 3.

Step 2: Min-Max Scaling of Latent Features. After obtaining the latent representations z, min-max scaling is applied, using Equation (1), to ensure that the latent features have the same scale as the original features. This step is crucial to maintain consistency

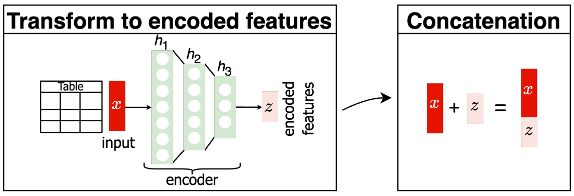


Figure 3: Process of augmentation latent features.

and enhance the effectiveness of the subsequent data augmentation process.

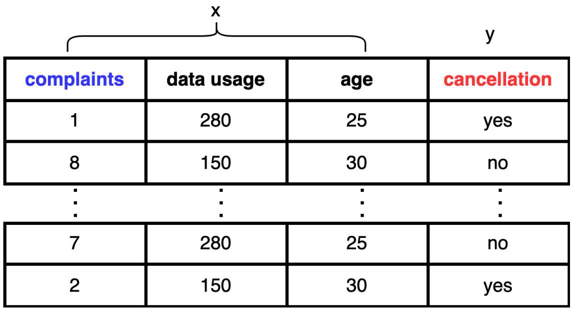
Step 3: Concatenation with Original Input Features. Once scaled, these latent features are concatenated with the original dataset to generate the augmented data, denoted as $x_{\text{augmented}}$, shown in Equation (7). This augmented dataset enhances the overall feature set and increases data variability, which is expected to improve the model's generalization capabilities.

$$x_{\text{augmented}} = [x : z] \tag{7}$$

This augmented dataset is then used in the MG process.

4 MULTITASKING GENERATION

Data augmentation at the task level is to build the common knowledge by performing multiple tasks from a dataset. STUNT is a specific framework that can generate those multiple diverse tasks using the K-means clustering as a pseudo-label generator (Nam et al., 2023). The idea behind this approach is that each feature can serve as a label for the other features. For instance, Figure 4 is an original dataset with three input features (x) (i.e., complaints, data usage, and age) and a target binary variable (y) (i.e., cancellation (yes/no)). In this example, we assume that there is a positive correlation between complaints and cancellation, from which we can use complaints as a new target variable and use the other variables as the input features. By generalizing this concept, we can first consider the data that is unlabeled; and this STUNT method randomly selects some features and then utilizes the K-means algorithm to perform the clustering. In each cluster, the pseudo-label can be obtained by computing the center of the cluster, also known as the centroids. By iteratively applying the task generation process using different combinations of features, we can generate the corresponding new datasets with their own labels from the original dataset.



complaints	data usage	age	cancellation
1	280	25	yes
8	150	30	no
⋮	⋮	⋮	⋮
7	280	25	no
2	150	30	yes

Figure 4: Example data from telecom dataset, adapted from Nam et al. (2023).

More specifically, following Nam et al. (2023), given a dataset $\mathbf{X}$ of unlabeled tabular data, we summarize their approach and formalize the process as follows:

Step 1: Masking Ratio Sampling. Sample a masking ratio p from a uniform distribution over a range of hyperparameter $[r_1, r_2]$, where $0 < r_1 < r_2 < 1$.

Step 2: Binary Mask Creation. Generate a random binary mask $\mathbf{m} \in \{0,1\}^d$, where d is the number of features and the sum of elements in $\mathbf{m}$ is $\lfloor dp \rfloor$ where $\lfloor \cdot \rfloor$ is floor function applied to dp.

Step 3: Column Selection. Use the mask $\mathbf{m}$ to select columns from the unlabeled data $\mathbf{X}$. The selected data is denoted by $sq(\mathbf{x} \circ \mathbf{m})$, where $\circ$ indicates element-wise multiplication, and $sq(\cdot)$ represents a squeezing operation that removes elements corresponding to zeros in $\mathbf{m}$.

Step 4: K-means Clustering. Apply K-means clustering on the selected columns to generate pseudo-labels $\tilde{\mathbf{y}}$. The objective function for the K-means is given by:

$$\min_{C \in \mathbb{R}^{\lfloor dp \rfloor \times k}} \frac{1}{N} \sum_{i=1}^{N} \min_{\tilde{\mathbf{y}}_i \in \{0,1\}^k} \| sq(\mathbf{x}_i \circ \mathbf{m}) - C\tilde{\mathbf{y}}_i \|_2^2, \tag{8}$$

$$\text{such that} \quad \tilde{\mathbf{y}}_i^T \mathbf{1}_k = 1,$$

where C is the centroid matrix, k is the number of centroids, $\mathbf{1}_k$ is a vector of ones and $\mathbf{x}_i$ represents the i-th sample in the dataset. $\| \cdot \|_2^2$ indicates squared Euclidean distance, used here to measure the distance between the transformed data points and the cluster centroids.

Step 5: Data Perturbation. To prevent trivial learning by the classifier, perturb the selected column features by:

$$\tilde{\mathbf{x}} := \mathbf{m} \circ \hat{\mathbf{x}} + (1 - \mathbf{m}) \circ \mathbf{x}, \tag{9}$$

where $\hat{\mathbf{x}}$ is sampled from the empirical marginal distribution of each column feature.

Step 6: Task Definition. The generated task $\mathcal{T}_{\text{STUNT}}$ from the process is defined as:

$$\mathcal{T}_{\text{STUNT}} := \{(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i)\}_{i=1}^{N}. \tag{10}$$

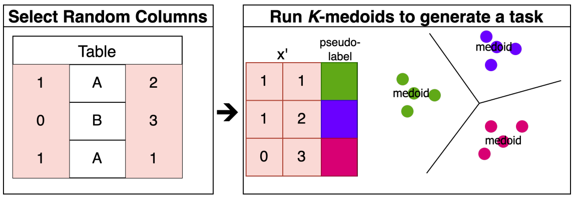


Figure 5: Multitasking generation using K-medoids, adapted from Nam et al. (2023).

In our work, we enhance and customize the STUNT approach by using the K-medoids as an alternative clustering method to the K-means shown in Figure 5 that is a high-level overview of the modified STUNT approach. We propose this approach because the K-medoids clustering is more robust to outliers, as it uses the actual data points as the centers, also known as medoids, thereby avoiding the influence of extreme values, unlike K-means. Thus, in Equation (8) above, we change to use the K-medoids clustering instead of the K-means clustering.

More precisely, we apply the K-medoids clustering on the selected features to generate the pseudo-label $\tilde{\mathbf{y}}$. The objective function for the K-medoids clustering using the Manhattan distance is given by:

$$\min_{C\in\mathbb{R}^{\lfloor dp\rfloor\times k}} \frac{1}{N}\sum_{i=1}^{N} \min_{\tilde{\mathbf{y}}_i\in\{0,1\}^k} \|sq(\mathbf{x}_i \circ \mathbf{m}) - C\tilde{\mathbf{y}}_i\|_1, \tag{11}$$

$$\text{such that} \quad \tilde{\mathbf{y}}_i^T \mathbf{1}_k = 1,$$

where C is the centroid matrix which is the medoids matrix, k is the number of medoids, and $\mathbf{x}_i$ represents the i-th sample in the dataset. $\|\cdot\|_1$ represents the Manhattan distance to measure the distance between the transformed data points and the cluster medoids. The medoids are selected from the dataset $\mathbf{X}$, and each data point $\mathbf{x}_i$ is assigned to the nearest medoid based on the Manhattan distance.

After the completion of diverse tasks generation, the learning process is undertaken by ProtoNet, which aims to develop a model capable of generalizing based on diverse inputs from various tasks.

5 PROTOTYPICAL NETWORKS

ProtoNet is a neural network that employs meta-learning to learn variety of tasks. Specifically, after the MG module generates the diverse tasks by Equation (10), data samples are taken from a collection of those tasks. For each task, the support $(\mathcal{S})$ set and the query $(\mathcal{Q})$ set are then selected. As shown in Figure 6, i.e., a high-level framework of few-shot learning concept using ProtoNet, the model is trained on the support set and evaluated on the query set, with the meta-learner being updated based on the query set's performance. Following this, the meta-learner is utilized for adaptation and prediction on a new test set using a fresh batch of labeled data, with a small portion serving as the support test set.

Several advantages have been identified by Nam et al. (2023) regarding the use of ProtoNet as an embedding function or learner in few-shot settings, including flexible centroids, agnostic application, and optimal performance. **Flexible centroids** refer to the adaptability of ProtoNet to various cases by adjusting the number of k or centroids. **Agnostic application** allows for the direct application of this architecture to tabular data without significant difficulty. Additionally, ProtoNet has demonstrated **strong performance** in various modalities, as reported in some studies (Nam et al., 2023; Yu et al., 2022; Snell et al., 2017). This study also highlights the flexibility of ProtoNet as a benefit. The original ProtoNet employs the Euclidean distance for metric learning, but the research work conducted by Yu et al. (2022) on image classification suggests that the Manhattan distance is a strong substitute for this metric, potentially improving performance. It would therefore be intriguing to apply this substitution to tabular data, as the Manhattan distance has advantages over the Euclidean one, especially in high-dimensional data. The differences between the Euclidean and the Manhattan distance are visually shown in Figure 7. The Euclidean distance (i.e., the red line) measures the shortest straight-line distance between the two points that is calculated by using the Pythagorean theorem. The Manhattan distance (i.e., the blue path) measures the distance between the two points by summing the absolute differences of their coordinates.

In this work, the architecture of ProtoNet follows

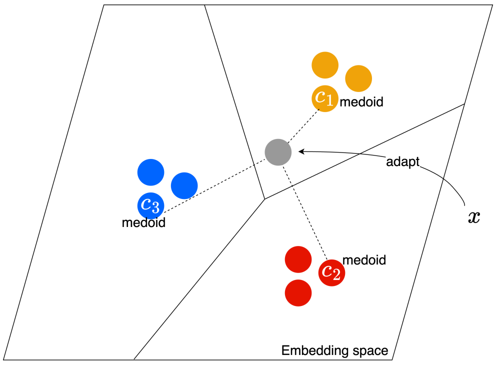


Figure 6: Few-shot learning concept using ProtoNet adapted from Snell et al. (2017).

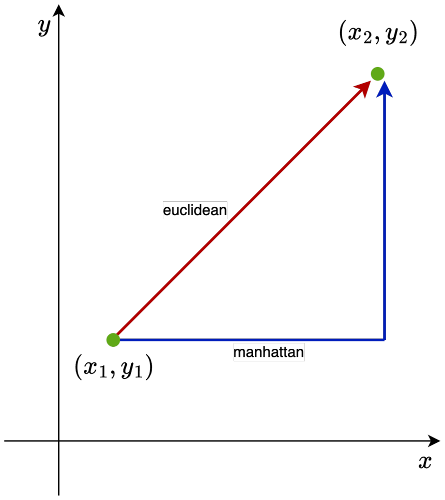


Figure 7: Comparison between Euclidean distance and Manhattan distance.

a multilayer perceptron (MLP) design that consists of a 2-layer fully connected neural network with a hidden dimension of 1,024, as recommended by Nam et al. (2023). Given a task selected by Equation (10), we construct the classifier using the episodic training way. Training episodes are created by selecting random subsets of classes and examples, with some examples acting as $(\mathcal{S})$ and $(\mathcal{Q})$ from each task. The Prototypical networks create a prototype or an average representation for each class using an embedding function $\mathcal{F}_\theta$ with a learnable parameter θ. Each prototype is the mean of the embedded points in its class calculated as follows:

$$c_k = \frac{1}{|\mathcal{S}_k|} \sum_{(\tilde{\mathbf{x}}_i, \tilde{y}_i) \in \mathcal{S}_k} \mathcal{F}_\theta(\tilde{\mathbf{x}}_i), \qquad (12)$$

where $\mathcal{S}_k$ is the support set associated with the prototype k. Using a distance function d, which is the Manhattan distance, the network calculates the probability of a class for a query point $\tilde{x}_i$ by taking a softmax over distances to the prototypes:

$$p_\theta(y = k \mid \tilde{\mathbf{x}}_i; \mathcal{S}) = \frac{\exp(-d(\mathcal{F}_\theta(\tilde{\mathbf{x}}_i), c_k))}{\sum_{k'} \exp(-d(\mathcal{F}_\theta(\tilde{\mathbf{x}}_i), c_{k'}))}, \qquad (13)$$

where the Manhattan distance is

$$d(\mathcal{F}_\theta(\tilde{\mathbf{x}}_i), c_k) = \sum_{i=1}^{N_{\mathcal{S}_k}} \|\mathcal{F}_\theta(\tilde{\mathbf{x}}_i) - c_k\|_1 . \qquad (14)$$

Next, we compute the cross-entropy loss on the classifier p_θ as follows:

$$\mathcal{L}_{CE}(p_\theta, \tilde{y}_i) = -\sum_{j=1} (\tilde{y}_i)_j \log p_\theta. \qquad (15)$$

The ultimate objective is to minimize the meta-learning loss over diverse tasks generated by Equation (10) as follows:

$$\mathcal{L}_{meta}(\theta, \mathcal{Q}) := \sum_{(x_i, y_i) \in \mathcal{Q}} \mathcal{L}_{CE}(p_\theta, \tilde{y}_i). \qquad (16)$$

After we finish the model training, we use the model obtained to adapt with the few-shot sample (x_i, y_i), where y_i is the existing label from 100 different seed. Finally, using an independent test set, we compute the mean test accuracy of this few-shot learning process.

6 EXPERIMENTAL RESULTS, ANALYSES, AND DISCUSSION

In our experimental studies, we utilize four different domain datasets. Three of them are publicly available from the UCI Machine Learning Repository, including Wine (Aeberhard and Forina, 1991), Dry Bean (UCI, 2020), and Forest Cover Type (Blackard, 1998). One of them is a proprietary dataset provided by a telecommunications corporation, specifically related to the NPS segmentation. The Wine dataset contains 178 instances and 13 attributes that are used for the classification of wine variants. The Dry Bean dataset includes 13,611 instances and 16 attributes that are aimed at classifying different types of beans. The Forest Cover Type dataset is composed of 581,012 instances and 54 attributes that are used for predicting forest cover types based on cartographic variables. The proprietary NPS segmentation dataset consists of customer demographic profile and feedback data, segmented into promoters, passives, and detractors based on their likelihood to recommend the company's services. Table 1 provides the detailed descriptions of these four datasets, including the number of instances and attributes, as well as the primary classification objective for each dataset.

During our experimental evaluations, we examine various TE models as the benchmark, including Random Forest, CatBoost, and One-vs-Rest (OvR) Classifier. We also combine these three baseline models with augmentation techniques utilizing autoencoders to enhance the feature representation and improve classification performance. Random Forest, known for its robustness and ease of implementation, provides a strong baseline through its ensemble of decision trees. CatBoost, a gradient boosting algorithm, is particularly effective in handling categorical features and improving accuracy. The OvR Classifier, a strategy for multiclass classification, breaks down the problem into multiple binary classification tasks. In addition to these models, we employ the standard STUNT framework as a comparison benchmark for our proposed method. The STUNT framework,

Table 1: Summary of Datasets.

No	Name	# N	# features	Description	Source
1	Net Promoter Score (NPS) segmentation	100,000	11	Predict a customer segmentation into three groups: promoters, passives, detractors, based on demographics and customer experiences.	Private
2	Drybean types	13,611	16	Predict seven various sorts of dry beans according to market conditions, including form, shape, type, and structure.	Public
3	Wine types	178	13	Predict three different types of wines using the findings of a chemical analysis of wines grown in the same region of Italy.	Public
4	Forest cover types	581,012	54	Predict seven forest cover classes based on variables such as elevation, aspect, slope, hill shade, soil type, and others.	Public

Table 2: Baselines Details.

No	Methods	Description
1	Random Forests (RF)	An ensemble of tree predictors, where each tree's predictions are based on the values of a random vector that is separately sampled and has the same distribution for all trees in the forest.
2	CatBoost (CB)	A gradient boosting method that utilizes binary decision trees as its base predictors.
3	Autoencoders (AE) + Classifier	Using only encoded features to be trained into classifier (RF or CB)
4	Concatenation Autoencoders (ConcatAE) + Classifier	Using original and encoded features (concatenation) to be trained into classifier (RF or CB)
5	One-vs-the-rest (OvR) multiclass strategy	The one-vs-the-rest (OvR) multiclass strategy, often referred to as one-vs-all, involves training a separate classifier for each class.
6	Self-generated Tasks from unlabeled Tables (STUNT)	A few-shot tabular learning system that utilizes meta-learning to train on self-generated problems derived from unlabeled tables.

known for its comprehensive approach to generate multiple tasks on tabular data setting, served as a rigorous benchmark to evaluate the efficacy of our proposed enhancements. Table 2 shows a more detailed and extensive explanation of these baseline methods.

In the 1-shot learning, we observe varying levels of performance among the baseline models in terms of mean test accuracy. The results in Table 3 illustrate several noteworthy trends in the performance of different classification methods across the datasets examined. First, RF classifier generally outperforms CB classifier in all cases by 0.74% in average. Second, OvR strategy specifically on CB, consistently outperforms the baseline models (RF and CB) on most datasets by 0.99% in average. Additionally, ConcatAE approach surpasses both the OvR strategy by 2.7% in average and the base models by 2.8% in average. These findings suggest that employing advanced techniques such as OvR and ConcatAE can significantly enhance classification accuracy in 1-shot learning scenarios. Compared to standard STUNT, our method, which employs ConcatAE in conjunction with K-medoids clustering and Manhattan ProtoNet, achieved the highest mean test accuracy across all datasets and tasks by 4.03% in average, showcasing the superiority of this approach in 1-shot learning classification.

In the 5-shot learning, the performance patterns observed in Table 4 are similar to those seen in 1-shot settings for various datasets in base models and when combined with augmentation techniques. For instances, RF classifier generally still outperforms CB classifier in all cases by 2.04% in average. Then, OvR strategy specifically on CB, consistently outperforms the baseline models (RF and CB) on most datasets by 2.13% in average. In addition, ConcatAE approach surpasses both the OvR strategy by 0.99% and the base models by 1.09%. These findings suggest that employing advanced techniques such as OvR and ConcatAE can significantly enhance classification accuracy in 5-shot learning scenarios. Compared to standard STUNT, our method, which employs Con-

Table 3: Mean test accuracy on 1-shot setting.

Methods	NPS	Dry Bean	Wine	Cover Type	Average
RF	32.74	68.41	81.39	24.19	51.68
CB	34.01	64.48	84.17	22.54	51.30
AE + RF	34.41	63.14	85.56	22.41	51.38
AE + CB	33.99	61.77	86.53	22.57	51.22
ConcatAE + RF	32.54	68.47	87.64	23.76	53.10
ConcatAE + CB	33.14	66.15	87.92	23.81	52.76
OvR RF	32.31	69.24	81.25	22.46	51.32
OvR CB	33.42	69.54	81.81	22.47	51.81
AE + OvR RF	33.99	60.49	86.94	21.38	50.70
AE + OvR CB	32.91	63.03	86.81	22.02	51.19
ConcatAE + OvR RF	31.77	68.28	87.36	22.39	52.45
ConcatAE + OvR CB	32.40	70.63	87.36	23.43	53.46
STUNT (k-Means + Euclidean ProtoNet)	35.69	67.44	85.75	24.66	53.39
ConcatAE + STUNT	34.47	70.43	87.64	23.76	54.07
ConcatAE + k-Medoid + Manhattan ProtoNet	**36.06**	**71.17**	**88.86**	**26.07**	**55.54**

Table 4: Mean test accuracy on 5-shot setting.

Methods	NPS	Dry Bean	Wine	Cover Type	Average
RF	39.56	84.37	92.50	35.73	63.04
CB	38.51	82.68	88.47	37.44	61.78
AE + RF	39.47	80.62	89.58	31.49	60.29
AE + CB	39.96	79.84	90.97	31.78	60.64
ConcatAE + RF	40.52	84.68	92.92	34.91	63.26
ConcatAE + CB	40.06	83.71	89.86	**38.07**	62.93
OvR RF	39.55	84.54	92.92	33.54	62.64
OvR CB	39.88	85.25	91.81	35.44	63.10
AE + OvR RF	38.69	79.85	90.28	30.52	59.84
AE + OvR CB	39.47	81.69	92.64	31.69	61.37
ConcatAE + OvR RF	40.18	84.66	94.03	33.43	63.08
ConcatAE + OvR CB	40.53	85.53	93.15	35.82	63.91
STUNT (k-Means + Euclidean ProtoNet)	40.76	83.48	94.03	34.72	63.25
ConcatAE + STUNT	40.93	84.15	95.00	31.76	62.96
ConcatAE + k-Medoid + Manhattan ProtoNet	**41.25**	**85.62**	**95.28**	34.58	**64.18**

catAE in conjunction with K-medoids clustering and Manhattan ProtoNet, achieved the highest mean test accuracy in 3 out of 4 datasets — NPS, Dry Bean, and Wine — by 1.47%, showcasing the optimal performance of this approach in 5-shot learning classification.

We also observe some significant result on comparison between scenarios with augmentation — ConcatAE + k-Medoid + Manhattan ProtoNet (our approach)— and no augmentation — RF, CB, OvR RF, and OvR CB. Figure 8 clearly shows that the method with augmentation gives improvement both on 1-shot and 5-shot settings. Specifically, our approach outperforms the traditional ensemble (TE) models and the OvR classifiers by 7.8% in the 1-shot setting and 2.5% in the 5-shot setting. This enhancement underscores the effectiveness of our approach in multiclass classification, demonstrating optimal generalization capabilities compared to models without augmentation.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we propose advancing ProtoNet that employs augmented LF by an autoencoder and MG by STUNT in the few-shot learning mechanism. Specifically, the achieved contributions to this work are threefold. First, we propose an FSL-LFMG framework to develop an end-to-end few-shot multiclass classification workflow on tabular data. This framework is composed of three main stages that include (i) data augmentation at the sample level utilizing autoencoders to generate augmented LF, (ii) data aug-

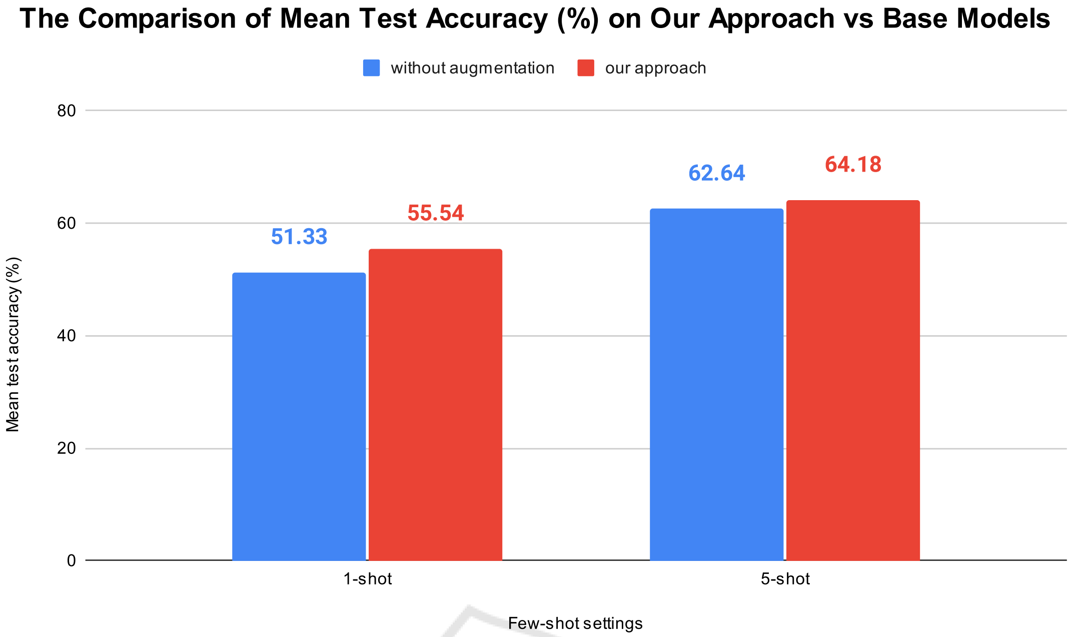


Figure 8: The effect of data augmentation techniques compared to base models.

mentation at the task level involving self-generating multitasks using the STUNT approach, and (iii) the learning process taking place on ProtoNet, followed by various model evaluations in our FSL mechanism. Second, due to the outlier and noise sensitivity of K-means clustering and the curse of dimensionality of Euclidean distance, we enhance and customize the STUNT approach by using K-medoids clustering that is less sensitive to noisy outliers and Manhattan distance that is preferable for high-dimensional data. Finally, we conduct an extensive experimental study on four diverse domain datasets— NPS segmentation, Dry Bean type, Wine type, and Forest Cover type—to prove that our FSL-LFMG approach on the multiclass classification outperforms the TE models and the OvR classifiers by 7.8% in 1-shot and 2.5% in 5-shot learning. Moving forward, we plan to investigate more data augmentation techniques for tabular data including variational autoencoder and generative adversarial network. We also aim to explore more state-of-the-art few-shot learning techniques, such as meta-learning algorithms and advanced metric learning approaches, which have shown the promising results on tabular data in other real-world domains and areas.

REFERENCES

Abdulsalam, S. O., Arowolo, M. O., Saheed, Y. K., and Afolayan, J. O. (2022). Customer churn prediction in telecommunication industry using classification and regression trees and artificial neural network algorithms. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, 10(2):431–440.

Adebiyi, M. O., Ogundokun, R. O., Abokhai, A. A., et al. (2020). Machine learning–based predictive farmland optimization and crop monitoring system. *Scientifica*, 2020.

Aeberhard, S. and Forina, M. (1991). Wine. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5PC7J.

Arik, S. O. and Pfister, T. (2021). TabNet: Attentive Interpretable Tabular Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6679–6687.

Arora, P., Varshney, S., et al. (2016). Analysis of k-means and k-medoids algorithm for big data. *Procedia Computer Science*, 78:507–512.

Attri, I., Awasthi, L. K., and Sharma, T. P. (2024). Machine learning in agriculture: a review of crop management applications. *Multimedia Tools and Applications*, 83(5):12875–12915.

Biau, G. and Scornet, E. (2016). A random forest guided tour. *Test*, 25:197–227.

Blackard, J. (1998). Covertype. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C50K5N.

Blier-Wong, C., Cossette, H., Lamontagne, L., and Marceau, E. (2020). Machine learning in p&c insurance: A review for pricing and reserving. *Risks*, 9(1):4.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., and Kasneci, G. (2022). Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. (2022). Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*.

Katzir, L., Elidan, G., and El-Yaniv, R. (2020). Net-dnf: Effective deep modeling of tabular data. In *International conference on learning representations*.

Khan, M. S., Nath, T. D., Hossain, M. M., Mukherjee, A., Hasnath, H. B., Meem, T. M., and Khan, U. (2023). Comparison of multiclass classification techniques using dry bean dataset. *International Journal of Cognitive Computing in Engineering*, 4:6–20.

Li, W., Wang, Z., Yang, X., Dong, C., Tian, P., Qin, T., Huo, J., Shi, Y., Wang, L., Gao, Y., et al. (2023). Libfewshot: A comprehensive library for few-shot learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Loukili, M., Messaoudi, F., and El Ghazi, M. (2022). Supervised learning algorithms for predicting customer churn with hyperparameter optimization. *International Journal of Advances in Soft Computing & Its Applications*, 14(3).

Matloob, I., Khan, S. A., Hussain, F., Butt, W. H., Rukaiya, R., and Khalique, F. (2021). Need-based and optimized health insurance package using clustering algorithm. *Applied Sciences*, 11(18):8478.

Nam, J., Tack, J., Lee, K., Lee, H., and Shin, J. (2023). Stunt: Few-shot tabular learning with self-generated tasks from unlabeled tables. *arXiv preprint arXiv:2303.00918*.

Parnami, A. and Lee, M. (2022). Learning from few examples: A summary of approaches to few-shot learning. *arXiv preprint arXiv:2203.04291*.

Popov, S., Morozov, S., and Babenko, A. (2019). Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*.

Şahin, C. (2023). Predicting base station return on investment in the telecommunications industry: Machine-learning approaches. *Intelligent Systems in Accounting, Finance and Management*, 30(1):29–40.

Shwartz-Ziv, R. and Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90.

Sikri, A., Jameel, R., Idrees, S. M., and Kaur, H. (2024). Enhancing customer retention in telecom industry with machine learning driven churn prediction. *Scientific Reports*, 14(1):13097.

sklearn (2024). sklearn Documentation. https://scikit-learn.org/0.15/modules/generated/sklearn.multiclass.OneVsRestClassifier.html. Accessed: April 4, 2024.

Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

Sun, B., Yang, L., Zhang, W., Lin, M., Dong, P., Young, C., and Dong, J. (2019). Supertml: Two-dimensional word embedding for the precognition on structured tabular data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0.

Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J. B., and Isola, P. (2020). Rethinking few-shot image classification: a good embedding is all you need? In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 266–282. Springer.

Tunguz, B., Dieter, or Tails, H., Kapoor, K., Pandey, P., Mooney, P., Culliton, P., Mulla, R., Bhutani, S., and Cukierski, W. (2023). 2023 kaggle ai report.

UCI (2020). Dry Bean. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C50S4B.

Wang, R., Pontil, M., and Ciliberto, C. (2021). The role of global labels in few-shot classification and how to infer them. *Advances in Neural Information Processing Systems*, 34:27160–27170.

Wang, Y., Yao, Q., Kwok, J. T., and Ni, L. M. (2020). Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34.

Ye, A. and Wang, Z. (2023). *Modern deep learning for tabular data: novel approaches to common modeling problems*. Springer.

Yu, Z., Wang, K., Xie, S., Zhong, Y., and Lv, Z. (2022). Prototypical network based on manhattan distance. *Cmes-Comput. Model. Eng. Sci*, 131:655–675.

Zhang, R. and Liu, Q. (2023). Learning with few samples in deep learning for image classification, a mini-review. *Frontiers in Computational Neuroscience*, 16:1075294.