Int. J. Appl. Math. Comput. Sci., 2024, Vol. 34, No. 1, 149–165
DOI: 10.61822/amcs-2024-0011

AN EMPIRICAL STUDY OF A SIMPLE INCREMENTAL CLASSIFIER BASED ON VECTOR QUANTIZATION AND ADAPTIVE RESONANCE THEORY

SYLWESTER CZMIL [a,*], JACEK KLUSKA [a], ANNA CZMIL [a]

[a] Department of Control and Computer Engineering
Rzeszow University of Technology
Powstańców Warszawy 12, 35-959 Rzeszów, Poland
e-mail: sylwesterczmil@gmail.com

When constructing a new data classification algorithm, relevant quality indices such as classification accuracy (ACC) or the area under the receiver operating characteristic curve (AUC) should be investigated. End-users of these algorithms are interested in high values of the metrics as well as the proposed algorithm's understandability and transparency. In this paper, a simple evolving vector quantization (SEVQ) algorithm is proposed, which is a novel supervised incremental learning classifier. Algorithms from the family of adaptive resonance theory and learning vector quantization inspired this method. Classifier performance was tested on 36 data sets and compared with 10 traditional and 15 incremental algorithms. SEVQ scored very well, especially among incremental algorithms, and it was found to be the best incremental classifier if the quality criterion is the AUC. The Scott–Knott analysis showed that SEVQ is comparable in performance to traditional algorithms and the leading group of incremental algorithms. The Wilcoxon rank test confirmed the reliability of the obtained results. This article shows that it is possible to obtain outstanding classification quality metrics while keeping the conceptual and computational simplicity of the classification algorithm.

Keywords: incremental learning, data classification, vector quantization, adaptive resonance theory, classification performance.

1. Introduction

Traditional machine learning algorithms do not continually integrate new data into the already trained model. This is a significant problem, especially in the case of the exponentially growing amount of collected data. Increasing the size of the data set leads to a longer training time of the model. Additionally, after generating a new class, the model cannot automatically update it without first updating the model itself. Thus, researchers have been strongly motivated to propose techniques that allow one to update the classification model as new data arrive instead of starting algorithms from scratch. This has resulted in proposals for incremental classifiers (Luo *et al*., 2020). An incremental classifier can identify and update new classes and does not require access to the original data used to train an existing classifier (Polikar *et al*., 2001; Lee *et al*., 2021). It is worth noting that, alongside the shallow methods discussed earlier, deep learning methods have been rapidly developing recently, also in the domain of incremental learning (Pratama *et al*., 2020; Leo and Kalita, 2022). These methods demonstrate effective classification performance, but it often comes with a significant time burden. Theoretical results, especially with respect to the implications of the "no free lunch" theorem (Wolpert and Macready, 1997) and empirical studies, do not exclude the need to create new shallow algorithms (Banerjee *et al*., 2017).

To measure the quality of any new classifier, appropriate metrics, e.g., ACC, AUC, and others, should be defined. End users of these algorithms are interested in high values of these metrics for many different groups of data sets as well as their intelligibility, transparency (Villuendas-Rey *et al*., 2017), and interpretability (Kluska and Madera, 2021). Therefore, decision trees are among the most popular machine learning algorithms (James *et al*., 2013). It is also very important for any newly developed algorithm to compare it with existing ones and show statistically reliable results for a sufficiently large data set (Stapor, 2018).

*Corresponding author

This paper aims to introduce a simple and efficient method of incremental classification that yields high classification quality performance. Category generation on the fly was inspired by vector quantization (VQ) (Kohonen *et al*., 2001) and adaptive resonance theory (ART) (Carpenter *et al*., 1992), but we did not use vigilance to match categories. We chose a way closer to LVQ matching for category matching and compared the closest category with a new record based on label matching. Our algorithm decides whether to create a new category or update an existing one. The main contribution of this work is a novel incremental classifier applied to multiclass classification tasks. We experimentally show that conceptual and computational simplicity can go hand in hand with improving recognized classification quality metrics, which is not an obvious result (Holte, 1993).

The literature review indicates that only a few papers show that straightforward classification algorithms perform well on the most commonly used data sets. This empirical observation is significant for our motivation to set forth a new approach (Fernández-Delgado *et al*., 2014; Kluska and Madera, 2021). Our goal is also to eliminate the drawbacks often found in previous work on vector quantization. First, authors often do not consider the statistical significance of the experimental results. Second, the source code is often unavailable, making it impossible to reproduce the study. Third, algorithms are often published in pseudocode, with their informality risks being overlooked or introducing new bugs when translated into real programming languages. Fourth, algorithms are often tested on a small number of data sets.

The remainder of the paper is organized as follows. In Section 2 we present a short overview of the related work and incremental algorithms that inspired the development of SEVQ. In Section 3, we introduce SEVQ, a simple design for an incrementally trained supervised evolving algorithm for data classification, and briefly discuss the apparent differences between our SEVQ and inspiring algorithms to highlight our contributions to this work. Section 4 concludes with a visualization of our algorithm for sample data sets. Section 5 describes the data sets and provides details on the classifier setup used for comparison with SEVQ. Section 6 outlines the classification assessment methods. Next, we present the experimental results in Section 7. Sections 8 and 9 contain the outcomes of the Scott–Knot analysis and the Wilcoxon signed rank test. Finally, in Section 10, we conclude the article with our observations and directions for future research.

2. Related work

Many classification algorithms have been proposed throughout the history of machine learning, for example, popular support vector machines (SVMs), random forests (RFs), the K-nearest neighbors (KNNs) algorithm, naive Bayes (NB), decision trees (DTs), adaptive boosting (AB), logistic regression (LR) or XGBoost (XGB). It is worth mentioning that XGB has become very popular in recent years and has attracted attention as the algorithm of choice of the winners of data science and machine learning competitions (Shi *et al*., 2019).

Hulten *et al*. (2001) proposed the Hoeffding tree (HT) classifier, an incremental decision tree algorithm capable of learning from data streams in which the input data distribution does not change over time. Bifet and Gavaldà (2009) introduced the Hoeffding adaptive tree (HAT) classifier based on the HT, capable of learning adaptively from data streams that drifted over time. Oza and Russell (2001) proposed the Oza bagging (OB) ensemble learning method, which improved the bagging ensemble method for batching. Gomes *et al*. (2017) introduced an adaptive random forest classifier (ARF). Manapragada *et al*. (2018) proposed an extremely fast decision tree (EFDT) classifier that incrementally constructed a tree. Carpenter and Grossberg (1987) presented adaptive resonance theory (ART). It concerns the brain's ability to quickly learn to categorize, recognize, and predict objects and events in a changing world. ART models used for supervised learning applications typically follow an ARTMAP architecture (Carpenter *et al*., 1991), the first ART supervised predictive mapping model consisting of two binary modules connected via a map field. Fuzzy ARTMAP (FAM) extended the capabilities of ARTMAP to enable the processing of real-valued data by replacing a logical AND operator with a fuzzy AND intersection (Carpenter *et al*., 1992). The simplified fuzzy ARTMAP (SFAM) method was explicitly designed for classification tasks (Kasuba, 1993); in this approach, vectors representing class labels replaced one of the modules. Another simplified architecture was discussed by Vakil-Baghmisheh and Pavešić (2003).

2.1. Family of learning vector quantization (LVQ) algorithms. LVQ was introduced by Kohonen *et al*. (2001) as a supervised classification algorithm based on clustering and classification processes. Class boundaries are defined based on a codebook vector, the winner-takes-it-all paradigm, and a nearest-neighbor rule (Elsayad, 2009; Duch *et al*., 2000). The LVQ family consists of LVQ1, LVQ2, LVQ3, and the improved version of LVQ, such as LVQ2.1, the one-dimensional Kohonen LVQ net or predictive vector quantization (Skubalska-Rafajlowicz, 2000; Rutkowski and Cierniak, 1996). Lughofer (2008b), motivated by the ART network approach, extended the conventional vector quantization (VQ) by incorporating a vigilance parameter and proposed an evolving variant of vector quantization (EVQ). Evolving fuzzy classifiers (eClass), proposed,

e.g., by Pratama *et al*. (2018) or Škrjanc *et al*. (2019), are also relevant algorithms due to their ability to start learning from scratch and do not need to pre-specify the number of fuzzy rules and classes.

2.1.1. Learning vector quantization (LVQ). LVQ is a prototype-based supervised classification algorithm that seeks to adjust the prototypes to obtain a positioning that can correctly discriminate the classes of the training set (Kohonen *et al*., 2001). The algorithm iterates over the set, and each instance verifies the closest prototype using the KNN technique. If they are of the same class, a correction is made in the prototype seeking to approximate the same instance. Otherwise, a separation is carried out. The weight vector update (from the codebook) is as follows:

$$\mathbf{w}_k = \mathbf{w}_k + c\alpha\left(\mathbf{x} - \mathbf{w}_k\right), \tag{1}$$

where $c = 1$ if $\mathbf{x}$ and $\mathbf{w}_k$ belong to the same class, otherwise $c = -1$ and a learning rate α decreases, e.g.,

$$\alpha(i) = \alpha_0\Big(1 - \frac{i}{N_e}\Big), \tag{2}$$

where i is an epoch, N_e is the maximal number of epochs, $\alpha_0 \in (0, 1)$, and $\alpha_0 = 0.4$ by default. Thus, adjustable parameters are α_0 and k (number of neighbours).

2.1.2. Learning vector quantization 2 (LVQ2). LVQ2 is an improved version of LVQ (Kohonen *et al*., 2001). This algorithm iterates on the set and verifies the two closest prototypes using the KNN technique. If the neighboring prototypes are from different classes, a correction is made in the two prototypes, a correction of distance, and another of approximation relative to the instance of the training set. This correction will only be performed if the instance is within a window defined as follows:

$$\min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > \frac{1-s}{1+s}, \tag{3}$$

where the distance $d_k = ||\mathbf{x} - \mathbf{w}_k||$ and s is the size of the 'window' (usually $s \in [0.2, 0.6]$). Suppose $\mathbf{w}_i$ and $\mathbf{w}_j$ are the two closest codebook vectors to $\mathbf{x}$, and $\mathbf{x}$ falls into the 'window.' If $\mathbf{x}$ and $\mathbf{w}_i$ belong to the same class, then $\mathbf{w}_i$ is updated according to (1) for $k = i$ and $c = 1$; if $\mathbf{x}$ and $\mathbf{w}_j$ belong to different classes, then $\mathbf{w}_j$ is updated according to (1) for $k = j$, $c = -1$, where α decreases as in (2). Thus, the adjustable parameters are s—the size of the window, α_0, k, and N_e.

2.1.3. Learning vector quantization 3 (LVQ3). LVQ3 is an improved version of the LVQ2.1 algorithm developed to avoid overfitting in LVQ 2.1. It adjusts the prototypes even when the two closest ones are from the same class as the training pattern (approaching them from the pattern). The weight vector is updated according to (1) for $k \in \{i, j\}$, if $\mathbf{x}$, $\mathbf{w}_i$ and $\mathbf{w}_j$ belong to the same class. However, the positive parameter p now means the ratio between the closest subclasses that trigger double weight update and α decreases as in (2). Thus, adjustable parameters are: s—the size of the 'window', ($s = 0.6$ by default), α_0 (0.01 by default), k—the number of neighbors ($k = 1$ or 3 by default), the parameter $p \in [0.1, 0.5]$, and the number of epochs N_e.

Note that, in LVQ algorithms, input data records need to be normalized.

2.2. Evolving vector quantization (EVQ). EVQ is a supervised version of the original VQ and the LVQ described above. It can evolve new clusters on demand by comparing incoming samples with already generated clusters. The primary differences between EVQ and LVQ approaches are a modified winning selection strategy, which is based on distances from cluster surfaces, rather than from cluster centers, a specific cluster evolution strategy, and a specific classification scheme, which takes into account the relative position of the sample to be classified to the boundary of the influence range of two neighboring clusters (Kohonen *et al*., 2001). Essential for this algorithm is the specific incorporation of the class label in the incremental clustering process to achieve a supervised learning procedure and an evolving clustering-based classifier. Rather than quoting the entire algorithm, we point out three aspects of this algorithm. First, in EVQ, a hit matrix is defined, containing in the i-th row and j-th column the number of samples falling into cluster i and class j. If a new class is introduced by a new sample falling into cluster i, a column is appended whose entries are set to 0, except for the i-th, which is set to 1. Second, the class label is included directly in the feature vectors (if the number of classes is known *a priori*). Third, the so-called vigilance parameter is used, which steers the trade-off between plasticity and stability during incremental online learning (see Lughofer, 2008a, p. 781). The data must be normalized to $[0, 1]$ in this algorithm.

2.3. Simplified fuzzy ARTMAP (SFAM). SFAM is a simpler and faster variant of FAM (Vakil-Baghmisheh and Pavešić, 2003) and is identified with a neural network. Let N be the total number of neurons in the second layer (winner-takes-all) of the SFAM network (N is number of classes). The normalized input of $\mathbf{x} \in \mathbb{R}^M$ we denote by $\mathbf{a} \in [0, 1]^M$, and the neural network input $\mathbf{I} = [\mathbf{a}, 1 - \mathbf{a}]$. The activity level of the r-th neuron we define by $T_r(\mathbf{I}) = (\alpha + |\mathbf{w}_r|)^{-1} |\min(\mathbf{I}, \mathbf{w}_r)|_1$, where the norm $|\cdot|_1$ of $\mathbf{m} = [\mu_1, \ldots, \mu_n] \in [0, 1]^n$ is $|\mathbf{m}|_1 = \mu_1 + \ldots + \mu_n$ (a sigma-count of a fuzzy set). The SFAM algorithm can be briefly described as follows.

S1. Set the vigilance factor equal to its baseline value ρ, compute the nework input $\mathbf{I}$, and calculate the second layer activities (winner-takes-all cells) $T_r(\mathbf{I})$ for $r = 1, \ldots, N-1, T_N = T_0$.

S2. Find the winning neuron as $K = \arg\max_{1 \le r \le N} T_r$. If the winner neuron is uncommitted, go to Step S4. Otherwise, if the input is similar enough to the winner's prototype, i.e.,

$$\frac{|\min\{\mathbf{I}, \mathbf{w}_K\}|_1}{M} \ge \rho \tag{4}$$

is satisfied (resonance), then go to Step S3. If not, reset the winner ($T_K = -1$), go to step S2, and check the next winner.

S3. If the class label of the winner matches the class label of input, update the winner prototype

$$\mathbf{w}_K = \alpha \min\{\mathbf{I}, \mathbf{w}_K\} + \beta \mathbf{w}_K \tag{5}$$

and go to step S5. Otherwise, reset the winner ($T_K = -1$), and

$$\rho = \frac{|\min\{\mathbf{I}, \mathbf{w}_K\}|}{M} + \varepsilon. \tag{6}$$

If $\rho > 1$, (data mismatch), go to Step S5; otherwise, go to Step S2.

S4. Assign the winner neuron $\mathbf{w}_N = \mathbf{I}$ and set the class label of $\mathbf{I}$. Next, create a new uncommitted neuron, and $N = N + 1$.

S5. Go to Step S1 and repeat the algorithm for the next input.

3. Simple evolving vector quantization algorithm

The labeled learning data set is given by

$$D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_z, y_z)\} \subset \mathbb{R}^j \times L, \tag{7}$$

where $L = \{l_1, \ldots, l_r\}$ contains original labels, $2 \le r \le z$ and $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,j})$, $i = 1, \ldots, z$. SEVQ is an incremental learning algorithm that produces the *categories* $c_1, \ldots, c_k$, ($k \le r$), each of which is an 'aggregate' of several records that define their label. For each *category* c_i, there is one assigned *vector of weights* $\mathbf{w}_i \in \mathbb{R}^j$, and there are two scalars n_i and l_i, where n_i is the number of records from set D that are involved in calculating the category, and l_i is a class label to which the category is assigned. The input data can be continuously used to update existing weights. Training can be performed not only record-by-record, but also on a batch of records in several epochs. This algorithm consists of two phases. The first one (the *training* phase) is shown in Algorithm 1, and the second one (the *prediction* phase) in Algorithm 2. The number of epochs N_e is specified by the user, while the outputs are weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_k$, categories' counts of committed samples $n_1, \ldots, n_k$, and categories' labels $l_1, \ldots, l_k$, ($1 \le k \le r$). A "for loop" on the second line in Algorithm 1 signals that the same data should be provided in a different order. The first step is to calculate the Euclidean distances $\|\cdot\|$ between the current data record $\mathbf{x}_i$ and each category weight $\mathbf{w}_1, \ldots, \mathbf{w}_k$, and find the index c of the smallest one. If the category label has the same label as the data record, then the existing category weight is updated by adding the difference between the data record and the previous category weight divided by the number of records that were originally used to train this category. If the labels differ, a new category is created by assigning a data record to it.

During the *prediction* phase, the algorithm searches for the closest category to each tested record and assigns the label of the closest category to it. Due to the *prediction* phase's simplicity, the classification speed is proportional to the number of categories.

Figure 1 presents a matrix of category weights, a vector of category labels, and a vector containing the number of samples committed to creating a given category. As can be seen, the idea of the SEVQ algorithm is straightforward. The clear advantage of SEVQ is that it does not require manual setting of any parameters. Additionally, it does not require data normalization. Users

Algorithm 1. SEVQ training phase.

Inputs:
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_z, y_z)$: a labeled data set,
N_e: the number of epochs
Outputs:
$\mathbf{w}_1, \ldots, \mathbf{w}_k$: weight vectors,
$n_1, \ldots, n_k$: categories' counts of committed samples,
$l_1, \ldots, l_k$: categories' labels

1: $k = 0$
2: **for** $1, 2, \cdots, N_e$, **do**
3: **for** $i = 1, 2, \cdots, z$, **do**
4: find index $c = \arg\min_{s \in \{1,\ldots,k\}} \|\mathbf{x}_i - \mathbf{w}_s\|$,
for this index c, check:
5: **if** $l_c = y_i$ **then**
6: update the existing category weights:
7: $\mathbf{w}_c = \mathbf{w}_c + \frac{1}{n_c}(\mathbf{x}_i - \mathbf{w}_c)$
8: $n_c = n_c + 1$
9: **else**
10: create a new category:
11: $k = k + 1$
12: $\mathbf{w}_k = \mathbf{x}_i$
13: $n_k = 1$
14: $l_k = y_i$
15: **end if**
16: **end for**
17: **end for**

can also set the number of epochs (N_e) during batch learning in the learning phase, but $N_e = 1$ is sufficient for the algorithm to perform well on most data sets. To further improve the results, we propose to increase N_e, but $N_e \leq 10$. Too many epochs can lead to category proliferation. The algorithm is available as an open-source Python package at `https://github.com/sylwekczmil/sevq`. In addition to the algorithm's code, we shared the code used to perform this study so that it is simple to replicate all results in this article. We also ensured that the data sets used to test the performance of the presented algorithms are downloaded automatically.

To clarify the similarities and differences between the newly introduced SEVQ algorithm and the methods that inspired it, we briefly compare it with LVQ, EVQ, and SFAM below.

LVQ has a fixed number of prototypes, whereas SEVQ has an evolving number of categories. Algorithms also have a distinctly different way of decreasing the learning rate during training. Furthermore, SEVQ selects the category to be trained more straightforwardly than LVQ. LVQ requires setting many parameters, determining the appropriate number of prototypes, and initializing the values for the codebook vectors. This task is not trivial due to its sensitivity to initialization (Kohonen *et al.*, 2001).

The main differences between SEVQ and EVQ are that the latter uses the vigilance parameter in the learning phase to decide if the closest cluster should be updated or a new one created. SEVQ makes this decision based on category class comparison: if the closest category has the same class, it will be updated; otherwise, a new category will be created. This difference affects the construction of the model. EVQ uses a hit matrix to represent how many times a cluster has been updated by a sample assigned to a specified class; multiple samples from different classes can update a cluster. In SEVQ, there is no need to use a hit matrix; the category contains only a single-class label because only samples from the same class can update it in the same way as SFAM. There is also a difference in the learning rate: EVQ uses $\alpha = 0.5/n$, and SEVQ employs $\alpha = 1/n$. Having a learning rate equal to $1/n$ ensures that the category weight is exactly at the center of the samples used to update this category.

SEVQ differs from SFAM in terms of updating the prototype weights and the match criterion, since SEVQ is based on distance and SFAM is based on vigilance. Choosing all SFAM parameters, α, β, ρ, and ε, is not trivial in any data set. Such parameters are absent in the SEVQ algorithm.

Finally, we can conclude that SEVQ is considerably different and much more straightforward than all incremental algorithms such as LVQ, LVQ1, LVQ2, LVQ3, EVQ, and SFAM. Therefore, it is crucial to investigate its performance.

Category 1	$w_{1,1}$	$w_{1,2}$	$\cdots$	$w_{1,j}$	l_1	n_1
Category 2	$w_{2,1}$	$w_{2,2}$		$w_{2,j}$	l_2	n_2
$\vdots$	$\cdots$	$\cdots$	$\ddots$	$\cdots$	$\vdots$	$\vdots$
Category k	$w_{k,1}$	$w_{k,2}$	$\cdots$	$w_{k,j}$	l_k	n_k

Fig. 1. Graphical representation of the SEVQ model (k: category index, j: weight vector dimension, $l_1, \ldots, l_k$: labels, $n_1, \ldots, n_k$: numbers of committed samples).

Algorithm 2. SEVQ prediction phase's pseudocode.

Inputs:
$\mathbf{x}_1, \ldots, \mathbf{x}_{z'}$: vectors of data features,
Loading a pretrained model:
$\mathbf{w}_1, \ldots, \mathbf{w}_k$: weight vectors,
$n_1, \ldots, n_k$: categories' counts of committed samples,
$l_1, \ldots, l_k$: categories' labels
Outputs:
$y_1, \ldots, y_{z'}$: data labels
1: **for** $i = 1, 2, \cdots, z'$, **do**
2: find index $c = \arg \min_{s \in \{1,\ldots,k\}} \|\mathbf{x}_i - \mathbf{w}_s\|$
3: $y_i = l_c$
4: **end for**

4. Visualization

The 2D visualizations shown in Figs. 2–5 were prepared to demonstrate the operation of SEVQ. Each shows a plot of records and categories generated for records on the left and the corresponding accuracy plot for the number of records processed by the algorithm on the right to emphasize the incremental nature of this algorithm. Figures were prepared only for visualization purposes, and in these cases, learning and testing were performed on the same data, and the algorithm without epochs was used.

To start our test of the SEVQ algorithm, we randomly generated a multiclass data set that contained 100 data points. Normally distributed clusters of points were assigned to each class. The data set is presented in Fig. 2 on the left. Individual samples are marked on the plot with triangles. We tested SEVQ on this synthetic data set to resolve the problem of classifying samples into one of three classes and observe how many categories the algorithm would create. It generated exactly one category marked with a filled circle for each class. The line plot for accuracy on the right shows that the training process converged well, and the SEVQ algorithm required only five examples to reach 100% accuracy. In the case shown in Fig. 2, one example was randomly selected from the 0th class, three were from the 1st class, and the last was

from the 2nd class. In the optimistic case, if one case were randomly selected from each class, it would certainly turn out that only three examples would suffice to obtain 100% accuracy.

Continuing our test of SEVQ, we generated a new synthetic data set comprised of 100 data points. These points were assembled forming a pair of moons facing each other in an asymmetrical arrangement, as depicted in Fig. 3. Due to their unique arrangement, these moons are not linearly separable. Despite the complexity, SEVQ allocated the categories very well, generating three categories for samples from the 0th class (represented by triangles) and four categories for samples from the 1st class. This experimental scenario was more intricate than the preceding ones, so it required more sample inputs for the SEVQ algorithm to learn accurately. A validation accuracy of 100% was achieved after providing 31 randomly selected samples.

Another type of natural pattern is concentric circles. We generated two concentric circles for the test, each containing 50 data points. These points were assigned to two respective classes, as depicted in Fig. 4. SEVQ created 16 categories for Class 0 and 14 for Class 1 samples. After presenting all the examples to our model, we observed a high accuracy rate of 96%.

To further evaluate our algorithm, we used a two-spiral data set, as shown in Fig. 5. Each spiral contained 100 data points. The two-spiral problem has long served as a standard benchmark for neural network algorithms since its introduction by Wieland (Lang and Witbrock, 1988). Despite its simplicity in visual representation, the extreme nonlinearity of the two-spiral configuration presents a significant challenge for many learning models. Impressively, SEVQ achieved 100% accuracy after processing just 158 examples.

As shown in Figs. 2–5, the order in which the examples are supplied affects the outcome, i.e., providing the samples in a different order would result in the generation of different categories. Experiments show that, for the best results, records should be provided from different classes without class repetition. It is possible to improve the algorithm's results using multiple learning epochs; however, it is recommended to shuffle the records in subsequent epochs.

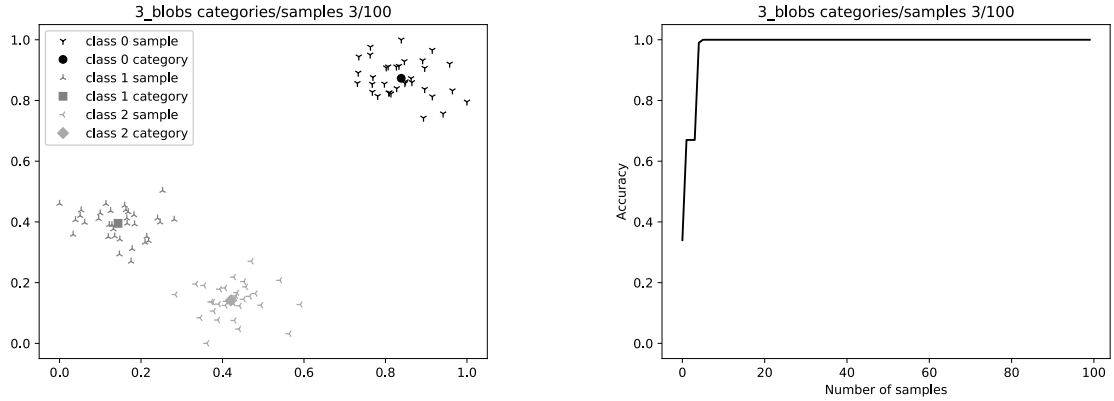
5. Experimental data sets and algorithms

5.1. Data sets. For the experiments, 36 publicly available standard classification data sets with numeric values were selected from the KEEL data set repository (Alcalá-Fdez *et al.*, 2011). To evaluate the SEVQ algorithm reliably, we considered instances, attributes and classes, and data collections from multiple domains during selection. The data sets downloaded from the KEEL data set repository had already been partitioned using a 10-fold stratified cross-validation procedure. Table 1 shows the details of the data sets selected for the experiments, including the name, counts of instances, attributes, and classes (the last ones being the number of possible values of the output variable) and an imbalance score of each data set. Each imbalance score shown in the table represents the average of imbalances calculated for the respective pairs of classes.

5.2. Algorithms. In our study, we consider two groups of classifiers that were compared with the SEVQ algorithm. The first one contains traditional algorithms, most of which were implemented using the scikit-learn library (Pedregosa *et al.*, 2011), and the sole exception—the XGB method—was implemented using the XGBoost library (Chen and Guestrin, 2016).

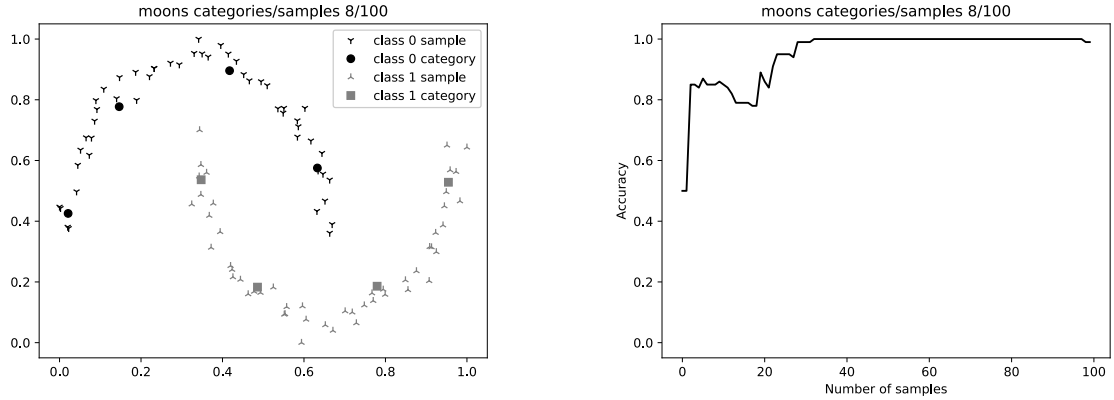
- *AdaBoost (AB)*: here the AdaBoost-SAMME implementation (Hastie *et al.*, 2009).
- *Decision tree (DT)*: the optimized version of the CART algorithm (Breiman *et al.*, 1984; Kusy and Zajdel, 2021).
- *Gaussian NB (GNB)*: the implementation of a Gaussian naive Bayes learning algorithm (Chan *et al.*, 1979).
- *K-nearest neighbors (KNN)*: a nonparametric classification method (Altman, 1992).
- *Multilayer perceptron (MLP)*: a classifier that optimizes the logarithm of the loss function using a stochastic gradient-based optimizer (Kingma and Ba, 2015).
- *Nearest centroid (NC)*: a classification method that assigns observations to the class of training samples with the closest mean or centroid (Tibshirani *et al.*, 2002).
- *Quadratic discriminant analysis (QDA)*: a classifier with a quadratic decision boundary that uses conditional densities of classes and Bayes' rule (Hastie *et al.*, 2008).
- *Random forest (RF)*: a method that combines the output of multiple decision trees to produce a single result (Breiman, 2001).
- *C-support vector classification (SVC)*: an implementation of the SVM based on LibSVM with a radial basis kernel function (Chang and Lin, 2011).
- *XGBoost (XGB)*: a highly scalable decision tree ensemble based on gradient boosting (Friedman, 2001).

The second group contains incremental algorithms implemented using the scikit-multiflow (Montiel *et al.*, 2018) and NeuPy (Shevchuk, 2015) librariers, except for the SFAM algorithm with an implementation based on AIOpenLab's code (Galbraith, 2017) and custom implementation of EVQ (Czmil, 2021).



(a) (b)

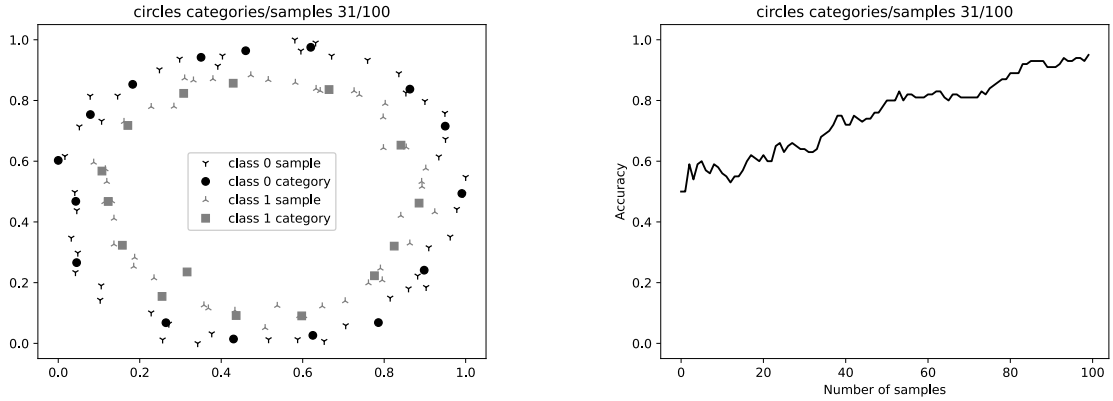
Fig. 2. "Blobs" artificial data set: data visualization before and after the learning process (a), accuracy vs. the number of learning samples (b).



(a) (b)

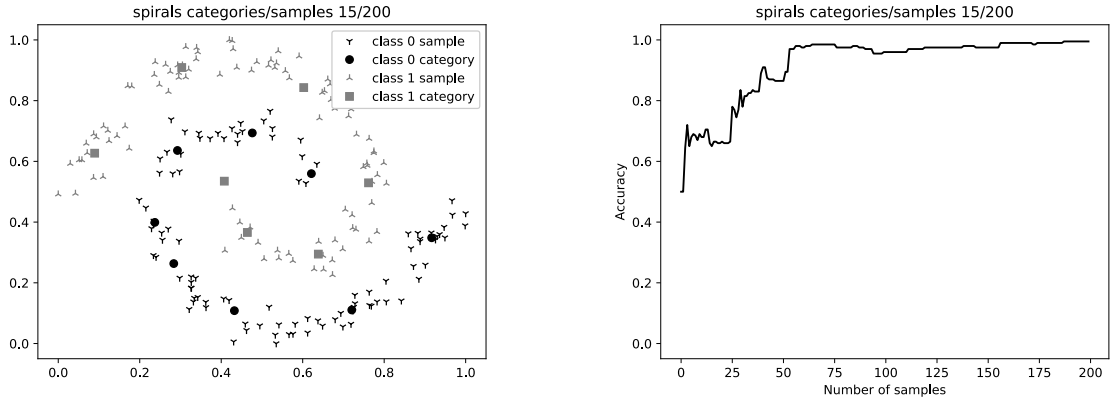
Fig. 3. "Moons" artificial data set: data visualization before and after the learning process (a), accuracy vs. the number of learning samples (b).

- *Additive expert ensemble (AEE)*: a method for using any online learner for drifting concepts (Kolter and Maloof, 2005).
- *Adaptive random forest (ARF)*: an algorithm that adapts the original random forest classifier to the context of evolving data streams (Gomes *et al.*, 2017).
- *Dynamic weighted majority (DWM)*: an algorithm that uses an ensemble of classifiers that are dynamically weighted according to their performance on streaming data with concept drift (Kolter and Maloof, 2007).
- *Extremely fast decision tree (EFDT)*: an incremental decision tree learning algorithm that can revisit and change the split decisions at the nodes to adapt to concept drift and achieve higher accuracy than the Hoeffding tree classifier (Manapragada *et al.*, 2018).
- *Hoeffding adaptive tree (HAT)*: a method that monitors the performance of branches on the tree and replaces them with new items when their accuracy decreases if the new branches are more accurate (Bifet and Gavaldà, 2009).
- *Hoeffding tree (HT)*: an incremental decision tree learning method for stream data that uses a statistical bound to split nodes (Hulten *et al.*, 2001).
- *K-nearest neighbors incremental (KNNI)*: an online version of the KNN classifier that keeps track of the last maximum size of the window storing the last observed samples (Montiel *et al.*, 2018).
- *Naive Bayes (NB)*: a classifier that provides classical Bayesian predictions while naively assuming that all inputs are independent (Montiel *et al.*, 2018; Kulczycki and Kowalski, 2015).
- *Oza bagging (OB)*: an ensemble learning method that improves the bagging ensemble method for the batch setting (Oza and Russell, 2001).
- *Simplified fuzzy ARTMAP (SFAM)*: an algorithm based on AIOpenLab's code (Galbraith, 2017).



(a) (b)

Fig. 4. "Circles" artificial data set: data visualization before and after the learning process (a), accuracy vs. the number of learning samples (b).



(a) (b)

Fig. 5. "Two spirals" artificial data set: data visualization before and after the learning process (a), accuracy vs. the number of learning samples for this artificial data set (b).

- *Evolving vector quantization (EVQ)*: we used an implementation based on the pseudocode EVQ-Class (variant B) included in (Lughofer, 2008a).
- *Learning vector quantization (LVQ, LVQ2, LVQ2.1, LVQ3)*: an implementation and default parameter settings included in (Shevchuk, 2015) were used.

All parameter setups of the compared algorithms are available in the documentation provided at `https://sevq.readthedocs.io/`.

6. Classification assessment methods

For the multi-class classification problem, we used the most popular measures, mainly ACC and AUC. Additionally, *weighted precision (Pre)*, *weighted sensitivity (Sen)*, and the F_1 *measure* were used. Let $L^n = L \times \ldots \times L$ be an n-ary Cartesian product, $\mathbf{t} = [t_1, \ldots, t_n] \in L^n$ the vector containing *true* labels, and $\mathbf{p} = [p_1, \ldots, p_n] \in L^n$ be a vector containing *predicted* labels. For the i-th class $i \in L$, we define the subsets T_i and P_i as follows:

$$T_i = \left\{k \mid \exists\, \mathbf{x}_k \in \mathbb{R}^j,\ (\mathbf{x}_k, i) \in D_{\text{test}},\ i \in \mathbf{t}\right\}, \quad (8)$$

$$P_i = \left\{k \mid \exists\, \mathbf{x}_k \in \mathbb{R}^j,\ (\mathbf{x}_k, i) \in D_{\text{test}},\ i \in \mathbf{p}\right\}, \quad (9)$$

where a test data set $D_{\text{test}} \subset D$. Accuracy is defined as

$$\text{ACC} = \frac{1}{n} \sum_{i \in L} |T_i \cap P_i|, \quad (10)$$

where $|X|$ is the cardinality of the set X. AUC is one of the most important measures of a classifier's performance (Huang and Ling, 2005) and is computed according to the formula

$$\text{AUC} = \frac{1}{r\,(r-1)} \sum_{j=1}^{r} \sum_{\substack{k=1 \\ k \neq j}}^{r} \left(\text{AUC}_{j,k} + \text{AUC}_{k,j}\right), \quad (11)$$

where $\text{AUC}_{p,q}$ is the AUC with class p as the positive class and q as the negative class ($\text{AUC}_{p,q} \neq \text{AUC}_{q,p}$).

Table 1. Data sets used in the experiments.

#	Dataset	Records	Attributes classes	Imbalance score
1	appendicitis	106	7 / 2	0.2471
2	australian	690	14 / 2	0.8016
3	banana	5300	2 / 2	0.8126
4	bupa	345	6 / 2	0.7250
5	cleveland	297	13 / 5	0.4136
6	coil2000	9822	85 / 2	0.0634
7	contraceptive	1473	9 / 3	0.6645
8	dermatology	358	34 / 6	0.5647
9	glass	214	9 / 6	0.4079
10	hayes-roth	160	4 / 3	0.6486
11	heart	270	13 / 2	0.8000
12	hepatitis	80	19 / 2	0.1940
13	led7digit	500	7 / 10	0.8739
14	mammographic	830	5 / 2	0.9438
15	marketing	6876	13 / 9	0.7003
16	monk-2	432	6 / 2	0.8947
17	movement libras	360	90 / 15	1.0000
18	newthyroid	215	5 / 3	0.4302
19	optdigits	5620	64 / 10	0.9858
20	page-blocks	5472	10 / 5	0.2135
21	phoneme	5404	5 / 2	0.4154
22	ring	7400	20 / 2	0.9807
23	satimage	6435	36 / 6	0.6479
24	segment	2310	19 / 7	1.0000
25	shuttle	57999	9 / 7	0.1143
26	spambase	4597	57 / 2	0.6506
27	spectfheart	267	44 / 2	0.2594
28	tae	151	5 / 3	0.9613
29	texture	5500	40 / 11	1.0000
30	thyroid	7200	21 / 3	0.1771
31	titanic	2201	3 / 4	0.5380
32	twonorm	7400	20 / 2	0.9984
33	vowel	990	13 / 11	1.0000
34	wine	178	13 / 3	0.7735
35	wisconsin	683	9 / 2	0.5383
36	zoo	101	16 / 7	0.4364

We compute the average AUC of all possible pairwise combinations of classes.

For two subsets A and B, where $B \neq \emptyset$, we define a function $q(A,B) = |A \cap B| \cdot |B|^{-1}$ and $q(A,B) = 0$ for $B = \emptyset$. *Weighted precision* (Pre, or the *positive predictive value*) and *weighted sensitivity* (Sen, also called *recall*, *hit rate* or the *true positive rate*) are defined by

$$\text{Pre} = \sum_{i \in L} v_i \cdot q(T_i, P_i), \tag{12}$$

$$\text{Sen} = \sum_{i \in L} v_i \cdot q(P_i, T_i), \tag{13}$$

where $v_i = |T_i| / n$ is the weight of the i-th class, ($i \in L$). A *weighted* F_1 *measure* is defined as

$$F_1 = \frac{2}{n} \sum_{i \in L} \frac{|T_i|}{|T_i| + |P_i|} |T_i \cap P_i|. \tag{14}$$

7. Experimental results

In this section, we compare the performance of the new classification algorithm and state-of-the-art traditional algorithms: AB, DT, GNB, KNN, MLP, NC, QDA, RF, SVC, and XGB, and incremental algorithms: SFAM, HT, OB, HAT, KNNI, NB, EFDT, ARF, AEE, DWM, LVQ, LVQ2, LVQ2.1, LVQ3 and EVQ. Results are presented for the traditional and incremental algorithms because the SFAM method and algorithms from the LVQ family

require normalized data. Hence, we used normalized data for testing the incremental algorithms and non-normalized data for testing the traditional ones. The average results obtained for the 36 data sets presented in Table 1 for each incremental algorithm were calculated from the results of each data set.

7.1. Calculated metrics. In this section, we compare the performance of the new classification algorithm and state-of-the-art traditional ones AB, DT, GNB, KNN, MLP, NC, QDA, RF, SVC, and XGB as well as incremental: SFAM, HT, OB, HAT, KNNI, NB, EFDT, ARF, AEE, DWM, LVQ, LVQ2, LVQ2.1, LVQ3, and EVQ. Results are presented in separate tables for the traditional and the incremental algorithms because the SFAM method and algorithms from the LVQ family (with which we primarily wanted to compare the results) require normalized data. Hence, we used normalized data to test incremental algorithms and non-normalized data for traditional ones.

To date, no common consensus has emerged regarding the measures used to evaluate the performance of classifiers to compare data classification algorithms (Sokolova and Lapalme, 2009). Thus, in this study, we chose the most popular measures, such as ACC, precision (Pre), sensitivity (Sen), F_1 score (F_1), and AUC, to evaluate and compare classifiers.

Tables 2 and 3 describe the performance of traditional and incremental classifiers, respectively, calculated as the average classification results obtained for the 36 data sets for each algorithm from each of the fold results on each data set. AUC sorts results in both tables in descending order.

Given such an ordering among the traditional algorithms (Table 2), XGB is the clear winner because it achieved the highest average AUC (0.840±0.149) and other metrics, followed by the MLP with an average AUC of 0.788±0.168. SEVQ is the third-best for AUC (0.787±0.165) and the fifth best for ACC (0.764±0.221). NC was the worst, with an average AUC of 0.716±0.145 and ACC of 0.629±0.201.

Among the incremental algorithms (Table 3), SEVQ achieved the best results for an AUC of 0.794±0.166, followed by NB (0.791±0.144) and EVQ (0.783±0.160). It was second best for ACC (0.775±0.218) and outperformed only by EVQ (0.785±0.193). The worst with average AUC and ACC achieved LVQ3: 0.671±0.164 and 0.658±0.213, respectively.

7.2. Distribution of basic classification metrics. The results of the traditional algorithms and SEVQ are presented using box plots and standardized charts often employed in explanatory data analysis.

Figures 6 and 7 present box plots of accuracy and AUC for each traditional algorithm on 36 data sets subjected to a 10-fold cross-validation. Box plots are arranged in descending order of the medians of accuracy and AUC, respectively. The plots also show several outliers that lower the average results of the algorithm. SEVQ is in the second position among all traditional algorithms for accuracy and AUC; in other words, it is a good algorithm for general usage. It only loses to XGB, which has become the best non-incremental algorithm for winning competitions at Kaggle. Figures 8 and 9 present a box plot of ACC and AUC arranged in descending order. SEVQ is in the first position among all incremental algorithms tested for accuracy and AUC.

7.3. Ranking of the algorithms. AUC and ACC for the 10 folds were calculated for each data set for traditional and incremental algorithms. The algorithm with the highest average value was first. The counts of wins and the instances of second and third are presented in Tables 4 and 5.

Among the traditional algorithms, XGB won most often. SEVQ took third place in this ranking. However, SEVQ scored the highest number of wins due to the highest AUC on six data sets and the highest accuracy on five data sets among the incremental algorithms, surpassing even SFAM, LVQ, LVQ2, LVQ2.1, and LVQ3.

8. Scott–Knott analysis

Additionally, we used the Scott–Knott effect size difference (ESD) test to show the difference in the performance of state-of-the-art classification methods. This test produces the ranking of mean values while ensuring that the magnitude of the difference for all values in each group is negligible and the magnitude of the difference of values between groups is non-negligible (Kluska and Madera, 2021). We used version 2.0.2 of the Scott–Knott ESD test with the default parameter settings (Tantithamthavorn *et al.*, 2019). We performed a Scott–Knot ESD analysis for accuracy. The results are shown in Fig. 10. SEVQ was placed in the third group of traditional algorithms, among algorithms such as the KNN, SVC, and DT. In this analysis, the clear winner is again XGB.

The outcome of the Scott–Knot ESD analysis of algorithm performance in terms of mean AUC is plotted in Fig. 11. In this case, SEVQ performs even better. It is in the second group and its position has improved. XGB remains the best. Next, as in the case of comparing SEVQ with the traditional algorithms, we performed the Scott–Knot ESD analysis for the incremental algorithms.

The outcome of the algorithm performance analysis in terms of mean accuracy (Fig. 12) places SEVQ among

Table 2. Results of a comparison of the traditional algorithms.

	Algorithm	AUC	ACC	Pre	Sen	F_1
1	XGB	0.840±0.149	0.829±0.173	0.831±0.175	0.829±0.173	0.822±0.182
2	MLP	0.788±0.168	0.799±0.181	0.791±0.193	0.799±0.181	0.783±0.196
3	SEVQ	0.787±0.165	0.764±0.221	0.779±0.211	0.764±0.221	0.763±0.222
4	GNB	0.786±0.141	0.703±0.223	0.774±0.189	0.703±0.223	0.699±0.230
5	DT	0.785±0.140	0.757±0.185	0.758±0.191	0.757±0.185	0.743±0.196
6	RF	0.782±0.149	0.793±0.175	0.783±0.187	0.793±0.175	0.775±0.191
7	KNN	0.779±0.170	0.776±0.205	0.779±0.211	0.776±0.205	0.766±0.213
8	AB	0.751±0.150	0.701±0.246	0.681±0.270	0.701±0.246	0.675±0.271
9	SVC	0.746±0.175	0.763±0.202	0.736±0.237	0.763±0.202	0.733±0.228
10	QDA	0.746±0.176	0.675±0.284	0.698±0.287	0.675±0.284	0.655±0.300
11	NC	0.716±0.145	0.629±0.201	0.700±0.194	0.629±0.201	0.632±0.201

Table 3. Results of a comparison of the incremental algorithms.

	Algorithm	AUC	ACC	Pre	Sen	F_1
1	SEVQ	0.794±0.166	0.775±0.218	0.789±0.207	0.775±0.218	0.774±0.219
2	NB	0.791±0.144	0.756±0.192	0.769±0.199	0.756±0.192	0.747±0.205
3	EVQ	0.783±0.160	0.785±0.193	0.791±0.193	0.785±0.193	0.778±0.198
4	HAT	0.780±0.159	0.761±0.193	0.750±0.217	0.761±0.193	0.741±0.213
5	SFAM	0.779±0.155	0.759±0.204	0.779±0.196	0.759±0.204	0.757±0.205
6	HT	0.775±0.159	0.762±0.192	0.748±0.221	0.762±0.192	0.741±0.213
7	AEE	0.765±0.149	0.718±0.218	0.723±0.233	0.718±0.218	0.701±0.234
8	OB	0.762±0.169	0.763±0.217	0.759±0.233	0.763±0.217	0.745±0.235
9	KNNI	0.760±0.168	0.759±0.217	0.756±0.233	0.759±0.217	0.742±0.234
10	EFDT	0.757±0.157	0.736±0.212	0.733±0.233	0.736±0.212	0.713±0.232
11	ARF	0.753±0.165	0.717±0.242	0.683±0.285	0.717±0.242	0.681±0.279
12	LVQ2	0.750±0.160	0.732±0.223	0.726±0.237	0.732±0.223	0.715±0.240
13	LVQ	0.738±0.166	0.725±0.229	0.710±0.254	0.725±0.229	0.699±0.255
14	LVQ2.1	0.736±0.167	0.729±0.227	0.707±0.266	0.729±0.227	0.699±0.260
15	DWM	0.735±0.160	0.674±0.262	0.675±0.288	0.674±0.262	0.656±0.283
16	LVQ3	0.671±0.164	0.658±0.213	0.649±0.226	0.658±0.213	0.626±0.223

the best and well-known competitors, namely EVQ, OB, HT, HAT, KNNI, SFAM, and NB. Furthermore, the outcome of the Scott–Knot ESD analysis of algorithm performance in terms of mean AUC (Fig. 13) indicates that SEVQ's performance ranked first among the top six algorithms.

9. Wilcoxon's signed-rank test

To compare statistically multiple algorithms on multiple data sets, we use the Wilcoxon signed rank test, which does not assume a data distribution (Trawiński *et al.*, 2012). This nonparametric statistical hypothesis test determines whether there is a statistically significant difference between classifiers (Kluska and Madera, 2021). The result of this test depends only on the two algorithms being compared. The following cases are considered: SEVQ vs. the traditional algorithms for the ACC and AUC measures, and SEVQ vs. the incremental algorithms for ACC and AUC measures. Let X be a vector containing elements that are mean values of the ACC (or AUC) measure for the SEVQ algorithm tested on 10 random stratified folds for each data set, and Y_i a vector containing the corresponding values for the i-th algorithm tested on exactly the same folds. The index i corresponds to the analyzed algorithm. Table 6 shows the probability (p-value) of a paired two-sided Wilcoxon test for the null hypothesis H_0 that the difference $(X - Y_i)$ comes from a distribution with zero median. The two-sided p-value is computed by doubling the most significant one-sided value.

The results in Table 6 indicate that the test fails to reject the null hypothesis of no significant difference in the mean values of ACC at the significance level of $\alpha = 0.05$ for SEVQ and three algorithms: AB, DT and SVC. The alternative hypothesis H_1 is accepted: there is a significant difference in the mean values of ACC for SEVQ compared with the GNB, KNN, MLP, NC, QDA, RF and XGB algorithms. In addition, according to the distribution of accuracy values in Fig. 6, the SEVQ

Table 4. Ranking of the compared traditional algorithms.

#	Alg.	AUC 1st	AUC 2nd	AUC 3rd	ACC 1st	ACC 2nd	ACC 3rd
1	XGB	7	9	3	8	8	4
2	AB	6	0	5	6	1	3
3	SEVQ	4	6	1	4	3	2
4	GNB	4	4	3	4	0	1
5	SVC	4	1	5	6	0	7
6	NC	4	1	0	1	1	0
7	MLP	3	4	8	4	5	9
8	QDA	2	2	4	1	2	5
9	KNN	1	4	2	1	4	1
10	RF	1	2	4	1	6	3
11	DT	0	3	1	0	6	1

Table 5. Ranking of the compared incremental algorithms.

#	Alg.	AUC 1st	AUC 2nd	AUC 3rd	ACC 1st	ACC 2nd	ACC 3rd
1	SEVQ	6	6	2	5	6	2
2	EVQ	4	6	2	6	3	3
3	AEE	4	5	1	5	1	1
4	SFAM	4	3	5	3	4	5
5	ARF	4	1	2	7	1	2
6	DWM	3	2	3	3	1	2
7	HAT	3	2	3	1	0	4
8	OB	2	3	4	2	4	4
9	EFDT	2	3	4	1	7	1
10	NB	1	2	2	0	2	1
11	KNNI	1	1	3	0	2	6
12	LVQ2.1	1	1	0	2	1	0
13	LVQ2	1	0	2	0	1	2
14	LVQ	0	1	1	1	2	0
15	HT	0	0	2	0	1	2
16	LVQ3	0	0	0	0	0	1

algorithm is worse than XGB, but SEVQ is better than the following six algorithms: RF, MLP, KNN, QDA, GNB and NC. Furthermore, according to the Scott–Knott ranking shown in Fig. 10, Wilcoxon's test result confirms, at the significance level of $\alpha = 0.05$ the placement of SEVQ in the third group of the best traditional algorithms, just after the KNN algorithm and better than GNB, QDA and NC, from the perspective of the ACC measure. Furthermore, Wilcoxon's test confirms, at the significance level of $\alpha = 0.05$, the Scott–Knott ranking result shown in Fig. 11, i.e., SEVQ is worse than XGB but better than the QDA, AB, SVC and NC algorithms.

Taking into account the p-values for the ACC measure in Table 7 and the distribution of ACC values in Fig. 8, we conclude that SEVQ is superior at the significance level of $\alpha = 0.05$ to the SFAM, LVQ2.1, LVQ2, ARF, NB, LVQ, EFDT, AEE, DWM and LVQ3 algorithms. Wilcoxon's test confirms, at the same significance level, the Scott–Knott ranking result shown in Fig. 12.

Finally, we conclude that SEVQ is superior at the significance level of $\alpha = 0.05$ to the HT, SFAM, EVQ, ARF, AEE, OB, KNNI, LVQ2, LVQ, EFTD, LVQ2.1, DWM and LVQ3 algorithms, and Wilcoxon's test confirms, at the same significance level, the Scott–Knott ranking result shown in Fig. 13.

10. Conclusions

This paper introduced the SEVQ method, a supervised incremental learning classifier. The proposed algorithm draws inspiration from the ART and LVQ algorithm families. It merges the capacity to incrementally generate new categories, as is done in SFAM and EVQ, with the critical difference being the elimination of the traditional vigilance parameter. SEVQ decides to update the nearest cluster or create a new one based on category class comparison.

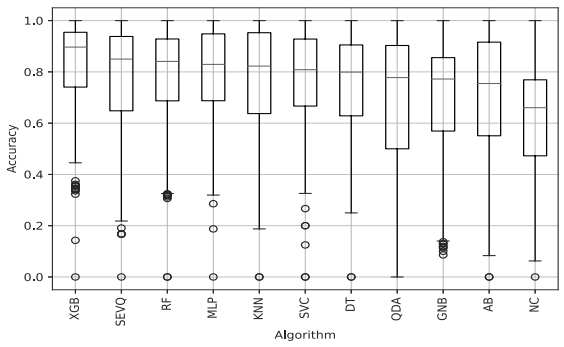


Fig. 6. ACC values for the traditional algorithms across data sets.

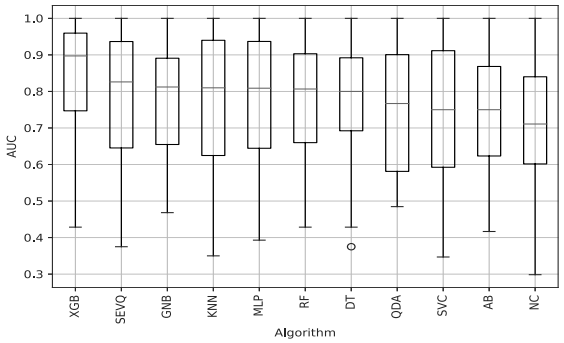


Fig. 7. AUC values for the traditional algorithms across data sets.

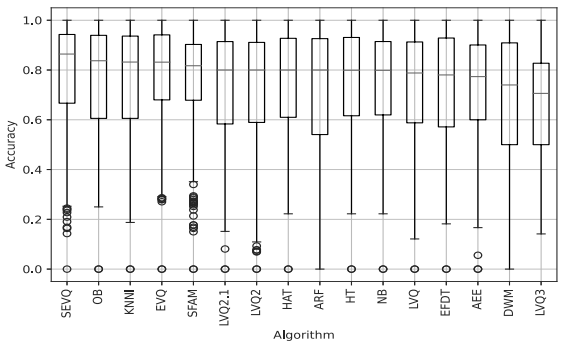


Fig. 8. ACC values for the incremental algorithms across data sets.

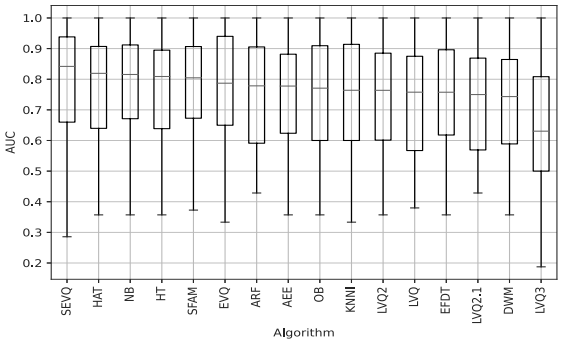


Fig. 9. AUC values for the incremental algorithms across data sets.

We thoroughly study the performance of the new classifier and state-of-the-art algorithms. We calculated the most popular measures, such as accuracy and AUC, to evaluate and compare classifiers. SEVQ was tested on 36 data sets, and its effectiveness was compared with 25 other incremental and traditional methods (non-incremental). When evaluated against the traditional machine learning algorithms, SEVQ ranked second in accuracy and AUC medians. Compared with incremental machine learning algorithms, SEVQ emerged as the best regarding these same performance metrics.

We ranked the algorithms based on the accuracy and AUC metrics, utilizing the calculated number of wins as well as second and third places across all data sets. On this basis, we assigned ranks. Among the traditional algorithms, SEVQ placed third, but it was the top performer in the incremental algorithms category, achieving the best results on the most significant number of data sets.

We also provide the results of relevant statistical tests. According to the Wilcoxon rank test results and the distribution of ACC values for each traditional algorithm across all data sets, the SEVQ algorithm underperformed, at the significance level of $\alpha = 0.05$, the XGB algorithm. However, it outperformed the RF, MLP, KNN, QDA, GNB, and NC algorithms. A review of p-values for the AUC measure and the distribution of AUC values for each traditional algorithm across all data sets shows that SEVQ underperformed XGB but outperformed the QDA, AB, SVC, and NC algorithms. The Wilcoxon test results confirmed the Scott–Knott analysis results because SEVQ was placed in the third group of the best traditional algorithms according to the ACC measure and in the second group of the best traditional algorithms according to the AUC measure.

Taking into account the p-values for the ACC measure and the distribution of ACC values per incremental algorithm across all data sets, we conclude that SEVQ is superior, at the significance level of $\alpha = 0.05$, to the SFAM, LVQ2.1, LVQ2, ARF, NB, LVQ, EFDT, AEE, DWM, and LVQ3 algorithms. Wilcoxon's test confirms the Scott–Knott ranking result that SEVQ is as good as EVQ and is better than the HT, SFAM, EVQ, ARF, AEE, OB, KNNI, LVQ2, LVQ, EFTD, LVQ2.1, DWM, and LVQ3 algorithms under the same criteria as above. The results of Wilcoxon's rank test and the distribution of AUC values for each incremental algorithm across all data sets in terms of the ACC criterion indicate that SEVQ is superior, at the significance level of $\alpha = 0.05$, to the HT, SFAM, EVQ, ARF, AEE, OB, KNNI, LVQ2, LVQ, EFTD, LVQ2.1, DWM, and LVQ3 algorithms. The results of Wilcoxon's rank test confirm

Table 6. Comparison of the traditional classifiers and SEVQ with Wilcoxon's signed-rank test.

#	Algorithm	ACC p-value	AUC p-value
1	AB	**0.0563**	0.0144
2	DT	**0.9889**	**0.5877**
3	GNB	0.0001	**0.5384**
4	KNN	0.0002	**0.6423**
5	MLP	0.0000	**0.3201**
6	NC	0.0000	0.0000
7	QDA	0.0008	0.0012
8	RF	0.0001	**0.2342**
9	SVC	**0.2323**	0.0000
10	XGB	0.0000	0.0000

Table 7. Comparison of the incremental classifiers and SEVQ with Wilcoxon's signed-rank test.

#	Algorithm	ACC p-value	AUC p-value
1	AEE	0.0001	0.0013
2	ARF	0.0374	0.0001
3	DWM	0.0000	0.0000
4	EFDT	0.0199	0.0000
5	EVQ	**0.3275**	0.0209
6	HAT	**0.1737**	**0.0587**
7	HT	**0.2214**	0.0120
8	KNNI	**0.9419**	0.0003
9	LVQ	0.0000	0.0000
10	LVQ2	0.0001	0.0000
11	LVQ2.1	0.0000	0.0000
12	LVQ3	0.0000	0.0000
13	NB	0.0189	**0.5311**
14	OB	**0.3693**	0.0025
15	SFAM	0.0000	0.0000

Scott–Knott's ranking. Hence, we conclude that SEVQ achieves outstanding results in the group of incremental algorithms.

The experimental results demonstrate the effectiveness of SEVQ. They show that a very efficient supervised incremental classifier does not have to be complicated and can achieve a very high classification performance.

The algorithm developed has numerous advantages over other existing methods, stated below.

- SEVQ is conceptually very simple and suitable for incremental learning and multiclass classification tasks.
- It is less complicated than many incremental and non-incremental classifiers and, at the same time, achieves better classification indices for many data sets.
- It does not require manual setting of any parameters.
- Unlike algorithms from the LVQ and ART families, the SEVQ algorithm can be applied to unnormalized data. Our research on 36 data sets shows a slight effect of data normalization on the obtained results. The mean difference of the results achieved by SEVQ for normalized and non-normalized data in terms of ACC was 1.1%, and in terms of AUC it was 0.7%.
- The algorithm has been well tested, and the results are statistically reliable, as shown in the article.
- The algorithm code in Python is publicly available, and all results are easily reproducible.

The findings of this study have to be seen in the light of some limitations, outlined as follows.

- First, we did not apply quality indicators dedicated to the stream data classification algorithms intended for real-time classifiers.
- The second limitation concerns not taking into account concept drift.
- We also did not perform a memory requirement test.

The ideas and results presented in this paper contributed to creating the Classification Algorithms Comparison Pipeline (CACP) to compare newly introduced classification algorithms developed in Python with other commonly used classifiers to evaluate classification performance, reproducibility, and statistical reliability (Czmil *et al.*, 2022). In its current version, the SEVQ algorithm can be applied to industrial process diagnostics (Żabiński *et al.*, 2014; 2017), and we are confident that it can be a competitive algorithm for big data reduction. Future work aims to conduct additional research on the SEVQ algorithm in the context of concept drift and data reduction in big data environments using this algorithm. Due to its simplicity, SEVQ is a suitable algorithm for implementation on a field-programmable gate array (FPGA) for industrial applications. We have made some progress with this experiment up to now.

Acknowledgment

This work was partially supported by the Rzeszow University of Technology within the resources for maintaining the scientific research and development potential (UPB).

References

Alcalá-Fdez, J., Fernandez, A., Luengo, J. , Derrac, J. , García, S., Sanchez, L. and Herrera, F. (2011). KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, *Journal of Multiple-Valued Logic and Soft Computing* **17**(2–3): 255–287.

Altman, N.S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression, *The American Statistician* **46**(3): 175–185.

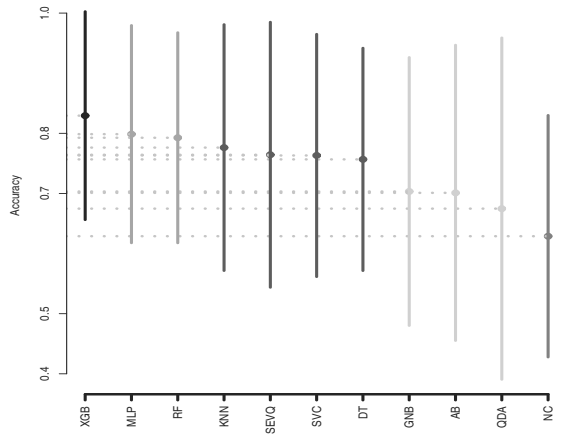


Fig. 10. Scott–Knott analysis for average ACC of the traditional algorithms over all data sets.

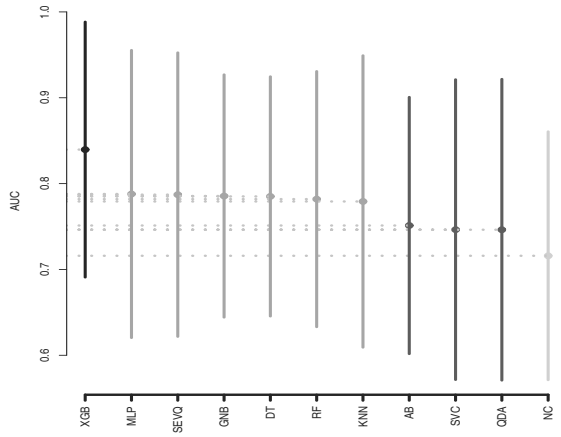


Fig. 11. Scott–Knott analysis for average AUC of the traditional algorithms over all data sets.

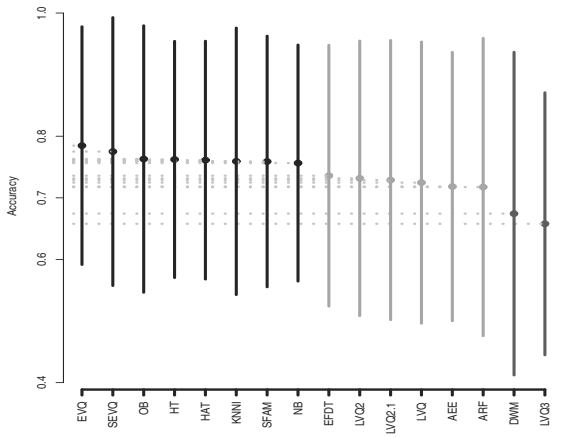


Fig. 12. Scott–Knott analysis for average ACC of the incremental algorithms over all data sets.

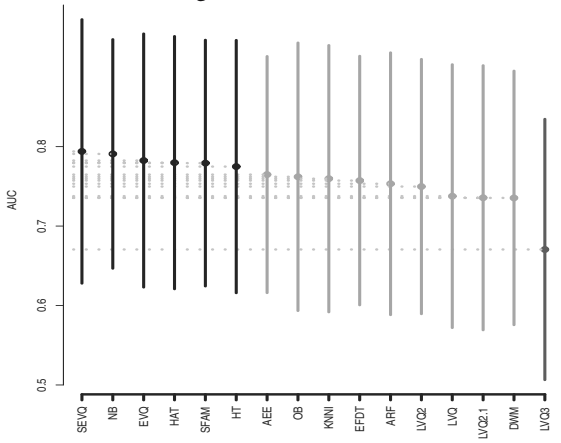


Fig. 13. Scott–Knott analysis for average AUC of the incremental algorithms over all data sets.

Banerjee, S., Bhattacharjee, P. and Das, S. (2017). Performance of deep learning algorithms vs. shallow models, in extreme conditions—Some empirical studies, *in* B.U. Shankar *et al.* (Eds), *Pattern Recognition and Machine Intelligence*, Springer International Publishing, Cham, pp. 565–574.

Bifet, A. and Gavaldà, R. (2009). Adaptive learning from evolving data streams, *in* N.M Adams *et al.* (Eds), *Advances in Intelligent Data Analysis VIII*, Springer, Berlin/Heidelberg, pp. 249–260.

Breiman, L. (2001). Random forests, *Machine Learning* **45**(1): 5–32.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984). *Classification and Regression Trees*, Wadsworth International Group, Belmont.

Carpenter, G.A. and Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine, *Computer Vision, Graphics, and Image Processing* **37**(1): 54–115.

Carpenter, G.A., Grossberg, S. and Reynolds, J.H. (1991). ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network, *Neural Networks* **4**(5): 565–588.

Carpenter, G., Grossberg, S., Markuzon, N. , Reynolds, J.H. and Rosen, D.B. (1992). Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps, *IEEE Transactions on Neural Networks* **3**(5): 698–713.

Chan, T.F., Golub, G.H. and LeVeque, R.J. (1979). Updating formulae and an pairwise algorithm for computing sample variances, *Stanford Working Paper STAN-CS-79-773*: 1–22, Stanford University, Stanford, `http://i.stanford.edu/pub/cstr/reports/cs/tr/79/773/CS-TR-79-773.pdf`.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machine, *ACM Transactions on Intelligent Systems and Technology* **2**(3): 1–27.

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, USA*, pp. 785–794.

Czmil, S. (2021). Python implementation of evolving vector quantization for classification of on-line data streams (Version 0.0.2), Computer software, https://github.com/sylwekczmil/evq.

Czmil, S., Kluska, J. and Czmil, A. (2022). CACP: Classification algorithms comparison pipeline, *SoftwareX* **19**: 101134.

Duch, W., Adamczak, R. and Diercksen, G.H.F. (2000). Classification, association and pattern completion using neural similarity based methods, *International Journal of Applied Mathematics and Computer Science* **10**(4): 747–766.

Elsayad, A.M. (2009). Classification of ECG arrhythmia using learning vector quantization neural networks, *2009 International Conference on Computer Engineering & Systems, Cairo, Egypt*, pp. 139–144.

Fernández-Delgado, M., Cernadas, E., Barro, S. and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems?, *Journal of Machine Learning Research* **15**(90): 3133–3181.

Friedman, J.H. (2001). Greedy function approximation: A gradient boosting machine, *The Annals of Statistics* **29**(5): 1189–1232.

Galbraith, B. (2017). Adaptive resonance theory models, Computer software, https://github.com/AIOpenLab/art.

Gomes, H.M., Bifet, A., Read, J., Barddal, J.P., Enembreck, F., Pfharinger, B., Holmes, G. and Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification, *Machine Learning* **106**(9–10): 1469–1495.

Hastie, T., Rosset, S., Zhu, J. and Zou, H. (2009). Multi-class AdaBoost, *Statistics and Its Interface* **2**(3): 349–360.

Hastie, T., Tibshirani, R. and Friedman, J. (2008). *The Elements of Statistical Learning*, Springer, New York.

Holte, R.C. (1993). Very simple classification rules perform well on most commonly used data sets, *Machine Learning* **11**(1): 63–90.

Huang, J. and Ling, C.X. (2005). Using AUC and accuracy in evaluating learning algorithms, *IEEE Transactions on Knowledge and Data Engineering* **17**(3): 299–310.

Hulten, G., Spencer, L. and Domingos, P. (2001). Mining time-changing data streams, *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'01, San Francisco, USA*, pp. 97–106.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). *An Introduction to Statistical Learning: With Applications in R*, Springer, New York.

Kasuba, T. (1993). Simplified fuzzy ARTMAP, *AI Expert* **8**: 18–25.

Kingma, D.P. and Ba, J. (2015). Adam: A method for stochastic optimization, *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, USA*.

Kluska, J. and Madera, M. (2021). Extremely simple classifier based on fuzzy logic and gene expression programming, *Information Sciences* **571**: 560–579.

Kohonen, T., Schroeder, M.R. and Huang, T.S. (2001). *Self-Organizing Maps*, 3rd Edn, Springer-Verlag, Berlin/Heidelberg.

Kolter, J.Z. and Maloof, M.A. (2005). Using additive expert ensembles to cope with concept drift, *Proceedings of the 22nd International Conference on Machine Learning (ICML-2005), Bonn, Germany*, pp. 449–456.

Kolter, J.Z. and Maloof, M.A. (2007). Dynamic weighted majority: An ensemble method for drifting concepts, *Journal of Machine Learning Research* **8**(91): 2755–2790.

Kulczycki, P. and Kowalski, P.A. (2015). Bayes classification for nonstationary patterns, *International Journal of Computational Methods* **12**(02): 1550008.

Kusy, M. and Zajdel, R. (2021). A weighted wrapper approach to feature selection, *International Journal of Applied Mathematics and Computer Science* **31**(4): 685–696, DOI: 10.34768/amcs-2021-0047.

Lang, K.J. and Witbrock, M.J. (1988). Learning to tell two spirals apart, *The 1988 Connectionist Models Summer School, Pittsburgh, USA*, pp. 52–59.

Lee, S., Chang, K. and Baek, J.-G. (2021). Incremental learning using generative-rehearsal strategy for fault detection and classification, *Expert Systems with Applications* **184**: 115477.

Leo, J. and Kalita, J. (2022). Incremental deep neural network learning using classification confidence thresholding, *IEEE Transactions on Neural Networks and Learning Systems* **33**(12): 7706–7716.

Lughofer, E. (2008a). Evolving vector quantization for classification of on-line data streams, *2008 International Conference on Computational Intelligence for Modelling Control & Automation (CIMCA 2008), Vienna, Austria*, pp. 779–784.

Lughofer, E. (2008b). Extensions of vector quantization for incremental clustering, *Pattern Recognition* **41**(3): 995–1011.

Luo, Y., Yin, L., Bai, W. and Mao, K. (2020). An appraisal of incremental learning methods, *Entropy* **22**(11): 1190.

Manapragada, C., Webb, G.I. and Salehi, M. (2018). Extremely fast decision tree, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK*, pp. 1953—-1962.

Montiel, J., Read, J., Bifet, A. and Abdessalem, T. (2018). Scikit-Multiflow: A multi-output streaming framework, *Journal of Machine Learning Research* **19**(72): 1–5.

Oza, N.C. and Russell, S.J. (2001). Online bagging and boosting, *in* T.S. Richardson and T.S. Jaakkola (Eds), *Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics, Key West, USA*, pp. 229–236.

Pedregosa, F., Varoquaux, G., Gramfort, A. Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* **12**: 2825–2830.

Polikar, R., Upda, L., Upda, S. and Honavar, V. (2001). Learn++: An incremental learning algorithm for supervised neural networks, *IEEE Transactions on Systems, Man, and Cybernetics C: Applications and Reviews* **31**(4): 497–508.

Pratama, M., Pedrycz, W. and Lughofer, E. (2018). Evolving ensemble fuzzy classifier, *IEEE Transactions on Fuzzy Systems* **26**(5): 2552–2567.

Pratama, M., Pedrycz, W. and Webb, G.I. (2020). An incremental construction of deep neuro fuzzy system for continual learning of nonstationary data streams, *IEEE Transactions on Fuzzy Systems* **28**(7): 1315–1328.

Rutkowski, L. and Cierniak, R. (1996). Image compression by competitive learning neural networks and predictive vector quantization, *Applied Mathematics and Computer Science* **6**(3): 431–445.

Shevchuk, Y. (2015). NeuPy (Version 1.18.5), Computer software, `http://neupy.com/`.

Shi, X., Wong, Y.D., Li, M.Z.-F., Palanisamy, C. and Chai, C. (2019). A feature learning approach based on XGBoost for driving assessment and risk prediction, *Accident Analysis & Prevention* **129**: 170–179.

Skubalska-Rafajlowicz, E. (2000). One-dimensional Kohonen LVQ nets for multidimensional pattern recognition, *International Journal of Applied Mathematics and Computer Science* **10**(4): 767–778.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks, *Information Processing & Management* **45**(4): 427–437.

Stapor, K. (2018). Evaluating and comparing classifiers: Review, some recommendations and limitations, *in* M. Kurzynski *et al.* (Eds), *Proceedings of the 10th International Conference on Computer Recognition Systems, CORES 2017*, Springer International Publishing, Cham, pp. 12–21.

Tantithamthavorn, C., McIntosh, S., Hassan, A.E. and Matsumoto, K. (2019). The impact of automated parameter optimization on defect prediction models, *IEEE Transactions on Software Engineering* **45**(7): 683–711.

Tibshirani, R., Hastie, T., Narasimhan, B. and Chu, G. (2002). Diagnosis of multiple cancer types by shrunken centroids of gene expression, *Proceedings of the National Academy of Sciences* **99**(10): 6567–6572.

Trawiński, B., Smętek, M., Telec, Z. and Lasota, T. (2012). Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms, *International Journal of Applied Mathematics and Computer Science* **22**(4): 867–881, DOI: 10.2478/v10006-012-0064-z.

Vakil-Baghmisheh, M. and Pavešić, N. (2003). A fast simplified fuzzy ARTMAP network, *Neural Processing Letters* **17**(3): 273–316.

Villuendas-Rey, Y., Rey-Benguría, C.F., Ángel Ferreira-Santiago, Camacho-Nieto, O. and Yáñez-Márquez, C. (2017). The naïve associative classifier (NAC): A novel, simple, transparent, and accurate classification model evaluated on financial data, *Neurocomputing* **265**: 105–115.

Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation* **1**(1): 67–82.

Żabiński, T., Maczka, T. and Kluska, J. (2017). Industrial platform for rapid prototyping of intelligent diagnostic systems, *in* W. Mitkowski *et al.* (Eds), *Trends in Advanced Intelligent Control, Optimization and Automation*, Springer International Publishing, Cham, pp. 712–721.

Żabiński, T., Maczka, T., Kluska, J., Kusy, M., Hajduk, Z. and Prucnal, S. (2014). Failures prediction in the cold forging process using machine learning methods, *in* L. Rutkowski *et al.* (Eds), *Artificial Intelligence and Soft Computing*, Springer International Publishing, Cham, pp. 622–633.

Škrjanc, I., Iglesias, J.A., Sanchis, A., Leite, D., Lughofer, E. and Gomide, F. (2019). Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A survey, *Information Sciences* **490**: 344–368.

Sylwester Czmil received his MSc (Eng) degree from the Rzeszow University of Technology, Poland, in 2018. His research interests include machine learning methods, artificial intelligence, incremental learning, algorithm performance evaluation, and practical applications. He is a PhD student at the Rzeszow University of Technology and a member of the Polish Information Processing Society.

Jacek Kluska received his MSc (Eng) and PhD degrees from the Wroclaw University of Technology, Poland, in 1977 and 1983, respectively. In 1995 he became an associate professor, and in 2010 a full professor at the Rzeszow University of Technology. His research interests include machine learning methods, artificial intelligence, fuzzy modeling and control, and fuzzy Petri nets. Professor Kluska is a member of the Committee on Automatic Control and Robotics of the Polish Academy of Sciences.

Anna Czmil obtained her MSc (Eng) degree from the Rzeszow University of Technology, Poland, in 2018. Her research interests include machine learning methods, artificial intelligence, and their applications in medicine. She is a PhD student at the Rzeszow University of Technology and a member of the Polish Information Processing Society.

Received: 1 June 2023
Revised: 26 July 2023
Accepted: 24 September 2023