Engineering Applications of Artificial Intelligence 152 (2025) 110772

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Research paper

Pulling the carpet below the learner's feet: Genetic algorithm to learn ensemble machine learning model during concept drift

Teddy Lazebnik *

Department of Mathematics, Ariel University, Ariel, Israel
Department of Cancer Biology, Cancer Institute, University College London, London, UK

ARTICLE INFO

Keywords:
Automatic machine learning
Heuristic optimization
Concept drift
Ensemble machine learning

ABSTRACT

Data-driven models, in general, and machine learning (ML) models, in particular, have gained popularity over recent years with an increased usage of such models across the scientific and engineering domains. When using ML models in realistic and dynamic environments, users often need to handle the challenge of concept drift (CD). In this study, we explore the application of genetic algorithms (GAs) to address the challenges posed by CD in such settings. Formally, we propose a novel two-level ensemble ML model, which combines a global ML model with a CD detector, operating as an aggregator for a population of ML pipeline models, each one with an adjusted CD detector by itself responsible for re-training its ML model. In addition, we show that one can further improve the proposed model by utilizing off-the-shelf automatic ML (AutoML) methods. Through extensive synthetic dataset analysis, we show that the proposed model statistically significantly outperforms an ML pipeline with a CD algorithm, particularly in scenarios with unknown CD characteristics or a mixture of moving and shifting CDs. Moreover, we show a sub-linear decline in the proposed method's performance with respect to a higher drifting rate and robustness to the underlying AutoML method utilized.

1. Introduction

Data-driven models, in general, and machine learning (ML) models, in particular, have gained popularity over recent years with increased usage of such models across the scientific and engineering domains (Lazebnik, 2023; Çubukçu et al., 2024; Lazebnik et al., 2022; Xia et al., 2024; Savchenko and Lazebnik, 2022; Shami and Lazebnik, 2023; Oren et al., 2022; Savchenko et al., 2023). While ML models show promising results in the lab as well as in realistic scenarios, deployed ML models experience a wide range of challenges in production settings. A common and important challenge these models encounter is adapting to dynamic and evolving environments (Žliobaite et al., 2016; Gama et al., 2014; Krongauz and Lazebnik, 2023; Shyaa et al., 2024). In particular, concept drift (CD), the phenomenon wherein the statistical properties of the target variable change over time, poses a significant hurdle to the stability and performance of learning-based models (Lu et al., 2019). The dynamic nature of real-world data introduces uncertainties, necessitating the continuous adaptation of models to maintain their relevance and accuracy. Recent research in the CD domain focuses on addressing three main challenges. Precisely identify CD within unstructured and noisy datasets (Goncalves et al., 2014; Harel et al., 2014; Wang et al., 2013), to comprehensively comprehend CD in a quantifiable and interpretable manner (Wang et al., 2024; Xiang et al., 2023), and to respond effectively to CD (Gama et al., 2014; Yang et al., 2022).

A genetic algorithm (GA) is a search and optimization technique inspired by the principles of natural selection and genetic inheritance (Holland, 1992; Kumar et al., 2010; Alhijawi and Awajan, 2024). It operates by iteratively evolving a population of potential "solutions" to a problem through mechanisms such as selection, crossover, and mutation, mimicking the process of biological evolution to find optimal or near-optimal solutions (Bo and Rein, 2005; Tyagi et al., 2024; Ghaheri et al., 2005). The motivation behind employing genetic algorithms lies in their ability to efficiently explore large solution spaces, enabling the discovery of diverse and effective solutions that may be elusive through traditional optimization methods (Zhao and Xu, 2013; Routledge, 2001; Drake and Marks, 2002). In particular, GAs have been adapted to the realm of ML to find a well-performing ML pipeline (Olson and Moore, 2016), solve a symbolic regression task (Worm and Chiu, 2013; Kammerer et al., 2020; Raissi et al., 2019), or even as part of an ML ensemble-based model (Shapiro, 2001; Kuptametee et al., 2024; De Jong, 1988).

Generally speaking, GAs hold significance in ML for three primary reasons. First, they operate in discrete spaces, making them applicable

* Correspondence to: Department of Cancer Biology, Cancer Institute, University College London, London, UK.
E-mail address: lazebnik.teddy@gmail.com.

https://doi.org/10.1016/j.engappai.2025.110772
Received 21 April 2024; Received in revised form 18 March 2025; Accepted 1 April 2025

in scenarios where gradient-based methods are impractical (Herrmann, 1999). Second, GAs function as reinforcement learning algorithms, evaluating the performance of a learning system based on a singular metric, commonly referred to as the "fitness" function, in contrast to approaches like back-propagation where different parts of the model received different optimization signals. This characteristic makes them particularly useful in situations where performance measurement is the sole available information (Sehgal et al., 2019; Chen et al., 2020). Third, GAs involve a population, making them suitable for scenarios where the desired outcome is not a single model but a set of models, as exemplified in learning within multi-agent systems (Heppenstall et al., 2007).

To this end, GAs can be used to overcome CD (Padmalatha et al., 2015; Smith and Ciesielski, 2016; Kou et al., 2024). In this study, we focused on the last point as in the context of deployed ML-based solutions where it is trained on initial data and more (tagged) data is gathered over time, GAs can be used to respond to changes in dynamics. Intuitively, one can think of CD as the change between the source (x) and target (y) features where some model (f) is applied at two points in time (t_1, t_2), such that $\|f_{t_1}(x) - y\| \leq \|f_{t_2}(x) - y\|$, where $\|\cdot\|$ is arbitrary norm operation, usually the L_1 or L_2 norms (Luo et al., 2016). Hence, as new data is introduced to the learner, a mechanism should alter the model f aiming to obtain $\|f^1_{t_1}(x) - y\| = \|f^2_{t_2}(x) - y\|$ where f^2 is originated from f^1 and altered to achieve the above condition. Building on this idea, we propose a two-level ML model derived using a GA algorithm. It consists of a global ML model with a CD detector functioning as an ensemble model for a population of ML models. Each of these models governs a subset of the data and adapts autonomously based on its own CD detector.

The rest of the paper is organized as follows. Section 2 presents an overview of CD properties, challenges, and previous solutions, as well as the recent developments in the field of GAs and Ensemble ML. Section 3 provides a technical background later used in the model definition. Section 4 formally outlines the task definition, the proposed algorithm based on GA, and its applicative improvement in the form of using automatic ML models with a divide-and-conquer approach. Section 5 introduces the experimental setup used to explore the proposed algorithm. Section 6 shows the obtained results. Finally, Section 7 discusses the results with their potential applicative usage as well as the limitations of this study and possible future work.

2. Related work

In this section, we present approaches in the field of CD adoption followed by an overview of the GA models used in dynamical systems. Afterward, we review several cases where GA is used in the context of CD and ensemble ML models.

2.1. Concept drift

A discussion about CD contains two interconnected dichotomy aspects — the theoretical and applied aspects of defining, detecting, and tackling CD. It is more often than not the applied aspect that governs the broad interest in CD as ML users experience first-hand the challenges that come with CD in their respective tasks (Zliobaite et al., 2016). Indeed, in a dynamic world, nothing is constant. For instance, let us consider a supply chain distribution system responsible for distributing a company's products between its physical stores. Would one should expect that a model that was trained before COVID-19 (Lazebnik and Bunimovich-Mendrazitsky, 2021), would work equally well during or even after the COVID-19 pandemic? It is reasonably easy to assume that due to these kinds of unforeseen circumstances, user behavior would change a lot, as indeed happened in practice (Chowdhury et al., 2021; Rahman et al., 2022; Pujawan and Bah, 2022).

As such, a field of repetitive ML model adoption has been proposed where new data is used to re-train ML models to detect, capture, and utilize the changes in the data over time, practically addressing CD (Maggi et al., 2009; Madireddy et al., 2019). One of the most direct approaches to address CD involves retraining a new model with the latest data to replace the outdated model and dynamics constructed it (Vivekanandan and Nedunchezhian, 2011; Buchgraber et al., 2011). This method necessitates an explicit CD detector to determine when model retraining is required. This approach does not work well when the change over time is relatively slow and "smooth" and excels in hard shifts in the system's dynamics. The complementary approach is to use a "window" strategy where the model is retrained on the latest data with some fixed size. A more sophisticated example of this approach is employed by *Paired Learners*, which utilizes two learners — the stable learner and the reactive learner (Bach and Maloof, 2008). If the stable learner consistently misclassifies instances correctly identified by the reactive learner, signaling a new concept, the stable learner is replaced with the reactive learner.

These two approaches cover the basic ideas of CD overcoming in ML. That said, each approach raises a new computational challenge one needs to tackle. The first approach requires the accurate detection of CD, while the latter challenges the user to find and use the optimal window size. On top of that, as each approach is appropriate to different types of CD, choosing the appropriate one for each case is a challenge in itself. This fertile soil was the base of multiple solutions.

Initially, attempts to find the optimal window size have been conducted as a compromise must be made in determining the suitable window size. A smaller window effectively mirrors the most recent data distribution, while a larger window affords more data for training a new model. Consequently, Bifet and G. (2007) proposed ADWIN, an algorithm that dynamically adjusts subwindow sizes based on the rate of change between sub-windows, eliminating the necessity for users to predefine a fixed window size. After determining the optimal window cut, the window containing outdated data is discarded, facilitating the training of a new model with the latest window data.

Moving beyond mere model retraining, researchers have delved into the integration of the drift detection process with the retraining mechanism tailored for specific ML algorithms rather than an "one method to rule them all" approach. For example, Huang et al. (2006) proposed DELM, which extends the conventional ELM algorithm to handle concept drift by adaptively modifying the number of hidden layer nodes. Moreover, instance-based lazy learners also show promising results for CD handling (Fdez-Riverola et al., 2007). For example, Lu et al. (2016) proposed NEFCS, a kNN-based adaptive model that utilizes a competence model-based drift detection algorithm to identify drift instances in the case base and distinguish them from noise instances.

2.2. Genetic algorithm

GAs belong to a category of approaches commonly known as evolutionary computation methods, employed in adaptive aspects of computation such as search, optimization, machine learning, and parameter adjustment (Sohail, 2023). What distinguishes these approaches is their characteristic reliance on a population of potential solutions. Unlike most search algorithms that focus on modifying a single candidate solution to enhance its performance, evolutionary algorithms dynamically adapt entire populations of candidate solutions to address the problem at hand. Drawing inspiration from biological populations, these algorithms incorporate selection operators to amplify the number of superior solutions within the population while diminishing the presence of inferior ones (Bo et al., 2006; Salehi and Bahreininejad, 2011). Additionally, they employ other operators to generate novel solutions. The variability among these algorithms lies in the standard representation of problems and the nature and relative significance of the operations introducing new solutions (Davis, 1985; Hassanat and Alkafaween, 2017; Kaya et al., 2011).

GAs have found application across diverse domains, including engineering (Bo and Rein, 2005; Huang et al., 2024), medicine (Ghaheri

et al., 2005; Mishra and Bajpai, 2024), and economics (Zhao and Xu, 2013; Joo et al., 2024). For instance, Salehi and Bahreininejad (2011) addressed the challenge of optimizing sequences of machines and their corresponding operations for process planning optimization. Employing GAs, the authors derived feasible processes initially and subsequently identified the optimal process from this set of viable alternatives. Similarly, Bo et al. (2006) investigated and assessed the utilization of GAs under various constraints in process route sequencing and astringency. The authors revamped the GA, encompassing the development of coding strategies, the evaluation operator, and the fitness function. Their findings demonstrated that these modified GAs could effectively fulfill the requirements of sequencing tasks and meet the criteria for astringency. In another study, Zhao and Xu (2013) introduced an optimization scheme based on GAs to enhance Atkinson fuel engine models, specifically targeting fuel consumption reduction (Zhao and Xu, 2013; Zhao et al., 2012). GAs were chosen due to the high-dimensionality and non-linearity of the optimization system, rendering classical methods time and resource-intensive. Furthermore, the authors proposed GAs as a financial model, illustrating their applicability in learning signal utilization, making inferences from market-clearing prices, and assessing the worthiness of acquiring a signal (Routledge, 2001). In the economic domain, Ariel et al. (2023) presented an agent-based model with a heterogeneous population and genetic algorithm-based decision-making to model and simulate an economy with taxation policy dynamics. Furthermore, for the clinical domain, GAs have been widely used as well. For example, Lazebnik (2022) used GA to obtain the parameters of a partial differential equations-based model describing an immunotherapy treatment for bladder cancer.

2.3. Genetic algorithm for concept drift

A growing body of work finds the usage of GA to detect and adapt to CD promising in a wide range of tasks and data settings (Iwashita and Papa, 2019). In particular, in realistic applications, CD is of interest when new data is obtained once an initial ML model is obtained (Oliveira et al., 2017; Agrahari and Singh, 2022).

Ghomeshi et al. (2019) introduces an innovative ensemble learning approach relying on evolutionary algorithms to address diverse forms of concept drifts in non-stationary data stream classification tasks. The authors employ random feature subspaces drawn from a feature pool to construct distinct classification types within the ensemble. Each type comprises a finite set of classifiers (decision trees) constructed at various instances throughout the data stream. Utilizing an evolutionary algorithm, specifically replicator dynamics, the system adapts to varying concept drifts by enabling types with superior performance to expand and those with inferior performance to diminish in size.

Tareq and Sundararajan (2020) proposed a novel Density-based method for Clustering Data streams employing Genetic Algorithm (DCDGA). This approach leverages a GA to optimize parameters, specifically the cluster radius and minimum density threshold, ensuring more precise coverage of density clusters. Additionally, a Chebyshev distance function is introduced to compute the distance between the center of Core Micro-Clusters (CMCs) and the incoming data points. The authors evaluated DCDGA on both artificial and real datasets and showed that the experimental results were comparable with another online density-based clustering in the field.

Kuranga and Pillay (2021) introduces a predictive model for temporal data with a numerical target, utilizing GA to capture changes in a dataset caused by concept drift. In the presence of environmental changes, which stands for the CD in these settings, the author's proposed algorithm responds by clustering the data and subsequently creating nonlinear models that characterize the formed clusters. These nonlinear models serve as terminal nodes within the GA model trees.

Henke et al. (2021) developed a spam detection system that examines the evolution of features. The author's proposed method encompasses three key steps. First, training a spam classification model; second, detecting CD using a new strategy that analyzes feature evolution based on the similarity between feature vectors extracted from training and test data; and finally, knowledge transfer learning. In the last step, the focus is on determining what knowledge to transfer, how to transfer it, and when to execute the knowledge transfer process.

2.4. Ensemble machine learning models

Ensemble ML methods leverage multiple ML algorithms to generate weak predictive results by extracting features through diverse projections of the data. These results are then fused using various voting mechanisms to achieve superior performance compared to that obtained from any individual algorithm in isolation (Dasarathy and Sheela, 1979). Indeed, ensemble ML models show superior results on a wide range of tasks (Schapire, 1990). The fundamental concept of a standard ensemble ML model involves two stages: producing prediction outcomes through numerous weak classifiers and consolidating these multiple results into a consistency function to obtain the ultimate result using voting schemes. The voting scheme can range in complexity, from a simple average or majority vote for regression and classification tasks, respectively, such as for the case of the Random Forest model (Breiman, 2001) which is based on a set (forest) of Decision Tree models (Swain and Hauska, 1977) to being an ML or deep learning model by itself (Drori et al., 2021).

The weak prediction ML models in the set of an ensemble method differ from one another by one or more of the following three properties. First, the samples provided to the model, the features provided to the model, and even the ML model itself (Dong et al., 2020). These changes allow each weak prediction model to focus on a less complex pattern in the data and excel in capturing it. Hence, the more weak models an ensemble model includes, the more complex patterns it can capture. However, it also increases the bias given the data training data (Dong et al., 2020). Importantly, this scheme can be generalized where weak models in an ensemble model can be ensemble models by themselves. For example, imagine a Random Forest model in which each Decision Tree is replaced with a Random Forest as well. This example will result in three levels of models.

3. Technical background

In this section, the necessary technical background, later used by the proposed model, is presented. Initially, a formal definition of CD with its two main types is provided. Next, an introduction to automatic ML (AutoML) models is presented.

3.1. Concept drift definition

A learning algorithm, A, observing samples with a stationary distribution would observe the training cohort in the form (x_i, y_i) such that x_i is the feature vector and y_i is the target feature. A class prediction at a specific point in time t^* would be given as y_{t^*} based on the feature vector x_{t^*}. Opposed to this, a data stream may produce samples with a non-stationary distribution. In such a scenario, the (x_i, y_i) is obtained by a distribution that explicitly depends on time or previous samples measured from the distribution. Formally, a CD between two points in time t_0 and t_1 can be defined as: $p_{t_0}(x_i, y_i) \neq p_{t_1}(x_i, y_i)$, where p_t is the joint distribution at time t between the feature vector x_i and the target feature y_i (Gama et al., 2014). Following this definition, CD can occur due to three main reasons: the distribution of samples in the target feature can change; the distribution of samples in the target feature can change concerning the samples of the source features; and the source features distribution can change while the target feature does not.

In addition to the fact that CD occurred, the rate at which it happens is also of interest. Simply put, the rate at which CD takes place can be roughly divided into two main forms: shift and moving CD. The shift drift is associated with sudden changes in the distribution while the

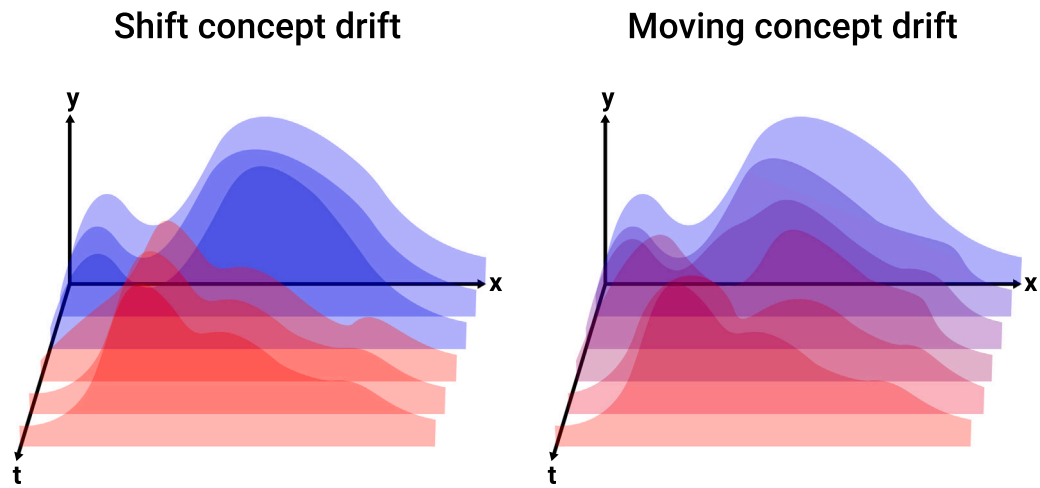

Fig. 1. A schematic view of shift and moving CD. One can notice that the shift CD moves from one two-dimensional distribution (x, y) to another distribution drastically. On the other hand, the moving CD gradually alters from the same source distribution to the other distribution. The distribution's color indicates their alteration over time between the initial distribution (marked in blue) and the final distribution (marked in red).

moving drift occurs at a much slower rate usually with multiple phases in between (Brzezinski and Stefanowski, 2014). Formally, let us assume two distributions $p_{t^*}(x_i, y_i)$ and $p_{t^*+\Delta t}(x_i, y_i)$ of the data associated with two points in time t^* and $t^* + \Delta t$, respectively, such that $\Delta t \in \mathbb{R}^+$ is the time passed since the initial point in time, t^*. In addition, let us assume a threshold value $\psi \in \mathbb{R}^+$. A drifting rate is defined to be

$$(1 - KS(p_{t^*}, p_{t^*+\Delta t}))/\Delta t, \tag{1}$$

such that $KS(a, b)$ is the p-value of a Kolmogorov–Smirnov test (Berger and Zhou, 2014) between the two distribution a and b. Fig. 1 presents a schematic view of shift and moving CD where the shift CD moves from one two-dimensional distribution (x, y) to another drastically while the moving CD gradually alters from the same source distribution to the other distribution.

3.2. Automatic machine learning

The process of ML model development is time-consuming, requires substantial expertise, and is susceptible to human errors (Lazebnik et al., 2023). To this end, AutoML has emerged as a promising approach that automates many steps in the ML development process, including data pre-processing, feature engineering, model selection, and hyperparameter tuning, thereby mitigating the challenges associated with using ML models (Yao et al., 2019; Nisioti et al., 2018; Pinto et al., 2017; Molino et al., 2019; Baratchi et al., 2024).

Multiple models have been proposed in recent years for Automatic machine learning (Lazebnik and Rosenfeld, 2023). For instance, the Tree-based Pipeline Optimization Tool (TPOT) library utilizes a GA search process to identify ML pipelines based on the popular Scikit-learn library (Pedregosa et al., 2011). TPOT uses a tree-based representation to evolve and optimize these pipelines based on their performance, aiming to find the most effective combination for a given dataset. The library represents machine learning pipelines as tree structures, providing a flexible and hierarchical way to organize and evolve complex combinations of data preprocessing and modeling steps. The AutoSklearn library (Feurer et al., 2019, 2020) employs various search methods to construct an ML pipeline, also based on the Scikit-learn library. It employs meta-learning and Bayesian optimization techniques to search efficiently through various preprocessing steps, feature engineering methods, and model configurations. AutoSklearn incorporates meta-learning, leveraging information from previous ML tasks to guide the search for effective pipelines. The AutoGluon library (Erickson et al., 2020) strategy is based on the idea of ensembling multiple models and stacking them in multiple layers. AutoGluon uses a fixed defaults (set adaptively) strategy for the search process of ML models in each layer and then combines multiple layers using the stacking and repeated bagging methods. The PyCaret library (Ali, 2020) is a Python wrapper around several popular ML libraries and frameworks which uses a multi-metric comparison of these models, in a brute-force manner, to find the best model for a given dataset and task.

4. Ensemble machine learning model for concept drift data

In this section, we outline the proposed GA-based solution for CD in data streams based on ensemble machine learning models. The proposed model is based on the fact that there are existing feasible, and even well-performing, solutions for CD types when these occur individually or under some assumptions. Furthermore, as there is no one solution to rule out all CD types, one is required to find an appropriate solution for each case. However, using multiple models, it is possible to activate one or a subset of these models as the dynamics of the system alter over time. Fig. 2 presents a schematic view of the learning problem during different CD types and possible remedy with an ensemble ML model obtained using an initial search process. Intuitively, one can perform a two-step optimization process where the first step is responsible for the ensemble configuration in terms of the model and the data it obtained during the training phase, and the second step is to find and train an ML model with the CD-handling method that best suits the data it obtained.

4.1. Task definition

Initially, to measure how well a population of ML models handles a CD scenario, a metric needs to be defined. Intuitively, the population of ML models should perform well on the initial dataset at some time, t, and not lose this performance over some fixed duration of time. Hence, let us consider a population of ML models, $\mathbf{M} := [M_1, \dots, M_k]$ and a dataset that increases over time $D(t) \in \mathbb{R}^{n \times m}$, such that each model $(M_i \in M)$ obtains a subset, $D_i \subset D(t)$ at some point in time t. For a given event horizon $\tau \in \mathbb{R}^+$ and a performance metric, ψ, the CD handling performance of the ML models population, $L(M, D(t))_{\psi,\tau}$, is defined as follows:

$$L(M, D(t))_{\psi,\tau} := \omega_1 \psi(M, D(t)) - \omega_2 \big(\psi(M, D(t)) - \psi(M, D(t+\tau))\big), \tag{2}$$

where $\omega_1 \in \mathbb{R}^+$ and $\omega_2 \in \mathbb{R}^+$ are the weights of the model's performance at the end of the training phase and the weight of the CD's influence on the models' performance, respectively. Based on this definition, one

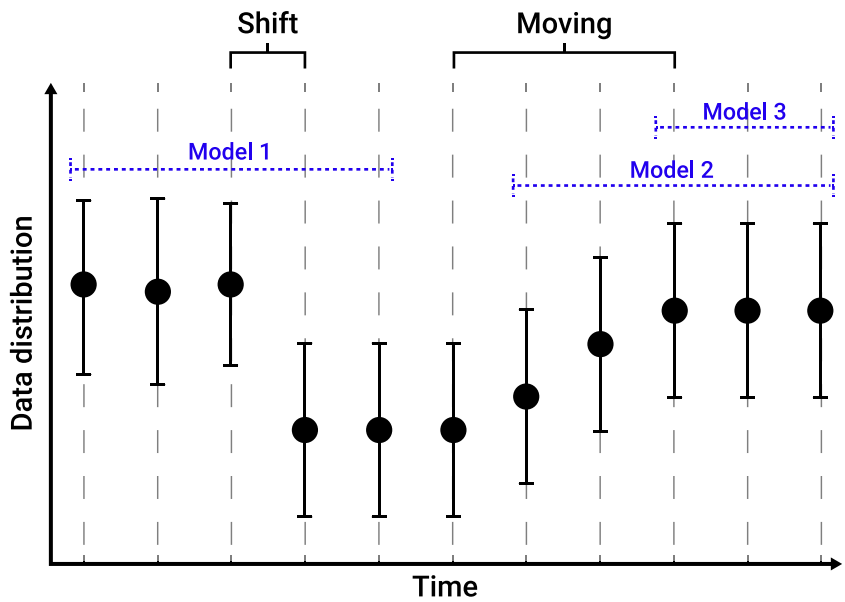

Fig. 2. A schematic view of the learning problem during different CD types and possible remedy with ensemble ML model obtained using an initial search process. The distributions over time are shown as the mean ± standard deviation of some random variable, as reflected on the y-axis. The x-axis indicates steps in time. In this example, a shift CD has occurred between the third and fourth steps in time. In addition, a moving CD has occurred between the sixth and tenth steps in time. A possible ensemble model to tackle this condition would detect the shift and moving CDs and use three models, one to capture the original data between the CDs, a second model that takes into account the recent, seemingly stable, data with some "tail" of the moving CD, and a third model that based only on the recent time without CD.

can formalize an optimization task to find the population of ML models as follows:

$$\max_{M} \; L(M, D(t))_{\psi,\tau}. \tag{3}$$

4.2. Genetic algorithm based solution

One can solve Eq. (3) in multiple ways. Naively, assuming a finite number of ML models and a maximum number of models in the population, one can theoretically brute-force the optimization task. Nonetheless, due to the extremely large set of possible solutions, such an approach is infeasible in practice (Chauhan et al., 2020). In an opposite manner, one can try to solve this optimization task analytically; however, without further assumptions over either M, $D(t)$, or ψ, it seems infeasible to obtain such a solution. Thus, one can use a heuristic approach to solve Eq. (3). In this study, we suggest an adoption of the classical GA for this task.

Formally, we assume that the population of ML models, M, is defined by both the number of models, n, as well as the models themselves. Each ML model, $M_i \in M$, is contracted from a feature engineering algorithm, a supervised ML algorithm, and a hyperparameters tuning algorithm, each of these algorithms is chosen from a pre-defined and finite set of algorithms. In addition, each ML pipeline model is affected by the data it is provided with during the training and testing phases. As such, each model should be provided with a non-empty subset, $d_i \subset D(t)$ that is used to train the ML pipeline model. Based on this representation, solving for Eq. (3) would optimize $\psi(M, D(t))$ while is not considered $\psi(M, D(t)) - \psi(M, D(t+\tau))$ as no remedy for the treating change over time for $D(t)$ is considered. Consequently, one should include a CD detection algorithm for each ML pipeline model as well as for the entire population. As such, M can be represented by $M := (f_g, m_g, h_g, D(t), cd_g), n, \big((f_1, m_1, h_1, d_1, cd_1), \dots, (f_{n-1}, m_{n-1}, h_{n-1}, d_{n-1}, cd_{n-1}), (f_n, m_n, h_n, d_n, cd_n)\big)$ such that $f_i \in \mathbb{F}$ is the feature engineering algorithm, $m_i \in \mathbb{M}$ is the supervised ML algorithm, $h_i \in \mathbb{H}$ is the hyperparameters tuning algorithm, $d_i \subset D(t)$ is subset of the data used to train (f_i, m_i, h_i), $cd_i \in \mathbb{CD}$ is the CD detection algorithm. In addition, (f_g, m_g, h_g, cd_g) is the global ML pipeline model responsible which gets as input the output of (f_i, m_i, h_i) and cd_i for each $i \in [1, \dots, n]$ and making the final prediction of the M model. To this end, we proposed a genetic-based algorithm for finding a population of ML models to handle CD. The algorithm works as follows. First, a population of M solutions (i.e., a population of ML pipeline model populations) is generated at random such that each ML pipeline is chosen at random with a uniform distribution. Specifically, the ML pipelines as well as the CD detector algorithms are chosen at random with a uniform distribution. However, d_i are chosen by picking indexes t_1 and t_2 with a Poissonian distribution decaying from the latest sample in $D(t)$ to the first one. In addition, the performance of the best solution from P_0, as defined by Eq. (2), is computed. Now, for $\psi \in \mathbb{N}$ generations, three operations are taking place, the *mutation*, *crossover*, and *selection* operators to generate the next-generation population P_{i+1}. If $M \in P_i$ is found to be better than the best performing M so far, it becomes the best M. The best-performing M during the entire process is returned as the answer of the model. The mutation operator is stochastically employed, for each $M \in P_i$, with probability $\xi \in [0, 1]$. First, we randomly decide if to mutate the global ML model or one of the inner ML models in the population w.r.t. with probability $\zeta \in [0, 1]$. Either way, we replace one of the components of the ML pipeline model with another one, in a uniform manner distribution. For the case of d_i, the start index (t_1) or the end index (t_2) is altered. First, the start or end index is randomly picked (with equal probability), and then a rounded value x which is distributed normally with $\mu \in \mathbb{R}^+$ and $\sigma \in \mathbb{R}^+$ as the mean and standard deviation of the distribution. Cross-over is employed for two M - M_a and M_b in population P_i with the goal of creating two next-generation M_a and M_b. We randomly choose a split-size $1 < s < min(|M_a|, |M_b|)$, and use it to split both ML populations, each to two random subsets — one of size s and one of size $|M_a| - s$ and $|M_b| - s$, respectively. i.e., $M_a = M_a^s \cup M_a^{|M_a|-s}$ and $M_b = M_b^s \cup M_b^{|M_b|-s}$. The cross-over then unifies complementing subsets from a and b, creating M_{ab} and M_{ba} as follows $M_{ab} = M_a^s \cup M_b^{|M_b|-s}$, $M_{ba} = M_b^s \cup M_a^{|M_a|-s}$. The cross-over operation is performed over the entire population P_i. P_i is first split into disjointed pairs of M, and then the cross-over is performed on each such pair. Last, after employing mutation and cross-over, we employ the selection operator, which forms the next-generation population P_{i+1}. We use the *royalty tournament* operator (Bo et al., 2006), which selects the best $\alpha \in [0, 1]$

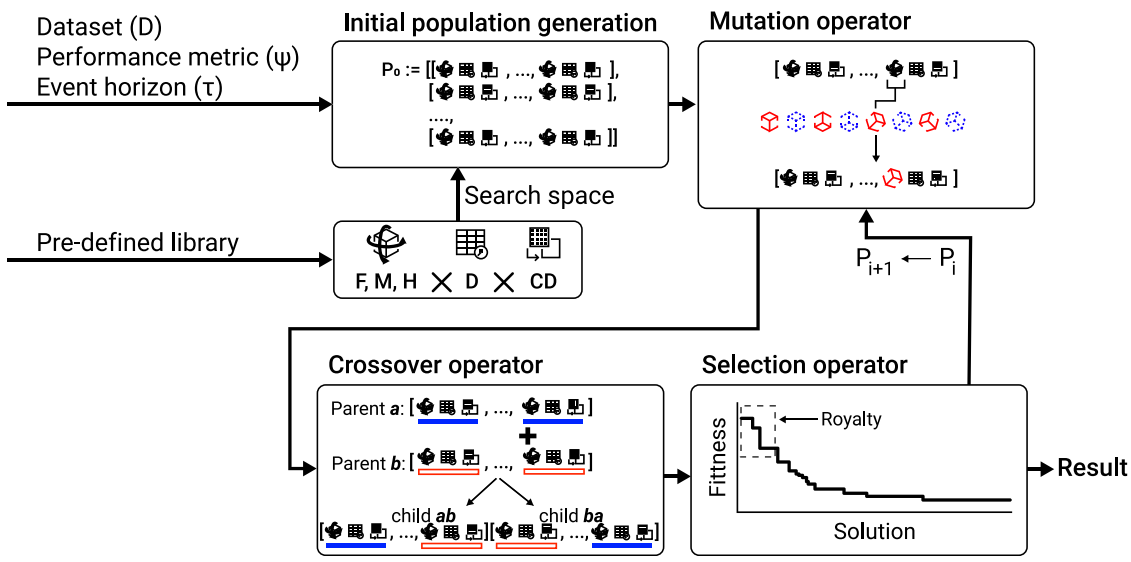

Fig. 3. A schematic view of Algorithm 1. Initially, the algorithm receives a dataset (D), performance metric (Ψ), and even horizon (τ), as well as a pre-defined library of feature engineering functions, ML models, hyper-parameter tuning functions, sub-set of the data used to train the ML models, and concept drift detection functions. These are provided to a GA-based optimization process, which starts with an initial population of ML pipelines, iteratively performing mutation, crossover, and selection operators.

ML populations from P_i according to the fitness function $L(M, D)$. The rest of the ML populations are sampled (with repetitions) according to their fitness score, i.e., with probability $p_{select}(M) = \frac{L(M,D)}{\sum_{M' \in P_i} L(M',D)}$. A pseudo-code representation of the proposed algorithm is presented in Algorithm 1. Fig. 3 provides a schematic view of Algorithm 1.

Algorithm 1 Genetic algorithm for population of ML models during concept drift

Input: dataset (D), performance measuring metric (ψ), event horizon (τ)
Output: population of ML models (M)
$P_1 \Leftarrow$ generate $(n_t \sim [2, N])$ ML pipelines in random
$L_{best} \Leftarrow \forall M \in P_1 : \max(L(M, D))$
for generation $i \in [1, \dots, \phi]$ **do**
  $P_i \Leftarrow Mutation\_Operator(P_i, D.\psi, \tau)$
  $P_i \Leftarrow Crossover\_Operator(P_i, D.\psi, \tau)$
  $P_{i+1} \Leftarrow Selection\_Operator(P_i, D.\psi, \tau)$
  if $\exists M \in P_i : \max(L(M, D)) > L_{best}$ **then**
    $L_{best} \Leftarrow \max_M(L(M, D))$
  end if
end for
return $argmax_{M \in P_i} \max(L(M, D))$

4.3. Improved version using the divide and conquer approach

The proposed model combines several optimization tasks: finding the optimal ML pipeline (f_i, m_i, h_i) for each ML model in the population, where each model is given a subset of the dataset d_i; determining the optimal subset of data and CD model (d_i, cd_i) for each ML pipeline; and identifying the global ML model and its associated CD model. Technically, the model aims to solve three interdependent optimization tasks, each focusing on different aspects of the overall solution. Notably, the CD model for each ML pipeline in the ensemble does not affect the other components in the pipeline; it only serves as an input to the global ML model. Therefore, once the subset of the dataset, d_i, has been processed, the optimization of each ML pipeline becomes independent of the others. After all the ML models in the ensemble have been obtained, the task simplifies the optimization of the global ML pipeline. In this context, the CD models (cd_i) for each pipeline can be treated as constraints within the feature engineering component of the global model f_g.

Thus, the model can be described as follows: First, the dataset $D(t)$ is divided into subsets, d_i, which are provided to the ML pipeline models in the ensemble. For each subset d_i, the goal is to find the optimal ML pipeline model (f_i, m_i, h_i), where f_i represents feature selection, m_i is the model, and h_i refers to the hyperparameters. After optimizing each pipeline, the next step is to find the optimal global ML pipeline, alongside the adjustment CD model.

To solve this representation, the second and third tasks can be handled using an AutoML model specifically designed for this purpose. For the data splitting task, we use Algorithm 1, with a slight modification: the algorithm assumes each model in the ensemble is defined solely by its corresponding subset d_i. This ensures that each subset is treated as an independent entity during optimization while still contributing to the global model's optimization. This approach guarantees that the global model optimally integrates all the local pipelines, considering the diverse data subsets and ensuring that concept drift is effectively addressed.

5. Experimental setup

Evaluating the proposed algorithm requires three components: dataset, baseline comparison, and performance evaluation metric. In this section, we outline these components which will be used in the following section to assess the proposed algorithm.

5.1. Dataset curation

In order to evaluate the proposed model, one is required to obtain a statistically large set of cases with CD in various levels and combinations. Moreover, as different CD can happen in multiple ways in parallel for different subsets (usually features) of the dataset, one should include this representation in the model. Due to the multiple moving parts and the challenge of detecting CD accurately (Demsar and Bosnic, 2018), we used synthetic datasets for our analysis. Intuitively, when developing an ML model, some dataset is already established based on data gathered, marked by $D(0)$. This can be considered the initial (or first) phase of the data curation process. At this point, we can assume no CD is present and there is some connection between the target and source features. After this point, as part of the second phase, more data streams to the dataset over time with discrete steps. At each step in time, there is either a CD event or not. If no CD is present, more data, according to

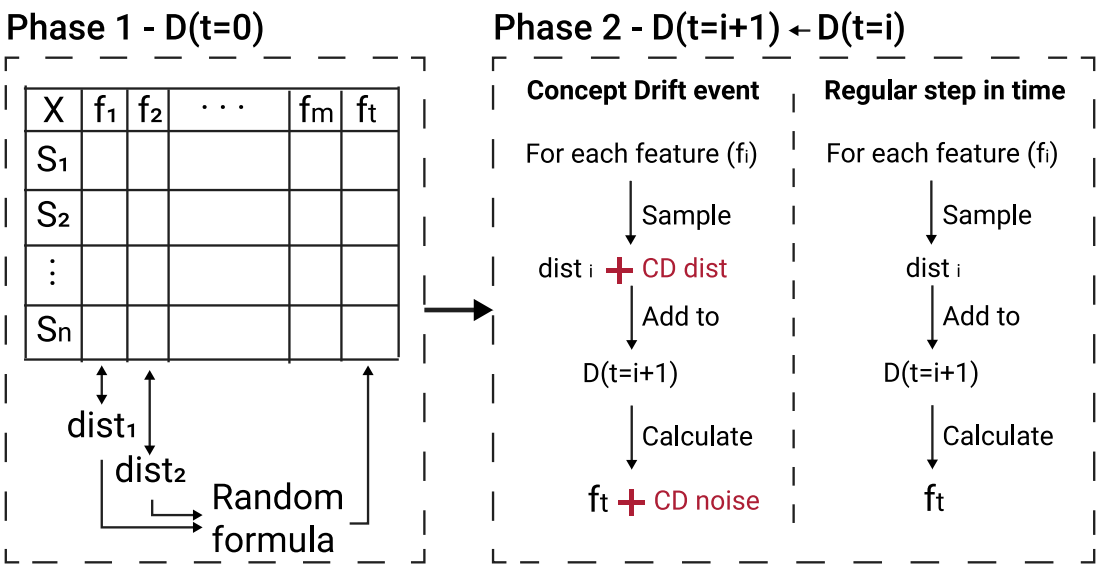

Fig. 4. A schematic view of a dataset generation process for the experiments. Initially, as the first phase, an initial dataset ($D(0)$) where no CD is present and a connection exists between target and source features is generated. Next, as a second phase, new data streams in discrete steps, either maintaining the original distribution or experiencing CD, which alters feature distributions and target-source relationships.

some distribution (which is unknown for the ML model) is generated at random and added to the dataset, $D(t)$, at time t. Otherwise, a concept drift event is associated with the change of a subset (or all) features' distributions of the dataset during a period of $\Delta t \in \mathbb{N}$. In addition, CD can also introduce change to the connection between the source and target feature, which for the ML is reflected like noise. Fig. 4 shows a schematic view of the dataset generation process for the experiments.

Formally, a dataset ($D(t)$) is generated as follows. First, the initial data, $D(0)$, is generated followed by a function $F : D(t) \rightarrow D(t+1)$ which accepts the dataset at a point in time, t, and returns the same dataset for the next point in time, $t+1$. For the initial data, a random number of samples, $s \in \mathbb{N}$, and features, $f \in \mathbb{N}$ are picked at random from some pre-defined distribution. Then, for each feature, a random distribution is chosen from a pre-defined set of known distributions, and s samples are obtained from it. In order to obtain a meaningful regression task, the target feature of each dataset is obtained by defining a random function (contracted from a combination of polynomials, exponential, trigonometric, logarithmic, and step-wise functions) which gets as an input the other features of the dataset. To be exact, the function is constructed by randomly picking a topology for an expression tree where each leaf node is a feature from the dataset and each decision node is a function from a pre-defined set of functions (Keren et al., 2023). For the function, F, a sequence of CD events is set alongside a default behavior. For the default behavior, which is used when there are no CD events for a specific step in time, a random number $\zeta \in \mathbb{N}$ of new samples is generated for each feature using the current distributions associated with each feature and then the target feature is computed for these samples using the current formula. In a complementary manner, a CD event at some time changes the distribution of a subset (or all) features in the dataset over a period of Δt steps in time as well as changing the formula used to calculate the target feature.

In order to obtain a robust representation of an algorithm's performance, we generated multiple datasets for each explored configuration. The hyperparameters used as part of the data generation process are summarized in Table 1.

For the experiments, we used the ML components available in the Scikit-learn library (Pedregosa et al., 2011). In addition, we used the CD algorithms provided by the Frouros library (Céspedes-Sisniega and López-García, 2022).

5.2. Baseline algorithms

For the comparison of the proposed algorithm with other configurations, we establish three baseline models. First, a *single-random* model where only a single ML pipeline and CD model are chosen at random. Second, a *multi-random* model where the proposed ML population with global ML model is used, but each of them is picked at random. Third, a single ML pipeline with a CD algorithm is obtained using TPOT and brute force, respectively. Fourth, an ensemble of models obtained using TPOT and updated using the Dynamic Weighted Majority (DWM) algorithm (Kolter and Maloof, 2007; Yan et al., 2022). In addition, the proposed model and its improvement are considered in the evaluation as the fifth and sixth candidates, respectively.

For both the TPOT and proposed algorithms, the underlined GA method requires setting several hyperparameter values. To this end, we set the population size to 100, the crossover rate to 0.8, and the mutation rate to 0.025. For the selection process, we set the royalty to 5%, and we applied a uniform crossover strategy. The algorithm was run for 50 generations with an early stopping criterion of 5 generations if no improvement was observed. These hyperparameters were chosen based on a manual trial-and-error approach and following best practices from other studies (Shmuel et al., 2024a; Alibrahim and Ludwig, 2021; Žegklitz and Pošík, 2021).

5.3. Performance metric

Assessing the performance of the overall model requires measuring the performance of the model on various tasks over time. To this end, for a specific dataset, $D(t)$, and a time frame of interest, $t_0 - t_1$ ($t_1 > t_0$), the performance of the model is defined to be the average performance over all steps in time. Formally, the ML pipeline with the CD adoption model's performance is defined to be (Bayram et al., 2022):

$$\theta(M) := \frac{1}{(t_1 - t_0)N} \sum_{i=0}^{N} \sum_{j=t_0}^{t_1} \psi\left(M, D_j(i)\right) \tag{4}$$

6. Results

In this section, we outline the results of the experiments. First, we compare the proposed model to other single- and multi- ML pipeline solutions. Afterward, we explore the robustness and sensitivity of the best solution to the concept drift rate, dataset's size, and dataset's complexity.

Table 1
The hyperparameters used as part of the data generation with their value ranges.

Hyperparameter	Description	Value range
n, m	The number of samples and features in the initial dataset [1]	$[10^2, 10^5, 3-50]$
$dist_i$	The distribution of a feature in the dataset	Normal, Exponential, Binomial, Geometric, Benford, Uniform [1]
η	Formula topology's size [1]	$[n, 10n]$
τ	Formula functions [1]	$[+, -, *, exp, log, inv, sin, scalar, step-wise]$

Table 2
Comparison of different ML pipelines with CD detection models for four different CD cases — shift, moving, mixed, and random. The results are shown as the mean with the standard deviation in brackets of Eq. (2).

Model	Shift	Moving	Mixed	Random
Single-random	0.59 (0.13)	0.65 (0.11)	0.47 (0.18)	0.48 (0.18)
Multi-random	0.63 (0.10)	0.69 (0.08)	0.56 (0.15)	0.58 (0.16)
TPOT + Brute-force CD	0.68 (0.09)	0.75 (0.06)	0.52 (0.16)	0.51 (0.16)
TPOT + DWM	0.66 (0.06)	0.77 (0.07)	0.59 (0.16)	0.55 (0.15)
Proposed	0.74 (0.07)	0.80 (0.06)	0.64 (0.11)	0.64 (0.11)
Proposed improved	0.76 (0.06)	0.81 (0.05)	0.67 (0.10)	0.67 (0.10)

6.1. Performance comparison

Table 2 presents the comparison between the five models with the metric presented in Eq. (4) such that Ψ is taken to be Eq. (2) for four cases — shift, moving, mixed, and random CD. For the shift concept drift, the datasets are generated with a single CD starting at a random point in time and have a random drift rate between 0.1 and 0.2, chosen in a uniformly distributed manner. Similarly, the moving CD case is identical to the shift case but with a drift rate between 0.01 and 0.02. For the mixed case, a random number of CD can occur ranging between 2 and 10 with the constraint that at least one would be a shift CD and another is a moving CD. Finally, the random case is like the mixed case but without any constrain on the CD type. For each case, we used $n = 1000$ datasets.

6.2. Sensitivity analysis

In order to evaluate the proposed model's performance in different settings, we explore the performance of both the proposed model and its improved version on different concept drift rates, dataset sizes, and dataset's complexity. For the dataset's complexity, we adopted the metric proposed by Shmuel et al. (2024b) which associated the dataset's complexity with its non-linearity measured by using the $1-R^2$ value obtained from a linear regression model trained on the dataset.

6.2.1. Drift rate

Table 3 summarizes the performance, in terms of Eq. (2), as the mean of $n = 1000$ cases for each drift rate. One can notice an average decrease in the performance of the model as the drift rate increases. Nonetheless, for a drift rate of 0.2, the performance of both models slightly increases as the shift CD is clearer and easier to detect by the CD algorithms in the ensemble. Importantly, we allow between 2 and 10 CDs for each sample so that all of them have the same drift rate.

6.2.2. Dataset size

Table 5 summarizes the performance as the mean of $n = 1000$ cases for datasets with different initial sizes and growth rates. The growth rate is the number of new samples added to the dataset in each step in time. For this analysis, we used the random CD case (see Section 6.1). The analysis shows that for datasets with relatively large initial sizes ($>10^4$), the performance over the growth rate is more stable for both models. However, for relatively small initial sizes ($<10^3$), the growth rate has the dominant effect on the model's performance. Overall, a tendency for more data is more beneficial for the proposed model.

6.2.3. Dataset's complexity

Table 5 summarizes the performance as the mean of $n = 1000$ cases for datasets with different complexity levels. For this analysis, we used the random CD case (see Section 6.1). This analysis shows that while more "complex" dataset has lower results in absolute terms due to the poorer performance of ML in general, the proposed model is only slightly negatively affected by the dataset's complexity.

6.2.4. AutoML library usage

Table 6 summarizes the performance as the mean of $n = 1000$ cases where the improved proposed model is utilized with different AutoML libraries. For this analysis, we used the random CD case (see Section 6.1). One can notice that all four examined libraries produced similar results, with TPOT and AutoGluon being the best and worst, respectively, on average.

6.2.5. Computational time

For all experiments, we utilized a machine equipped with an Intel Core i9-13900K processor, 64 GB of RAM, and an NVIDIA RTX 4090 GPU. The computational time for training and evaluating our proposed model, as well as its improved version, is presented in Table 7. To contextualize these results, we also include the computational time required by the TPOT model for the same datasets and experimental settings.

7. Discussion

In this study, we investigated the usage of ensemble ML models to handle CD in datasets that increase over time. In particular, we proposed a novel instance of the GA approach for an ML pipeline with CD detection algorithm ensemble. Using this structure, the overall model is more robust for different CD events. Moreover, we show that one can further improve the proposed algorithm by integrating existing off-the-shelf automatic ML approaches. Overall, the proposed model allows the use of existing ML and CD algorithms to bolster the resilience of ML models in the face of CD in realistic scenarios.

To be exact, the two-level ML model proposed in this study holds promise in addressing the dynamic challenges posed by CD by introducing a global ML model with a CD detector operating as an ensemble model for a population of ML pipeline models, which also can be adopted by an adjusted CD detection algorithm. Hence, this study takes a novel approach compared to the one-model-to-rule-them-all approach currently governing the field (Zliobaite, 2010; Hu et al., 2019). Simply put, the ensemble structure utilized by the proposed model allows individual ML to autonomously adapt to subset-specific changes while feeding the prediction of each model and how well it is operating, as indicated by its adjusted CD detector, showcasing the adaptability of GAs in governing diverse data subsets.

Table 3
Sensitivity analysis for the proposed model and its improved version in terms of drift rate.

Model\Drift rate	0.01	0.025	0.05	0.075	0.1	0.15	0.2
Proposed	0.81 (0.06)	0.80 (0.06)	0.79 (0.06)	0.77 (0.07)	0.73 (0.07)	0.74 (0.07)	0.75 (0.06)
Proposed improved	0.81 (0.05)	0.81 (0.05)	0.80 (0.06)	0.77 (0.06)	0.75 (0.06)	0.76 (0.06)	0.77 (0.06)

Table 4
Sensitivity analysis for the proposed model and its improved version for different initial sizes and growth rates.

Model	Initial size\ growth rate	10^0	10^1	10^2	10^3
Proposed	10^2	0.42 (0.18)	0.44 (0.16)	0.50 (0.14)	0.57 (0.13)
	10^3	0.49 (0.17)	0.49 (0.15)	0.53 (0.14)	0.57 (0.13)
	10^4	0.62 (0.12)	0.62 (0.12)	0.62 (0.12)	0.63 (0.11)
	10^5	0.64 (0.11)	0.64 (0.11)	0.64 (0.11)	0.64 (0.11)
Proposed improved	10^2	0.44 (0.19)	0.45 (0.18)	0.49 (0.17)	0.59 (0.12)
	10^3	0.49 (0.17)	0.49 (0.17)	0.50 (0.16)	0.59 (0.12)
	10^4	0.65 (0.12)	0.65 (0.12)	0.65 (0.11)	0.67 (0.10)
	10^5	0.67 (0.10)	0.67 (0.10)	0.67 (0.10)	0.67 (0.10)

Table 5
Sensitivity analysis for the proposed model and its improved version in terms of the dataset's complexity. The *absolute* comparison does not account for the ML pipeline performance in the substance of CD while the *relative* divides the result of Eq. (2) by $\psi(M, D(0))$.

Model	Comparison	0.1	0.3	0.5	0.7	0.9
Proposed	Absolute	0.71 (0.11)	0.78 (0.07)	0.81 (0.06)	0.80 (0.05)	0.79 (0.06)
	Relative	0.89 (0.04)	0.88 (0.04)	0.90 (0.03)	0.89 (0.04)	0.90 (0.04)
Proposed improved	Absolute	0.73 (0.09)	0.81 (0.05)	0.84 (0.05)	0.82 (0.06)	0.81 (0.05)
	Relative	0.92 (0.03)	0.90 (0.04)	0.90 (0.04)	0.90 (0.04)	0.91 (0.04)

Table 6
Sensitivity analysis for the proposed model and its improved version in terms of drift rate.

Model\AutoML library	TPOT	AutoSklearn	AutoGluon	PyCaret
Proposed improved	0.67 (0.10)	0.65 (0.09)	0.64 (0.11)	0.64 (0.09)

Table 7
Computational time comparison between TPOT, the proposed model, and its improved version. The results are shown as the mean with the standard deviation in brackets of $n = 100$ random synthetic datasets.

Model	Training time (minutes)	Inference time (minutes)
TPOT	52.4 (13.1)	0.074 (0.022)
Proposed	94.1 (32.5)	0.409 (0.158)
Proposed Improved	70.2 (20.8)	0.285 (0.094)

Indeed, Table 2 shows that the proposed solution provides a more robust solution, on average, for a large number of datasets compared to a single ML pipeline with a CD algorithm obtained using an AutoML library (such as TPOT) and brute-force of multiple (12) CD algorithms. For the most difficult and realistic settings where the number, as well as nature or the CD, are unknown (i.e., the *random* case) the proposed model improves the performance of the ML model by up to 0.13 while also reducing the diversity in the results between datasets from 0.16 to 0.11, indicating a more robust and consistent results across different datasets.

Moreover, we explore the performance of the proposed model over different settings. First, Table 3 shows that the proposed model performs better from moving CD rather than shifting CD. This outcome aligns with the behavior of other solutions (Khamassi et al., 2018; Yu et al., 2022). Second, Table 4 reveals that the proposed model performs worse on small-size datasets, which is also a well-known phenomenon in the realm of ML (Qi and Luo, 2022; Raissi and Karniadakis, 2018). However, once enough data is available, the proposed model performs similarly. To this end, Table 5 continues the same line where more "complex" datasets resulted in worse performance in absolute terms as the underline ML pipeline models are performing worse. Nonetheless, in relative terms, the complexity of the dataset does not play much of a role in the context of handling CD. Finally, Table 6 shows that the improved version of the proposed model is only slightly affected by the autoML library used, at least from the popular autoML libraries currently available, and one can choose its preferred autoML library. Notably, the proposed model is more computationally expensive compared to other AutoML methods based on GA, such as TPOT, as indicated by Table 7 while in the same order of magnitude.

This research is not without limitations. First, the proposed solution is evaluated on synthetic data due to the challenge and resources required to find real-world cases of CD in large numbers and for a wide range of CD behaviors. Hence, the proposed results should be taken with caution, and future work should re-evaluate the proposed method using a large number of real-world CD datasets. With this in mind, future work should consider domain-specific optimizations, using real-world data, to enhance its generalizability and robustness. Second, the proposed method provides each ML model in the population a subset of the dataset, $D(t)$, which is continuous, ignoring the more generic cases where one can union several continuous subsets of $D(t)$ which can improve the performance the ML model alone and the entire performance, as a whole. One can investigate the contribution of such an extension to the proposed method's performance. Third, GA in general, and for the discussed case, in particular, often requires tuning several hyperparameters, such as population size, crossover rate, and mutation rate to achieve their optimal performance (Angelova and Pencheva, 2011; Mosayebi and Sodhi, 2020). In this study, we chose such hyperparameter values through a manual trial and error approach, which probably does not result in the optimal values. This approach is chosen as performing an exhaustive hyperparameter search would require substantial computational resources and despite that, the proposed method outperforms the baseline models, which indicates it is relatively robust for the GA hyperparameter values. Future studies may explore the sensitivity of the proposed GA-based approach to variations in these parameters and find the optimal hyperparameter values with respect to assumptions on the CD dynamics or the data itself. Notably, even if such a search is conducted, the identified hyperparameters may not generalize well across different scenarios. Fourth, the presented experiments focused on regression tasks, ignoring classification tasks, where CD are also commonly presented, such as in spam email detection (Henke et al., 2021) and fraud detection (Adebayo et al., 2023). Extending the evaluation to classification tasks can shed more light on

the performance of the proposed model in a wider context. Finally, following a more general trend (Kronberger et al., 2022; Wu et al., 2020; Pan and Shen, 2017; Liu et al., 2008; Best et al., 2009), one can reduce the search space of the proposed task and therefore improve the proposed algorithm by introducing knowledge about which ML models are appropriate to which types of CD and the best matching between an ML model and a CD detector.

This study marks a significant stride in fortifying ML models against the persistent challenges posed by CD through the approach of a two-level ensemble of ML models working together with the CD application of genetic algorithms GAs. The adaptability demonstrated, particularly in discrete spaces where traditional optimization methods face limitations, highlights the promising role of GAs in addressing the nuanced demands of evolving data distributions. While the research outcomes are encouraging, avenues for refinement and future exploration are recognized. Extending the generalizability across diverse domains, conducting a more comprehensive sensitivity analysis, and optimizing computational complexity for broader applicability should be key considerations. In essence, this research contributes to the ongoing discourse on adaptive learning systems, leveraging GAs in an ensemble ML context to navigate the challenges presented by dynamic data landscapes.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of competing interest

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

Data availability

Data will be made available on request.

References

Adebayo, O.S., Favour-Bethy, T.A., Otasowie, O., Okunola, O.A., 2023. Comparative review of credit card fraud detection using machine learning and concept drift techniques. Int. J. Comput. Sci. Mob. Comput. 12 (7), 24–48.
Agrahari, S., Singh, A.K., 2022. Concept drift detection in data stream mining : A literature review. J. King Saud Univ. - Comput. Inf. Sci. 34 (10), 9523–9540.
Alhijawi, B., Awajan, A., 2024. Genetic algorithms: Theory, genetic operators, solutions, and applications. Evol. Intell. 17 (3), 1245–1256.
Ali, M., 2020. PyCaret: An open source, low-code machine learning library in Python. URL https://www.pycaret.org, PyCaret version 1.0.
Alibrahim, H., Ludwig, S.A., 2021. Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization. In: 2021 IEEE Congress on Evolutionary Computation. CEC, IEEE, pp. 1551–1559.
Angelova, M., Pencheva, T., 2011. Tuning genetic algorithm parameters to improve convergence time. Int. J. Chem. Eng. 2011, 646917.
Ariel, A., Lazebnik, T., Shami, L., 2023. Microfounded tax revenue forecast model with heterogeneous population and genetic algorithm approach. Comput. Econ..
Bach, S.H., Maloof, M.A., 2008. Paired learners for concept drift. In: 2008 Eighth IEEE International Conference on Data Mining. pp. 23–32.
Baratchi, M., Wang, C., Limmer, S., van Rijn, J.N., Hoos, H., Bäck, T., Olhofer, M., 2024. Automated machine learning: past, present and future. Artif. Intell. Rev. 57 (5), 122.
Bayram, F., Ahmed, B.S., Kassler, A., 2022. From concept drift to model degradation: An overview on performance-aware drift detectors. Knowl.-Based Syst. 245, 108632.
Berger, V.W., Zhou, Y., 2014. Kolmogorov–Smirnov test: Overview. In: Wiley StatsRef: Statistics Reference Online. John Wiley & Sons, Ltd, ISBN: 9781118445112.
Best, A., Terpstra, J.L., Moor, G., Riley, B., Norman, C.D., Glasgow, R.E., 2009. Building knowledge integration systems for evidence-informed decisions. J. Heal. Organ. Manag. 23 (6), 627–641.
Bifet, A., G., R., 2007. Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM International Conference on Data Mining. pp. 443–448.
Bo, Z.W., Hua, L.Z., Yu, Z.G., 2006. Optimization of process route by genetic algorithms. Robot. Comput.-Integr. Manuf. 22, 180–188.
Bo, L., Rein, L., 2005. Comparison of the Luus–Jaakola optimization procedure and the genetic algorithm. Eng. Optim. 37 (4), 381–396.
Breiman, L., 2001. Random forests. Mach. Learn. 45, 5–32.
Brzezinski, D., Stefanowski, J., 2014. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. IEEE Trans. Neural Netw. Learn. Syst. 25 (1), 81–94.
Buchgraber, T., Shutin, D., Poor, H.V., 2011. A sliding-window online fast variational sparse Bayesian learning algorithm. In: 2011 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, pp. 2128–2131.
Céspedes-Sisniega, J., López-García, A., 2022. Frouros: A python library for drift detection in machine learning systems. ArXiv.
Chauhan, K., Jani, S., Thakkar, D., Dave, R., Bhatia, J., Tanwar, S., Obaidat, M.S., 2020. Automated machine learning: The new wave of machine learning. In: 2020 2nd International Conference on Innovative Mechanisms for Industry Applications. ICIMIA, pp. 205–212.
Chen, R., Yang, B., Li, S., Wang, S., 2020. A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. Comput. Ind. Eng. 149, 106778.
Chowdhury, P., Paul, S.K., Kaisar, S., Moktadir, A., 2021. COVID-19 pandemic related supply chain studies: A systematic review. Transp. Res. Part E: Logist. Transp. Rev. 148, 102271.
Çubukçu, H.C., Topcu, D.I., Yenice, S., 2024. Machine learning-based clinical decision support using laboratory data. Clin. Chem. Lab. Med. (CCLM) 62 (5), 793–823.
Dasarathy, B.V., Sheela, B.V., 1979. A composite classifier system design: concepts and methodology. Proc. IEEE 67 (5), 708–713.
Davis, L., 1985. Applying adaptive algorithms to epistatic domains. In: Proceedings of the International Joint Conference on Artificial Intelligence. pp. 162–164.
De Jong, K., 1988. Learning with genetic algorithms: An overview. Mach. Learn. 3, 121–138.
Demsar, J., Bosnic, Z., 2018. Detecting concept drift in data streams using model explanation. Expert Syst. Appl. 92, 546–559.
Dong, X., Yu, Z., Cao, W., Shi, Y., Ma, Q., 2020. A survey on ensemble learning. Front. Comput. Sci. 14 (2), 251–258.
Drake, A.E., Marks, R., 2002. Genetic algorithms in economics and finance: Forecasting stock market prices and foreign exchange—A review. In: Chen, S.H. (Ed.), Genetic Algorithms and Genetic Programming in Computational Finance. Springer, Boston, MA, pp. 29–54.
Drori, I., Krishnamurthy, Y., Rampin, R., de Paula Lourenco, R., Ono, J.P., Cho, K., Silva, C., Freire, J., 2021. AlphaD3M: Machine learning pipeline synthesis. ArXiv.
Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., Smola, A., 2020. AutoGluon-tabular: Robust and accurate AutoML for structured data. ArXiv.
Fdez-Riverola, F., Iglesias, E.L., Diaz, F., Mendez, J.R., Corchado, J.M., 2007. Applying lazy learning algorithms to tackle concept drift in spam filtering. Expert Syst. Appl. 33 (1), 36–48.
Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., Hutter, F., 2020. Auto-sklearn 2.0: Hands-free AutoML via meta-learning. ArXiv.
Feurer, M., Klevin, A., Eggensperger, K., Springenberg, J.T., Blum, M., Hutter, F., 2019. Auto-sklearn: Efficient and robust automated machine learning. Automated Machine Learning: Methods, Systems, Challenges.
Gama, J.M., Zliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A., 2014. A survey on concept drift adaptation. ACM Comput. Surv. 46 (4), 1–37.
Ghaheri, A., Shoar, S., Naderan, M., Hoseini, S.S., 2005. The applications of genetic algorithms in medicine. Oman Med. J. 30 (6), 406–416.
Ghomeshi, H., Gaber, M.M., Kovalchuk, Y., 2019. EACD: evolutionary adaptation to concept drifts in data streams. Data Min. Knowl. Discov. 33, 663–694.
Goncalves, P.M., de Carvalho Santos, S.G.T., Barros, R.S.M., Vieira, D.C.L., 2014. A comparative study on concept drift detectors. Expert Syst. Appl. 41 (18), 8144–8156.
Harel, M., Mannor, S., El-Yaniv, R., Crammer, K., 2014. Concept drift detection through resampling. In: Xing, E.P., Jebara, T. (Eds.), Proceedings of the 31st International Conference on Machine Learning. Vol. 32, (2), pp. 1009–1017.
Hassanat, A.B.A., Alkafaween, E., 2017. On enhancing genetic algorithms using new crossovers. Int. J. Comput. Appl. Technol. 55 (3).
Henke, M., dos Santos, E.M., Souto, E., Santin, A.O., 2021. Spam detection based on feature evolution to deal with concept drift. J. Univers. Comput. Sci. 27 (4), 364–386.
Heppenstall, A.J., Evans, A.J., Birkin, M.H., 2007. Genetic algorithm optimisation of an agent-based model for simulating a retail market. Environ. Plan. B: Urban Anal. City Sci..
Herrmann, J.W., 1999. A genetic algorithm for minimax optimization problems. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). 2, pp. 1099–1103.
Holland, J.H., 1992. Genetic algorithms. Sci. Am. 267 (1), 66–73.
Hu, H., Kantardzic, M., Sethi, T.S., 2019. No free lunch theorem for concept drift detection in streaming data classification: A review. WIREs Data Min. Knowl. Discov. 10 (2), e1327.
Huang, Y., Liu, C., Li, W., Liu, X., Wu, J.H., Ma, F., 2024. A multifunctional metastructure with energy dissipation and low-frequency sound-absorption optimized for decoupling by genetic algorithm. Thin-Walled Struct. 199, 111815.

Huang, G.-B., Zhu, Q.-Y., Siew, C.-Q., 2006. Extreme learning machine: Theory and applications. Neurocomputing 70 (1), 489–501.

Iwashita, A.S., Papa, J.P., 2019. An overview on concept drift learning. IEEE Access 7, 1532–1547.

Joo, C., Lee, J., Lim, J., Kim, J., Cho, H., 2024. A genetic algorithm-based optimal selection and blending ratio of plastic waste for maximizing economic potential. Process. Saf. Environ. Prot. 186, 715–727.

Kammerer, L., Kronberger, G., Burlacu, B., Winkler, S.M., Kommenda, M., Affenzeller, M., 2020. Symbolic regression by exhaustive search: reducing the search space using syntactical constraints and efficient semantic structure deduplication. In: Genetic Programming Theory and Practice XVII. Springer, pp. 79–99.

Kaya, Y., Uyar, M., R., T., 2011. A novel crossover operator for genetic algorithms: ring crossover. ArXiv.

Keren, L.S., Liberzon, A., Lazebnik, T., 2023. A computational framework for physics-informed symbolic regression with straightforward integration of domain knowledge. Sci. Rep. 13, 1249.

Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., Ghedira, K., 2018. Discussion and review on evolving data streams and concept drift adapting. Evol. Syst. 9, 1–23.

Kolter, J.Z., Maloof, M.A., 2007. Dynamic weighted majority: An ensemble method for drifting concepts. J. Mach. Learn. Res. 8, 2755–2790.

Kou, A.-J., Huang, X., Sun, W.-X., 2024. Research on concept drift algorithm based on evolutionary computation. Discov. Appl. Sci. 6 (8), 424.

Kronberger, G., de França, F.O., Burlacu, B., Haider, C., Kommenda, M., 2022. Shape-constrained symbolic regression—Improving extrapolation with prior knowledge. Evol. Comput. 30 (1), 75–98.

Krongauz, D.L., Lazebnik, T., 2023. Collective evolution learning model for vision-based collective motion with collision avoidance. PLoS One 18 (5), e0270318.

Kumar, M., Husain, M., Upreti, N., Gupta, D., 2010. Genetic algorithm: Review and application. Int. J. Inf. Technol. Knowl. Manag. 2 (2), 451–454.

Kuptametee, C., Michalopoulou, Z.-H., Aunsri, N., 2024. A review of efficient applications of genetic algorithms to improve particle filtering optimization problems. Measurement 224, 113952.

Kuranga, C., Pillay, N., 2021. Genetic programming-based regression for temporal data. Genet. Program. Evol. Mach. 22, 297–324.

Lazebnik, T., 2022. Cell-level spatio-temporal model for a bacillus calmette–guérin-based immunotherapy treatment protocol of superficial bladder cancer. Cells 11 (15).

Lazebnik, T., 2023. Data-driven hospitals staff and resources allocation using agent-based simulation and deep reinforcement learning. Eng. Appl. Artif. Intell. 126, 106783.

Lazebnik, T., Bahouth, Z., Bunimovich-Mendrazitsky, S., Halachmi, S., 2022. Predicting acute kidney injury following open partial nephrectomy treatment using SAT-pruned explainable machine learning model. BMC Med. Inform. Decis. Mak. 22, 133.

Lazebnik, T., Bunimovich-Mendrazitsky, S., 2021. The signature features of COVID-19 pandemic in a hybrid mathematical model—Implications for optimal work–school lockdown policy. Adv. Theory Simul. 4 (5), e2000298.

Lazebnik, T., Fleischer, T., Yaniv-Rosenfeld, A., 2023. Benchmarking biologically-inspired automatic machine learning for economic tasks. Sustainability 15 (14), 11232.

Lazebnik, T., Rosenfeld, A., 2023. FSPL: A meta–learning approach for a filter and embedded feature selection pipeline. Int. J. Appl. Math. Comput. Sci. 33 (1), 103–115.

Liu, O.L., Lee, H.-S., Hofstetter, C., Linn, M.C., 2008. Assessing knowledge integration in science: Construct, measures, and evidence. Educ. Assess. 13 (1), 33–55.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G., 2019. Learning under concept drift: A review. IEEE Trans. Knowl. Data Eng. 31 (12), 2346–2363.

Lu, N., Lu, J., Zhang, G., Lopez de Mantaras, R., 2016. A concept drift-tolerant case-base editing technique. Artificial Intelligence 230, 108–133.

Luo, X., Chang, X., Ban, X., 2016. Regression and classification using extreme learning machine based on L1-norm and L2-norm. Neurocomputing 174, 179–186.

Madireddy, S., Balaprakash, P., Carns, P., Latham, R., Lockwood, G.K., Ross, R., Snyder, S., Wild, S.M., 2019. Adaptive learning for concept drift in application performance modeling. In: Proceedings of the 48th International Conference on Parallel Processing.

Maggi, F., Robertson, W., Kruegel, C., Vigna, G., 2009. Protecting a moving target: Addressing web application concept drift. In: Kirda, E., Jha, S., Balzarotti, D. (Eds.), Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, pp. 21–40.

Mishra, R., Bajpai, M.K., 2024. A novel multi-agent genetic algorithm for limited-view computed tomography. Expert Syst. Appl. 238, 122195.

Molino, P., Dudin, Y., Miryala, S.S., 2019. Ludwig: a type-based declarative deep learning toolbox. ArXiv.

Mosayebi, M., Sodhi, M., 2020. Tuning genetic algorithm parameters using design of experiments. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion. pp. 1937–1944.

Nisioti, E., Chatzidimitriou, K.C., Symeonidis, A.L., 2018. Predicting hyperparameters from meta-features in binary classification problems. In: ICML 2018 AutoML Workshop.

Oliveira, G.H.F.M., Cavalcante, R.C., Cabral, G.G., Minku, L.L., Oliveira, A.L.I., 2017. Time series forecasting in the presence of concept drift: A PSO-based approach. In: 2017 IEEE 29th International Conference on Tools with Artificial Intelligence. ICTAI, pp. 239–246.

Olson, R.S., Moore, J.H., 2016. TPOT: A tree-based pipeline optimization tool for automating machine learning. In: Workshop on Automatic Machine Learning. PMLR, pp. 66–74.

Oren, A., Turkcu, J.D., Meller, S., Lazebnik, T., Wiegel, P., Mach, R., Volk, H.A., Zamansky, A., 2022. BrachySound: machine learning based assessment of respiratory sounds in dogs. Sci. Rep. 4, 25–32.

Padmalatha, E., Reddy, C.R.K., Rani, B.P., 2015. Classification of concept-drifting data streams using optimized genetic algorithm. Int. J. Comput. Appl. 125 (15).

Pan, X., Shen, H.B., 2017. RNA-protein binding motifs mining with a new hybrid deep learning based cross-domain knowledge integration approach. BMC Bioinformatics 18, 136.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al., 2011. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

Pinto, F., Cerqueira, V., Soares, C., Mendes-Moreira, J., 2017. Autobagging: Learning to rank bagging workflows with metalearning. ArXiv.

Pujawan, I.N., Bah, A.U., 2022. Supply chains under COVID-19 disruptions: literature review and research agenda. Supply Chain Forum: Int. J. 23 (1), 81–95.

Qi, G.-J., Luo, J., 2022. Small data challenges in big data era: A survey of recent progress on unsupervised and semi-supervised methods. IEEE Trans. Pattern Anal. Mach. Intell. 44 (4), 2168–2187.

Rahman, N.A.A., Ahmi, A., Jraisat, L., Upadhyay, A., 2022. Examining the trend of humanitarian supply chain studies: pre, during and post COVID-19 pandemic. J. Humanit. Logist. Supply Chain Manag. 12 (4), 594–617.

Raissi, M., Karniadakis, G.E., 2018. Hidden physics models: Machine learning of nonlinear partial differential equations. J. Comput. Phys. 357, 125–141.

Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378, 686–707.

Routledge, B.R., 2001. Genetic algorithm learning to choose and use information. Macroecon. Dyn. 5 (2), 303–325.

Salehi, M., Bahreininejad, A., 2011. Optimization process planning using hybrid genetic algorithm and intelligent search for job shop machining. J. Intell. Manuf. 22 (4), 643–652.

Savchenko, E., Lazebnik, T., 2022. Computer aided functional style identification and correction in modern russian texts. J. Data, Inf. Manag. 4, 25–32.

Savchenko, E., Rosenfeld, A., Bunimovich-Mendrazitsky, S., 2023. Mathematical modeling of BCG-based bladder cancer treatment using socio-demographics. Sci. Rep. 13, 18754.

Schapire, R.E., 1990. The strength of weak learnability. Mach. Learn. 5 (2), 197–227.

Sehgal, A., La, H., Louis, S., Nguyen, H., 2019. Deep reinforcement learning using genetic algorithm for parameter optimization. In: 2019 Third IEEE International Conference on Robotic Computing. IRC, pp. 596–601.

Shami, L., Lazebnik, T., 2023. Implementing machine learning methods in estimating the size of the non-observed economy. Comput. Econ..

Shapiro, J., 2001. Genetic algorithms in machine learning. In: Paliouras, G., Karkaletsis, V., Spyropoulos, C.D. (Eds.), Machine Learning and Its Applications: Advanced Lectures. pp. 146–168.

Shmuel, A., Glickman, O., Lazebnik, T., 2024a. A comprehensive benchmark of machine and deep learning across diverse tabular datasets. ArXiv.

Shmuel, A., Glockman, O., Lazebnik, T., 2024b. Symbolic regression as a feature engineering method for machine and deep learning regression tasks. Mach. Learn.: Sci. Technol. 5.

Shyaa, M.A., Ibrahim, N.F., Zainol, Z., Abdullah, R., Anbar, M., Alzubaidi, L., 2024. Evolving cybersecurity frontiers: A comprehensive survey on concept drift and feature dynamics aware machine and deep learning in intrusion detection systems. Eng. Appl. Artif. Intell. 137, 109143.

Smith, M., Ciesielski, V., 2016. Adapting to concept drift with genetic programming for classifying streaming data. In: 2016 IEEE Congress on Evolutionary Computation. CEC, pp. 5026–5033.

Sohail, A., 2023. Genetic algorithms in the fields of artificial intelligence and data sciences. Ann. Data Sci. 10, 1007–1018.

Swain, P.H., Hauska, H., 1977. The decision tree classifier: Design and potential. IEEE Trans. Geosci. Electron. 15 (3), 142–147.

Tareq, M., Sundararajan, E.A., 2020. A new density-based method for clustering data stream using genetic algorithm. Technol. Rep. Kansai Univ. 62 (11), 6557–6572.

Tyagi, K., Kumar, D., Gupta, R., 2024. Application of genetic algorithms for medical diagnosis of diabetes mellitus. Int. J. Exp. Res. Rev. 37, 1–10.

Vivekanandan, P., Nedunchezhian, R., 2011. Mining data streams with concept drifts using genetic algorithm. Artif. Intell. Rev. 36, 163–178.

Wang, S., Minku, L.L., Ghezzi, D., Caltabiano, D., Tino, P., Yao, X., 2013. Concept drift detection for online class imbalance learning. In: The 2013 International Joint Conference on Neural Networks. IJCNN, pp. 1–10.

Wang, P., Yu, H., Jin, N., Davies, D., Woo, W.L., 2024. QuadCDD: A quadruple-based approach for understanding concept drift in data streams. Expert Syst. Appl. 238, 122114.

Worm, T., Chiu, K., 2013. Prioritized grammar enumeration: symbolic regression by dynamic programming. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation. pp. 1021–1028.

Wu, R., Fujita, Y., Soga, K., 2020. Integrating domain knowledge with deep learning models: An interpretable AI system for automatic work progress identification of NATM tunnels. Tunn. Undergr. Space Technol. 105, 103558.

Xia, S., Wei, M., Zhu, Y., Pu, Y., 2024. AI-driven intelligent financial analysis: Enhancing accuracy and efficiency in financial decision-making. J. Econ. Theory Bus. Manag. 1 (5), 1–11.

Xiang, Q., Zi, L., Cong, X., Wang, Y., 2023. Concept drift adaptation methods under the deep learning framework: A literature review. Appl. Sci. 13 (11).

Yan, Z., Hongle, D., Gang, K., Lin, Z., Chen, Y.-C., 2022. Dynamic weighted selective ensemble learning algorithm for imbalanced data streams. J. Supercomput. 78 (4), 5394–5419.

Yang, L., McClean, S., Donnelly, M., Burke, K., Khan, K., 2022. Detecting and responding to concept drift in business processes. Algorithms 15 (5).

Yao, Q., Wang, M., Chen, Y., Dai, W., Li, Y.-F., Tu, W.W., Yang, Q., Yu, Y., 2019. Taking human out of learning applications: A survey on automated machine learning. ArXiv.

Yu, E., Song, Y., Zhang, G., Lu, J., 2022. Learn-to-adapt: Concept drift adaptation for hybrid multiple streams. Neurocomputing 496, 121–130.

Žegklitz, J., Pošík, P., 2021. Benchmarking state-of-the-art symbolic regression algorithms. Genet. Program. Evol. Mach. 22 (1), 5–33.

Zhao, J., Xu, M., 2013. Fuel economy optimization of an Atkinson cycle engine using genetic algorithm. Appl. Energy 105, 335–348.

Zhao, J., Xu, M., Li, M., Wang, B., Liu, S., 2012. Design and optimization of an atkinson cycle engine with the artificial neural network method. Appl. Energy 92, 492–502.

Zliobaite, I., 2010. Learning under concept drift: an overview. ArXiv.

Žliobaite, I., Pechenizkiy, M., Gama, J., 2016. Big Data Analysis: New Algorithms for a New Society. vol. 16, Springer, pp. 91–114.

Zliobaite, I., Pechenizkiy, M., Gama, J., 2016. An overview of concept drift applications. In: Japkowicz, N., Stefanowski, J. (Eds.), Big Data Analysis: New Algorithms for a New Society. Springer International Publishing, pp. 91–114.