scientific reports

Check for updates

OPEN

Intrusion detection using metaheuristic optimization within IoT/IIoT systems and software of autonomous vehicles

Pavle Dakic[1,2,9], Miodrag Zivkovic[1,9], Luka Jovanovic[1,9], Nebojsa Bacanin[1,3,4,9]✉, Milos Antonijevic[1,9], Jelena Kaljevic[5,9] & Vladimir Simic[6,7,8,9]

The integration of IoT systems into automotive vehicles has raised concerns associated with intrusion detection within these systems. Vehicles equipped with a controller area network (CAN) control several systems within a vehicle where disruptions in function can lead to significant malfunctions, injuries, and even loss of life. Detecting disruption is a primary concern as vehicles move to higher degrees of autonomy and the possibility of self-driving is explored. Tackling cyber-security challenges within CAN is essential to improve vehicle and road safety. Standard differences between different manufacturers make the implementation of a discreet system difficult; therefore, data-driven techniques are needed to tackle the ever-evolving landscape of cyber security within the automotive field. This paper examines the possibility of using machine learning classifiers to identify cyber assaults in CAN systems. To achieve applicability, we cover two classifiers: extreme gradient boost and K-nearest neighbor algorithms. However, as their performance hinges on proper parameter selection, a modified metaheuristic optimizer is introduced as well to tackle parameter optimization. The proposed approach is tested on a publicly available dataset with the best-performing models exceeding 89% accuracy. Optimizer outcomes have undergone rigorous statistical analysis, and the best-performing models were subjected to analysis using explainable artificial intelligence techniques to determine feature impacts on the best-performing model.

Keywords Autonomous vehicles, AI and ML, Metaheuristics optimization, IoT/IIoT systems

Attack and intrusion detection processes within internet of things (IoT) systems require an understanding of all interrelated aspects that can be abused in the event of an attack. It is vital to comprehend the complex interlay of different components along with their dependencies within IoT systems. These systems are comprised of numerous connected devices, sensors, networks and services, each with a set of its own potential vulnerabilities. This specifically pertains to existing approaches and forecasting new harmful assaults using machine learning and various natural algorithms for optimization and categorization. Machine learning (ML) and the application of appropriate models can achieve the prevention and early detection of various forms of attacks[1]. To spot patterns and anomalies that indicate prospective assaults, we can use algorithms that have been trained on vast datasets of normal and abnormal activity. Algorithms can consequently be employed to perform a classification of fresh data inputs and predict their likelihood of being an attack.

The integration of IoT technologies and metaheuristic optimization is important for self-driving vehicles. IoT networks, with their web of interconnected devices and sensors, are capable of collecting real-time environmental

[1]Faculty of Informatics and Computing, Singidunum University, Belgrade 11000, Serbia. [2] Faculty of Informatics and Information Technologies, Institute of Informatics, Information Systems and Software Engineering, Slovak University of Technology in Bratislava, 84 216 Bratislava, Slovakia. [3]Department of Mathematics, Saveetha School of Engineering, SIMATS, Kuthambakkam, Tamilnadu 602105, India. [4]MEU Research Unit, Middle East University, Amman, Jordan. [5]Faculty of Health and Business Studies, Singidunum University, Valjevo 14000, Serbia. [6]Faculty of Transport and Traffic Engineering, University of Belgrade, Belgrade 11010, Serbia. [7] Department of Industrial Engineering and Management, College of Engineering, Yuan Ze University, Taoyuan City 320315, Taiwan. [8]Department of Computer Science and Engineering, College of Informatics, Korea University, Seoul 02841, Republic of Korea. [9]Pavle Dakic, Miodrag Zivkovic, Luka Jovanovic, Nebojsa Bacanin, Milos Antonijevic, Jelena Kaljevic, Vladimir Simic have contributed equally to this work. ✉email: nbacanin@singidunum.ac.rs

data. Metaheuristic algorithms can process and evaluate this information, allowing vehicles to make independent and informed decisions. This connection allows algorithms to process and evaluate information more quickly, opening up a new world of possibilities for self-driving cars through the use of real-time data. Vehicles in this scenario can adapt to constantly changing road conditions, traffic patterns, and environmental considerations. Identifying optimal routes can guarantee effective navigation and improve overall performance and passenger safety. This synergy demonstrates how metaheuristic optimization can increase vehicle efficiency and safety.

The availability of artificial intelligence (AI) models has had an unprecedented impact on people, and its applications can now be used in the development and operation of autonomous systems. Some researchers aimed to recognize the issues related to the use and impact of regenerated AI-based decision-making systems and provide a collection of study recommendations for information technology scientists[2]. Proper implementation of models is not without challenges. Chief among these is the proper choice of control parameters and the attainment and properly preparation of the data. The selection of acceptable parameters is considered an NP-hard challenge due to the large search space[3]. Therefore, this research examines the capability of metaheuristics to tackle this challenge with realistic computational resources and within acceptable optimization time periods.

An approach to optimizing and improving attack detection is metaheuristic optimization. Metaheuristics can be used to find optimal parameters for ML models, such as feature selection, the number of hidden layers in a neural network, and learning speed. This can lead to improved accuracy and a reduced false-positive rate, leading to a more efficient and effective attack detection system. A more effective and efficient attack detection system can be developed, leading to improved security and reliability of IoT systems[4,5]. It is possible to increase the percentage of successful attack prevention by using predictive models and metaheuristic optimization in IoT systems. However, it is vital to highlight that the complete removal of all attacks is not realistically attainable due to the ever-changing nature of security threats.

In the world involving autonomous vehicles, IoT is enabling critical communication and data flow by linking vehicles to infrastructure, other vehicles, and cloud networks. As such, this allows for real-time decisions and heightened awareness. With its industrial precision that integrates autonomous vehicles into complex systems such as traffic control and logistics, we can see their significance as the backbone of the whole system[6–8]. Since these technologies ensure the safety of autonomous vehicles through continuous monitoring, they enable predictive maintenance and traffic optimization. By providing the necessary infrastructure for intelligent and efficient operation, we create a foundation for the future of autonomous transportation.

As part of this research, combining machine learning with metaheuristic optimization can improve the accuracy and efficiency of attack detection systems, increasing the possibility of successful attack prevention. Two classification methods are explored, the K-nearest neighbors (KNN)[9] and extreme gradient boosting (XGBoost)[10]. As the performance of classifiers is tightly coupled with proper hyperparameter selection, a modified version of a well-known particle swarm optimization (PSO)[11] metaheuristic optimizer is introduced in this work to meet the optimization needs. With respect to the no free lunch (NFL) theorem stated by Wolpert[12], explaining that a singular optimization algorithm capable of attaining superior scores for every optimization challenge does not exist, while small-scale simulations were conducted first, where PSO obtained promising outcomes. Moreover, another reason for choosing PSO was that its performance was not evaluated in this domain. Consequently, it was chosen for further improvements before executing the main experiments presented within this manuscript.

For our research, we employed a can-dataset[13,14], which comprises samples of both attack-free (normal) traffic and incidences of various forms of assaults. The results of the proposed modified PSO were compared with the baseline PSO, together with several cutting-edge metaheuristic algorithms included in the comparative analysis.

The most relevant scientific contributions provided in this research can be summarized in the following:

- A novel approach for vehicle IoT intrusion detection enhanced by ML-optimized metaheuristic optimization approaches.
- An introduction of a modified metaheuristic optimizer that improves on some of the observed drawbacks of the original classic optimizer
- The interpretation of the acquired outcomes utilizing explainable AI techniques to have a better grasp of the feature influences on automotive securityThe reminder of the work is structured as per the following sections: Section "Related works" discusses related works that helped motivate this work as well as the observed literature gaps. The introduced methodology is discussed in section "Methods". The experiments used to evaluate the approach are described in section "Experimental setup" followed by the discussion and outcomes in section "Outcomes". The work concludes in section "Conclusions", which also discusses the limitations of the completed research and future work ideas.

Related works

As part of the literature analysis, we examine the processes of integrating multiple systems in autonomous cars, which pose substantial issues in maintaining stable and secure communication and transmission. Ensuring this is crucial in modern systems that require real-time decision-making, but still a complicated challenge in covering all the aspects.

Furthermore, growing IoT networks in dense metropolitan settings is difficult because there are gaps in our understanding of the long-term impacts of widespread IoT adoption, notably infrastructure needs in communication modems inside autonomous vehicle networks. The communication between vehicles is vital even during parking, where it can assist in finding free car parking space[15]. Recently, the Social Internet of Vehicles (SIoV) continues to be increasingly used in transportation, and due to the vast volume of data that malevolent users can readily intercept, secure communication is necessary[16]. The absence of common communication protocols across several systems impedes seamless interoperability. These problems highlight the need for more

targeted and advanced research, as well as tactics that do not have an impact on system performance. To make it easier for us to review the literature, we have organized it into the following sections.

Strategic frameworks

According to research by Huang and Rust[17], it is important to create a strategic framework to use artificial intelligence (AI) to engage clients and improve services. They developed a three-tier framework for strategic marketing: mechanical AI to automate tasks, thinking AI to process data and make decisions, and emotional AI to analyze interactions and human emotions. In their paper, the authors, Tyukin et al.[18], suggest new strategies to make hidden attacks visible without affecting system performance, allowing desired outputs at specific triggers. In contrast, this article contends that individuals have the choice of whether or not to interact with AI explanations, demonstrating that in some circumstances, AI explanations can minimize overreliance. Another study by the authors, Vasconcelos et al.[19] formalize the strategic decision of weighing the costs and rewards of participating in an activity versus relying on AI.

The authors, Solmaz et al.[20] in their research propose a system that combines data from autonomous vehicles and IoT devices, addressing security and privacy concerns in dangerous IoT scenarios. Luntovskyy et al.[21] offer advanced approaches to optimize IoT performance, reliability, and scalability. discuss best practices and case studies for solving IoT network problems, noting that early SDN stages in IoT networks lead to low accuracy and high costs. Another research carried out by the authors, Kain et al.[22] introduced FDIRO (Fault Detection, Isolation, Recovery, and Optimization) for handling faults in autonomous driving, based on aerospace FDIR practices. In addition, academics, Suresh and Madhavu[23] present a self-adaptive, energy-efficient BAT algorithm for selecting features that employ metaheuristic optimization for neural networks.

Snort IDS and DDoS attacks in IoT networks

To address these issues, researchers Ujjan et al.[24] propose using flow-based sampling and adaptive polling, along with a Snort IDS and a deep learning model, to combat DDoS attacks in IoT networks. Other authors, Roopak et al.[25] present an IDS that combines deep learning and optimization techniques to detect DDoS attacks, and some authors Al-Haija and Zein-Sabatto[26] have tried to develop an intelligent system using convolutional neural networks (IoT-IDCS-CNN) to detect and classify cyber attacks in IoT networks. Finally, there is a paper in which the authors, Zekry et al.[27] propose utilizing ConvLSTM deep learning models to identify corrupted information streams in autonomous vehicle guidance and powertrain data.

Numerous recent works deal with the machine learning approaches in the intrusion detection in general IoT networks. Paper[28] employs XGBoost tuned by firefly algorithm to perform intrusion detection. XGBoost optimized by hybrid sine cosine algorithm was used proposed in paper[29], and applied for intrusion detection in healthcare IoT networks, with promising outcomes. Another interesting approach was taken by Salb et al.[30], who proposed a hybrid CNN and XGBoost structure for IoT security. CNN was designated to perform feature extraction, while XGBoost tuned by the metaheuristics algorithm was assigned the classification task. Musleh et al.[31] explored the capabilities of several feature extractors, including DenseNet and VGG-16, and multiple classifiers including random forest and support vector machine (SVM) for intrusion detection in IoT. Multilayer SVM model tuned by the mud ring optimization was proposed by Turukmane et al.[32], and it achieved satisfactory accuracy on several intrusion detection datasets.

Cyber-physical systems

To evaluate cyber-physical systems, researchers in this discipline try to develop an environment for testing that is compatible with test case creation and unexpected falsification approaches. This platform for testing needs has gathered 40 million opinions in ten different languages from millions of people across 233 countries and housing compounds[33,34]. The Moral Machine experiment describes the effects of this trial on a fully independent driving terrain, a driverless agent must provide and use perceived data with its neighboring instruments to navigate more safely[35]. In this environment, some journalists[36,37] propose an extended warning announcement result related to detecting the presence of the rambler in real-time, through the vehicle-to-vehicle messaging system[38]. The ability to perceive and understand the behavior of road users in the environment is crucial for self-driving vehicles to properly plan reliable responses[39,40]. Other research introduced a comprehensive foundational literacy system to train an independent and honest agent who provides direction to instruments[41,42].

Considering self-awareness from an agent-based perspective and machine perception

The research in the work covered by the authors, Solmaz et al.[20] offers a system that combines detector data from independent devices with IoT data derived from climbing biases. The study[43] describes a path to an independent construction of a system that supports not only the attention to tone, but also conventional verification. According to Cunneen[44], independent agents' perceptions and decisions are heavily influenced by their technological and ethical understanding. Focuses on topics such as HMI, machine perception, classification, and data protection, which are distinct from the MIT Moral Machine experiment's decision-making experiment.

The heavy reliance of AI algorithms on hyperparameter selection has led to the development of many optimizers. Additionally, proposals put forth by the "no free lunch" theorem by authors Wolpert and Macready[12] suggest that no single approach works well for all cases under all conditions. Therefore, many optimizers have been developed over the years to meet the specific needs of optimizations in various fields. Algorithms have taken inspiration from evolution, with a notable example being the genetic algorithm GA)[45]. Swarm inspired algorithms such as the artifical bee colony ABC[46] and firefly algorithm FA[47] algorithms are also popular among researchers for their ease of implementation and generally good performance. More abstract concepts have also served as inspiration, the notable examples being the botox optimization algorithm (BOA) covered by authors, Hubálovská et al.[48] and variable neighborhood search (VNS) from authors, Hansen et. al[49] algorithms.

Consequently, the abundance of metaheuristics algorithms led to their intensive application in different domains, including medicine[50,51], cloud computing[52,53], software testing[54,55] and economy[56,57]. Different machine learning models were successfully optimized by metaheuristics algorithms as well in the past, including long-short term memory and recurrent neural networks[58,59], CNNs[60], XGBoost[30], AdaBoost[61] and extreme learning machine (ELM)[62].

The literature study indicates that, while AI is an attractive use in cybersecurity and IoT systems, the combined use of AI for automotive has not yet been investigated. This study aims to use optimization metaheuristics to improve the categorization and detection of cyberattacks in the automotive area.

K-nearest neighbors

The K-nearest neighbors (KNN)[9] method is thought to be a reasonably simple yet highly successful supervised classification approach. It is a non-parametric classification algorithm, that for each data point necessitating to be classified retrieves its k neighbours. Majority voting between the neighborhood is utilized to decide the classification of that particular data point.

knn classifies new data by identifying the closest known points and assigning the most common label to them. To enhance prediction accuracy, XGBoost creates a series of decision trees that learn from each other's failures. This can be presented as a group of experts perfecting their responses after learning from each other's mistakes. The approaches outlined above are appropriate for anyone interested in the application and possibilities that can be realized in various combinations with machine learning. Both algorithms are popular in machine learning due to their performance, with KNN being easy and XGBoost providing powerful, fine-tuned results.

Training samples are arrays in a multidimensional feature space, each with its own class label. During the training process, the model only stores feature arrays and their corresponding classes. During the classification stage, an unlabeled array matrix (data point) will be classified by assigning the label that occurs the most frequently in k training samples closest to this data point. The most frequently used distance metric to determine the points in proximity is the Euclidean distance in the case of continuous parameters, while the Hamming distance is used for discrete variables. Other frequently used distance measures include Minkowski and Manhattan, as given in Eq. 1-3. The number k is a constant value specified by the user. This process is visualized in Fig. 1.

$$\text{Euclidean} \quad \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2} \tag{1}$$

$$\text{Manhattan} \quad \sum_{i=1}^{k} |x_i - y_i| \tag{2}$$

$$\text{Minkowski} \quad \left(\sum_{i=1}^{k} (|x_i - y_i|)^q \right)^{1/q} \tag{3}$$

The most suitable selection of k value depends on the underlying data being classified. Generally speaking, higher values of k will reduce the noise affecting the classification. However, this will also make the bounds

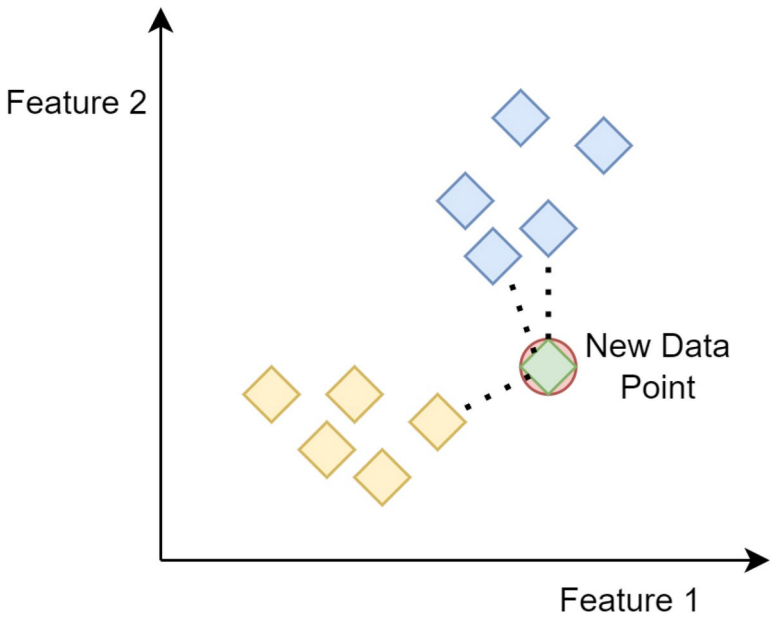

Figure 1. Visualization of KNN classification logic.

among classes far less distinguishable. The KNN hyperparameter tuning includes the selection of an appropriate value of *k*, accompanied by the choice of distance metrics and weights. This process can affect the accuracy of the models significantly, as poor choices of hyperparameters may degrade the performance of the method.

Tuning KNN involves selecting the best k value by testing different options, experimenting with various distance metrics like Euclidean or Manhattan, and scaling features so that each contributes equally. Based on some of the situations we discovered, it can be used to anticipate performance and detect undesired behaviors and achievements, while being heavily dependent on advanced machine learning (ML) methodologies for tuning. We might also attempt giving closer neighbors greater power to achieve specific goals[63,64].

Extreme gradient boosting

Extreme Gradient Boosting (XGBoost) is a remarkably effective technique for supervised ML tasks. The key reason for choosing this approach is its high performance in predictive analytics on massive data sets with numerous characteristics. This makes it perfect for real-time monitoring and security testing of automotive and industrial control systems. XGBoost processes increase efficiency, flexibility, and portability by using an efficient distributed gradient boost framework[65]. The key advantage of this algorithm is that it can help strengthen security measures by helping detect anomalies and protect against fraud and cybersecurity in a variety of industries. Its ability to detect suspicious activities proactively makes it a preferred solution for many problems. As a result, it can be applied to improve safety measures in enterprises, particularly financial institutions, technological companies, and, in the current case, the automotive industry. Furthermore, its compatibility with scattered systems enables scalable security evaluations, which are critical to staying ahead of evolving threats[10,65].

XGBoost's ability to handle complicated data patterns, optimize objective functions, and control model complexity makes it an important tool for industrial applications, particularly in CAN bus systems where real-time performance and precise forecasts are required. XGBoost, with its advanced algorithms and fast implementation, enables engineers to create resilient and adaptive solutions for monitoring, control, and predictive maintenance in complex industrial contexts. To fully comprehend and apply those listed above, the essential elements and the supervised learning approach must first be covered. This comprises the model, functions, and parameters that can be trained and used. Future project requests can employ these supervised learning challenges by involving training data with several attributes. The basic variables used in this case are x_i , which is used to predict a desired variable y_i[10,65].

In supervised learning, the model typically refers to the mathematical framework that the forecast y_i value that is made from the input of x_i. A common example is the implementation of a linear model that gives the prediction using the following equation[10,65]:

$$\hat{y}_i = \sum_j \theta_j x_{ij} \tag{4}$$

When it comes to route selection and security, the interpretation of predictions is extremely important. Taking into account Equation 4 from the security point of view, when determining vehicle and route movements, this application can have weighted input features that define decision-making processes. However, its knowledge differs according to the task at hand, whether it is regression or classification. For example, in a logistic regression model, this combination is transformed to give the probability of a class with a positive value. In different circumstances, it can serve as a rating score to compare results. Therefore, when navigating the landscape of optimal routes, the path chosen is determined by the way forecasts are interpreted.

The parameters indicate the unknown variables that are hoped to be determined using the given data. In linear regression, these parameters are commonly indicated by coefficients θ. This is commonly used to represent these parameters. However, it is important to note that a model might have several parameters.

A wide range of tasks can be expressed using prudent selections y_i, including regression, classification, and ranking. Training the model involves identifying the optimal parameters (θ) that best fit the training data (x_i) and the labels (y_i). To train the model, it is necessary to first establish the objective function (Equation 5), which determines how well the model fits the training data[10].

Objective functions contain two parts: the training loss and the regularization term, as shown below:

$$\mathrm{obj}(\theta) = L(\theta) + \Omega(\theta) \tag{5}$$

Equation 5 contains L, which represents the training loss function. The term of regularization is represented by the letter Ω. Training loss measures the predictiveness of the training data model. This may be demonstrated using Equation 6 since a standard choice for L is the mean squared error, which is provided by an equation that is detailed in its formulation.

$$L(\theta) = \sum_i (y_i - \hat{y}_i)^2 \tag{6}$$

Another important equation 7 that should be mentioned is related to logistic regression and analysis of loss function.

$$L(\theta) = \sum_i [y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})] \tag{7}$$

Regularization reduces the complexity of the model by preventing overfitting by punishing high parameter values for balanced flexibility and generalization. Improve the model's capacity to extrapolate patterns from training data to previously encountered occurrences, resulting in better real-world performance. This critical loss function quantifies the differences between the expected and actual values, directing optimization to improve performance.

Starting a journey into practical implementation, decision tree ensembles are an important choice within the XGBoost model repertoire. These ensembles, composed of classification and regression trees (CART), combine the simplicity of individual trees with the robustness of an ensemble technique[10,65,66]. Each tree improves prediction accuracy and generalization by independently training on different subsets of data. Decision tree ensembles are essential for dealing with complicated datasets in a variety of real-world applications because they reduce overfitting and capture complex patterns. Furthermore, examining the combinations of decision trees improves predictive capacity, resulting in more robust and accurate models[66].

Here's an example of a tree ensemble composed of two trees. The ultimate prediction score is calculated by combining the prediction values from each tree. The two trees in the representation are designed to complement each other. Mathematically, the model can be described as follows:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F} \tag{8}$$

where K is the number of trees, f_k is a function in the functional space $\mathcal{F}$, and $\mathcal{F}$ is the set of all possible CARTs. The objective function to be optimized is given by:

$$\text{obj}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \omega(f_k) \tag{9}$$

In boosting, trees are built in sequence, with each following tree attempting to reduce the defects of the previous tree. Each tree acquires information from its predecessors and corrects any faults that were made before it. As a result, the subsequent tree in the sequence will learn from an updated set of residuals[67,68].

When boosting, trees are constructed one after the other, trying to improve on the shortcomings of the one before it. Every tree gains knowledge from its forebears and rectifies whatever mistakes they committed. An updated set of remaining information will be used as a teaching set for the trees in the series that follow. Boosting makes use of trees with fewer splits as opposed to bagging techniques such as Random Forest, which grow trees to their maximum capacity. Small, shallow trees like this are readily understood.

When tuning XGBoost, we can start with a low learning rate to improve accuracy and adjust the number of trees to balance accuracy and overfitting. Testing multiple tree depths helps avoid potential overfitting. Regularization can be used to regulate the complexity of the model, ensuring that the model remains successful but does not become too complicated.

Previous approaches, such as XGBoost and K-Nearest Neighbors, showed excellent predictive power but did not have interpretability, which is critical in automobile security. Agent-based techniques and machine perception emphasize self-awareness, but are limited in scale. Cyber-physical systems and strategic frameworks offer a larger perspective, but frequently overlook detailed feature analysis. Our technique incorporates explainable AI, enabling understanding of feature impacts while maintaining prediction accuracy, resulting in improved security insights in automotive contexts. This balanced integration of interpretability and performance distinguishes it from other techniques.

The ideal selection would be the number of trees or repetitions, the learning rate boosted by gradients, and the depth of the tree, which can be performed using validation approaches such as k-fold cross-validation. An excessive amount of trees might lead to overfitting. Consequently, selecting the stopping circumstances for boosting must be done carefully[67,68].

Security concerns and vulnerabilities pose a significant barrier to future attempts to fully incorporate IoT into industry. Machine learning approaches can help secure IoT systems by detecting anomalies, protecting data privacy, and implementing intrusion detection and prevention (IDPS) tools that provide timely threat intelligence and forecasts[69,70]. As such, one of the techniques is the use of wrapper approaches that explore the space of candidate subsets of characteristics employing a search algorithm, similar to a metaheuristic algorithm, along with selecting the subset that delivers the greatest performance[71].

Methods

Within this section, the metaheuristics algorithm that was used in the experiments is described. First, a brief overview of the basic PSO is provided, followed by the proposed enhanced variant, devised in such way to address the known shortcomings of the baseline method.

Original particle swarm optimization algorithm

The baseline version of particle swarm optimization (PSO) is regarded as one of the oldest metaheuristics methods, developed and proposed back in 1995[11], however, it is still heavily used today. Inspirited by the collective bearings exhibited by the flocks of fish and bird, it is a population-based approach, where particles correspond to the search agents. PSO has been often utilized recently for solving of a variety of optimization

challenges, where notable examples include applications like cloud computing[52], energy efficient clustering[72] and optimization of various machine learning approaches[73,74].

In the elementary version of PSO, each particle is given an initial velocity. As the algorithm executes, each particle updates its location aiming to find a better one. Velocities of the particles are determined by the weight coefficients, that consist of the old speed, the best found so far, accompanied by the best determined velocities of the neighbouring particles, given by Eq. (10).

$$\begin{cases} \overrightarrow{v_i} \leftarrow \overrightarrow{v_i} + \overrightarrow{U}(0,\phi_1) \bigotimes (\overrightarrow{p_i} - \overrightarrow{x_i}) + \overrightarrow{U}(0,\phi_2) \bigotimes (\overrightarrow{p_g} - \overrightarrow{x_i}) \\ \overrightarrow{x_i} \leftarrow \overrightarrow{x_i} + \overrightarrow{v_i}, \end{cases} \tag{10}$$

here $\bigotimes$ marks the component-wise multiplication, where each component v_i ranges between $[-V_{max}, +V_{max}]$, and the array labeled as $\overrightarrow{U}(0,\phi_1)$ correlates to each solution arbitrarily produced from uniform distribution inside $[0,\phi_i]$. The best score of each solution is denoted as p_i, and *pg* represents the global best solution. The particles correlate to the potential solution of the optimization problem within *D*-dimensional search space, where their location is shown as Eq. (11), the best position discovered before the location update determined with Eq. (12), and the collection of velocities given by Eq. (13).

$$X_i = (x_{i1},\ x_{i2},\ \ldots,\ x_{iD}) \tag{11}$$

$$P_i = (p_{i1},\ p_{i2},\ \ldots,\ p_{iD}) \tag{12}$$

$$V_i = (v_{i1},\ v_{i2},\ \ldots,\ v_{iD}) \tag{13}$$

Particles having the optimal fitness, both globally (p_i) and within the group (p_g), are noted. The particles evaluate both while determining the next move in correlation to the distance among their locations and both p_i and p_g. By employing the inertia weight technique, this process can be mathematically determined in line with the Eq (14).

$$v_{id} = W * v_{id} + c_1 * r_1 * (P_{id} - X_{id}) + c_2 * r_2 * (P_{gd} - X_{id}) \tag{14}$$

The relative influence of inertia factors *w* is demonstrated by Eq. (14), where c_1 and c_2 are used as cognitive and social components. Furthermore, r_1 and r_2 denote arbitrary numbers, where the particle's current velocity and location are denoted as v_{id} and x_{id}. Lastly, p_{id} and p_{gd} correspond to p_i and p_g.

Inertia factors are determined by employing the Eq. (15). The starting weight is denoted as w_{max}, and the final weight is given by w_{min}. The maximum count of iterations in a single execution is labeled as *T*, where *t* represents the current iteration.

$$w = w_{max} - \frac{w_{max} - w_{min}}{T} \cdot t \tag{15}$$

Enhanced variant of PSO

A recognized limitation of baseline PSO lies in its slow converging speed, particularly as the algorithm progresses. Conversely, genetic operators, like uniform crossover and mutation, as discussed by[45], offer potential enhancements to the search process through combination of existing and/or novel solutions. Nevertheless, the challenge of striking a balance between exploration and exploitation persists. Hence, this manuscript suggests several modifications to the conventional PSO:

1. incorporating quasi-reflexive learning (QRL) based initialization to enhance the diversity of the initial population
2. introducing genetic operators, namely crossover and mutations, to enhance both exploration and exploitation processes
3. introducing adaptive parameters to measure population diversity and the crossover operator to create an improved balance between exploration and exploitation
4. implementing a mechanism for replacing existing solutions based on population diversity measured by the L1 norm metric

The QRL initialization mechanism

The altered PSO version employed in this study employs a typical initialization approach to generate individuals for the starting population:

$$X_{i,j} = lb_j + \psi \cdot (ub_j - lb_j), \tag{16}$$

where $X_{i,j}$ corresponds to *j*-th component belonging to *i*-th individual, lb_j and ub_j are specifying the lower and upper boundaries of the parameter *j*, while ψ is an arbitrary value drawn from normal distribution inside [0, 1].

Nevertheless, authors in[75] discuss that the broader region of the search space may be secured by supplementing the quasi-reflection-based learning (QRL) procedure to the individuals synthesized by Eq. (16). Moreover, for each individual's parameter *j* (X_j), a quasi-reflexive-opposite parameter (X_j^{qr}) may be calculated in the following way:

$$X_j^{qr} = \mathrm{rnd}\left(\frac{lb_j + ub_j}{2}, x_j\right), \tag{17}$$

where *rnd* aids in choosing the random value within $\left[\frac{lb_j + ub_j}{2}, x_j\right]$ limits.

The addition of the QRL procedure to the initialization process is not introducing supplementary complexity in comparison to the elementary PSO regarding the fitness function evaluations (*FFEs*), as just *NP*/2 individuals are synthesized. The suggested initialization method used in this manuscript is explained in Algorithm 1.

Stage 1: Spawn population P^{init} consisting of $NP/2$ units by Eq. (16)
Stage 2: Synthesize QRL population P^{qr} from P^{init} by Eq. (17)
Stage 3: Produce general initial population P by merging P^{init} and P^{qr} ($P \cup P^{qr}$)
Stage 4: Determine fitness values of all solutions from P
Stage 5: Arrange solutions in P regarding their fitness

Algorithm 1. Stages of QRL initialization algorithm

The described initialization mechanism leverages the diversification in the early stages of execution, and aids in covering a wider search region.

Population diversity procedure
Monitoring the search process of metaheuristics algorithms involves assessing the diversity of the population, as outlined in[76]. This study uses a predefined diversity metric, specifically the $L1$ norm[76], which considers the diversities between individuals within the populace and the dimensionality of the regarded optimization problem.

Let m denote the count of individuals within the populace, and n specify the dimensionality. $L1$ norm can be obtained with respect to the Eq. (18–20):

$$\overline{x_j} = \frac{1}{m}\sum_{i=1}^{m} x_{ij} \tag{18}$$

$$D_j^p = \frac{1}{m}\sum_{i=1}^{m} |x_{ij} - \overline{x}_j| \tag{19}$$

$$D^p = \frac{1}{n}\sum_{i=1}^{n} D_j^p, \tag{20}$$

where $\overline{x}$ is a vector that contains mean locations of the units in each dimension, D_j^p marks the solutions' location diversity vector as $L1$ norm, and D^p represents the diversity scalar value of the general population.

During the initial phases of the algorithm's execution, it's vital for the population to express a high level of diversity. As the execution progresses and the algorithm converges towards promising regions within the search space, this diversity should dynamically decrease. The altered PSO method proposed in this manuscript manages diversity using the $L1$ norm and implements a dynamic threshold parameter (D_t) for this purpose.

Another control variable, marked as *nrs* and denoting the number of replaced individuals, is utilized in the following manner. The value of *nrs* is determined empirically and is assigned to 2 in this research. At the onset of the algorithm's run, the initial value of D_t (D_{t0}) is computed. Subsequently, in every iteration, the condition $D^P < D_t$ is evaluated, where D^P represents the up-to-date diversity of the population.

In the event of low diversity (the case when the condition is met), the pair of poorest solutions ($nrs = 2$) are substituted with a hybrid individual generated through a standard initialization mechanism. This hybrid is created by applying a uniform crossover operator across all gene parameters between the poorest solution and an arbitrary individual, an then applying the mutation operation on each gene.

Alternatively, if the condition $D^P < D_t$ is false, indicating a large diversity that requires to be diminished, another hybrid solution is generated. This solution is formed by applying both the uniform crossover and mutation operators, this time between the best solution and arbitrarily chosen solution from the population.

The value of D_{t0} is obtainable through the following empirically determined expression:

$$D_{t0} = \sum_{j=1}^{NP} \frac{(ub_j - lb_j)}{2 \cdot NP}, \tag{21}$$

assuming that the most of the parameters of every individual are generate within the proximity of the mean value of lower and upper bounds for each component, described by Eq. (16) and Algorithm 1. As the execution continues, and the individuals move towards the optimum, D_t value must be reduced from the starting $D_t = D_{t0}$ in the following way:

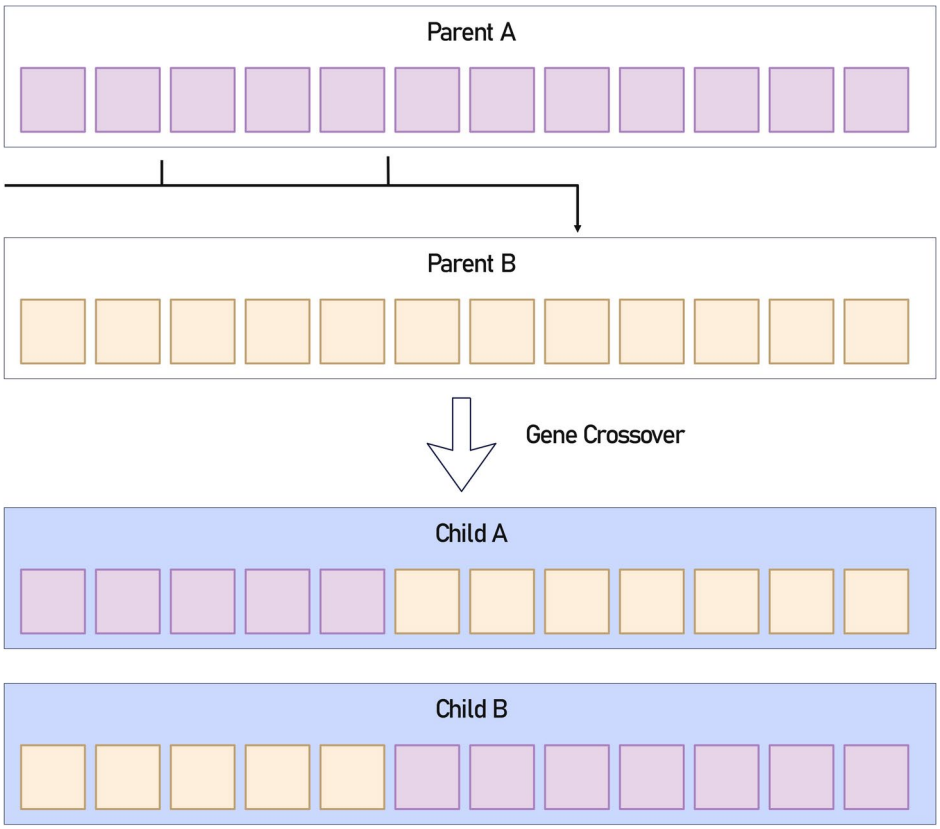

Figure 2. Genetic crossover mechanism.

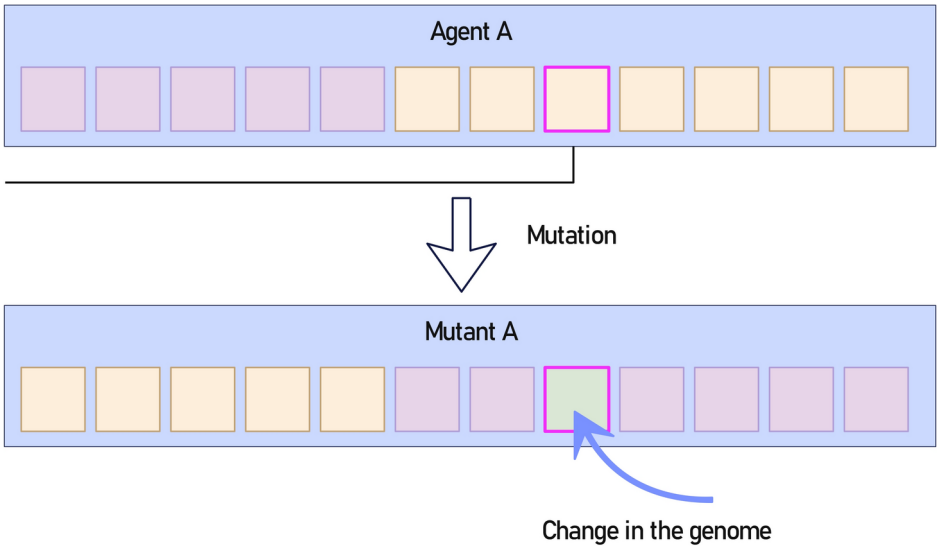

Figure 3. Genome mutation mechanism.

$$D_{t+1} = D_t - D_t \cdot \frac{t}{T} \tag{22}$$

here, t and $t+1$ denote ongoing and following iteration, and T represents the maximal number of rounds in every run. Ultimately, in final phases of execution, the D_t is dynamically reduced, and finally, will not be triggered again regardless of the value of D^P.

Crossover and mutation operators
Taking inspiration from GA[45], proposed method employs uniform crossover and mutation operators. Crossover probability on gene level is represented by *pc*. Crossover of a pair of individuals generates a pair of offspring solutions replacing $nrs = 2$ solutions in the populace, without supplementary evaluation of fitness.

When the algorithm begins, *pc* value is initialized to 0.1, and in every iteration this value is increased with coefficient of $(t/T)/10$, which is described in the following equation:

$$pc_{t+1} = pc_t + (t/T)/10, \tag{23}$$

where *t* marks the ongoing round. This increases the probability of crossover on each gene in later phases of the execution, therefore improving the exploitation. The crossover procedure is graphically presented within Fig. 2.

Following the execution of the crossover operator, a mutation is applied to each gene (parameter). This mutation alters the value of each gene (denoted by *j*) by $(ub_j - lb_j)/100$. The direction of the mutation is determined by a randomly generated value within the range [0, 1]: values less than 0.5 result in a decrease, whereas values larger than 0.5 lead to an increase. Figure 3 graphically illustrates the mutation mechanism.

Suggested Algorithm—GSAPSO
Considering every alteration to the basic variant of PSO that was introduced in this algorithm, the proposed method is labeled genetic self adjusted particle swarm optimization - GSAPSO. The pseudocode of GSAPSO is presented in Algorithm 2. The flowchart of the described GSAPSO is also provided in Fig. 4.

Spawn starting population P by utilizing the QRL approach explained in Algorithm 1
Establish values of D_{t0} and D_t
Initialize $pc = 0.1$
while $(t < T)$ **do**
  Arrange individuals in P with respect to their fitness
  for each individual X within P **do**
    Utilize PSO search
    Compute D^P
    if $(D^P < D_t)$ **then**
      Replace worst *nrs* solutions with a pair of fresh individuals synthesized as hybrid solution among the poorest solution in population and arbitrary solution employing the uniform crossover operator with p_c
    else
      Replace worst *nrs* solutions with a pair of fresh individuals synthesized as hybrid solution among the best solution in population and random solution using the uniform crossover operator with p_c
    end if
    Refresh *pc* with respect to to Eq. (23)
    Refresh D_t by expression (22)
  end for
end while
return The best solution from P

Algorithm 2. GSAPSO pseudocode

Experimental setup

The "can-dataset"[13,14] includes CAN traffic data from the 2017 Subaru Forester, 2016 Chevrolet Silverado, 2011 Chevrolet Traverse, and 2011 Chevrolet Impala. Each vehicle dataset contains samples of both attack-free (normal) traffic and occurrences of various types of attacks. This work focuses on the 2011 Chevrolet Impala portion of the dataset. A subset is used due to the heavy computational demands of optimization tasks. To facilitate testing, data are subjected to one-hot-encoding of categorical features. Attack data is labeled as attack and normal traffic is labeled as normal. The previously described datasets and their data are divided into training and testing portions of the dataset. The testing data is then withheld for evaluation and is comprised of 30% of the total available samples. The reminding 70% are used to model training. Throughout our investigation, we used open-access CAN intrusion detection data sets that have been disclosed under a public license and are freely available to everyone.

The experimental setup involves a comparative analysis between contemporary optimization algorithms, the base PSO[11] algorithm and the introduced modified version. Algorithms included in the comparison include swarm-based algorithms such as the VNS[49], ABC[46] and FA[47] optimizers. Furthermore, the well-established GA[45] is also included. The recently introduced BOA[48] is evaluated as well. Each algorithm is independently implemented with parameter settings taken from the original works that introduced the metaheursitic. Each optimizer is given ten optimization rounds to increase the performance of a solution population of eight agents. To ensure objective outcomes as well as fulfill criteria needed to facilitate statistical evaluation the optimization is repeated though 30 independent runs. Optimizers are tasked with selecting parameter values form that

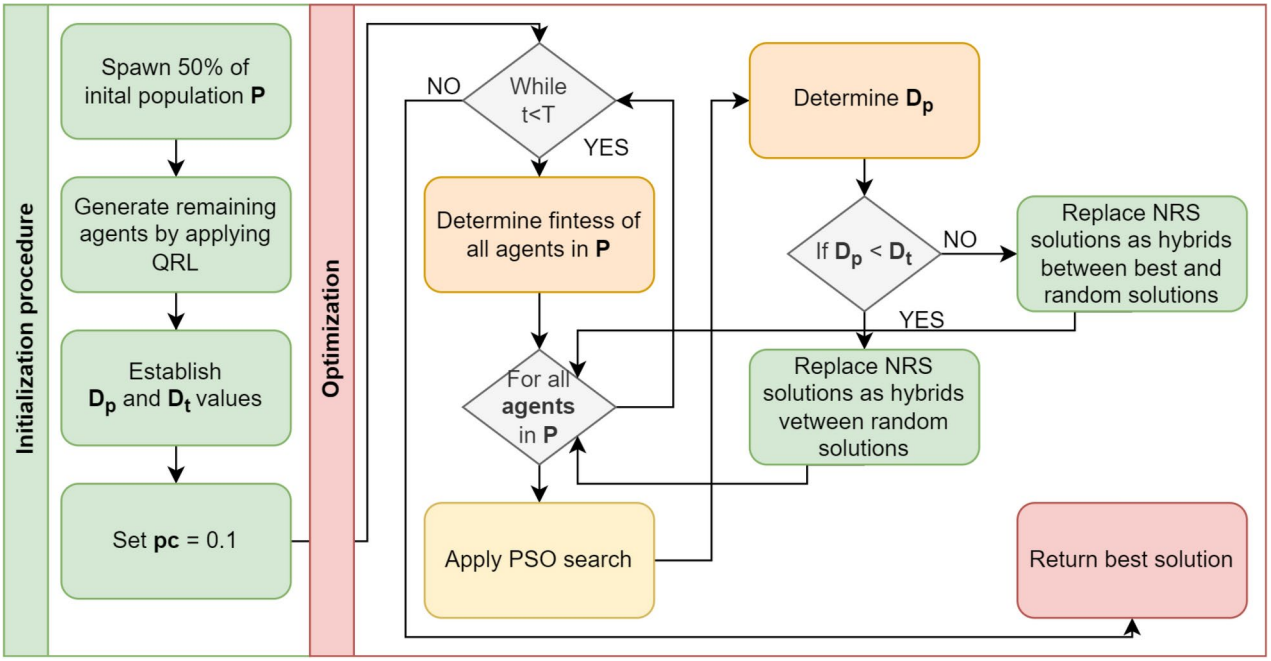

Figure 4. GSAPSO.

Parameter	XGBoost parameter lower bound	XGBoost parameter upper bound
Learning rate	0.1	0.9
Min child weight	1	10
Subsample	0.1	1.0
Colsample by tree	0.1	1.0
Max depth	1	3
Gamma	0	0.8

Table 1. XGBoost parameter constrains for the optimization.

Parameter	KNN parameter lower bound	KNN parameter upper bound
k Parameter	10	25
Weights	0	1
Distance	0	2

Table 2. KNN parameter constrains for the optimization.

attain the best outcomes for each respective classifier. The parameter constraints for the XGBoost algorithm are provided in Table 1 and the KNN classifier in Table 2. These parameters were chosen for optimization as they are considered to have the most impact on the respective model's performance.

The evaluation of vehicle CAN network detection is handled using a set of standard classification metrics. These include precision, accuracy, recall and F_1 score that are determined as per the following:

$$Precision = \frac{TP}{TP + FP} \tag{24}$$

$$Recall = \frac{TP}{TP + FN} \tag{25}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{26}$$

$$F_1 Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{27}$$

here, *TP* and *TN* correspond to true positives and negatives, respectively, while *FP* and *FN* denote false positives and negatives.

The second constructed dataset used an imbalanced dataset with Cohen's Kappa coefficient as the indicator function, which produces reliable results when working with imbalanced data. This coefficient is calculated according to Eq. (28):

$$\kappa = \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e} \tag{28}$$

where p_o denotes the observed values, and p_e marks the expected values. Additionally, to guide the optimization the $error\_rate$ function is used as the objective function due to it's intuitive nature. The $error\_rate$ can be determined as per:

$$error\_rate = 1 - accuracy \tag{29}$$

A flowchart describing the proposed experimental framework is provided in Fig. 5.

Outcomes

The next part displays and examines the experimental results of the implemented solutions. Two sets of experiments have been carried out: the first investigates the optimization of the KNN method, and the second investigates XGBoost optimization. The outcomes are subjected to a rigorous statistical validation and the outcomes presented following the results. Finally, the SHAP interpretation of the best performing models is presented.

Simulations with KNN

Table 3 displays the objective function (classification error rate) results for KNN models improved by each of the algorithms used in the comparative analysis.Based on the objective function findings, the provided technique is comparable to the top-performing model and outperforms the original PSO. In the worst-case execution scenario, the proposed optimizer also achieved the best results. Compared to other optimizers, the GA exhibits a high rate of stability. Good results are seen in the mean and median executions by the FA optimizer.

Further comparisons between algorithms regarding the objective function stability are showcased in Fig. 6. When considering stability, the introduced optimize consistently outperforms the base PSO. While other algorithms demonstrate better outcomes, these are associated with lower stability, such as the case with the VNS algorithm. The FA algorithm demonstrates admirable performance and stability in optimizing KNN hyperparameters for the given task.

A single metric may not provide a complete picture of the model's performance during optimization. In addition to the goal function, there is tracking for the indicator function (Cohen's kappa coefficient). Table 4 presents the results according to the indicator function. The newly implemented optimizer equaled the top performer in the best executions and achieved the finest outcomes under the worst runs. The FA produced an outstanding result, showing the best outcomes for the median and mean runs. The most stable rate is shown by the GA.

Further comparisons between algorithms in terms of indicator function stability are provided in Fig. 7. This figure shows a violin plot and box plot of the Cohen's kappa value over independent runs. It is possible to note that the KNN optimized by the proposed GSAPSO attains the best score in worst run, while several approaches obtain the same best value in the best run (GSAPSO, FA and BOA). On the other hand, FA attained the best mean and median values, while GA had the best stability scores (standard deviation and variance). When considering stability, the introduced optimizer consistently outperforms the base PSO attaining better grouped and higher rated outcomes. Although the GA exhibits good stability rates, the algorithms do not provide optimal results, indicating that the GA did not sufficiently explore this specific issue.

Table 5 contains comprehensive metrics for the best-constructed models optimized by each of the methods used in the comparison study.

Further details on algorithm performance, particularly details on an algorithms ability to avoid local optima and converge towards more promising outcomes can be discerned form optimizer convergence graphs. For every optimizer in the comparative study, the convergence rates for the objective and indicator functions are shown in Figs. 8 and 9, respectively.

Table 6 provides the parameter choices made by each optimizer for the corresponding best-performing KNN classifiers, supporting experimental repeatability.

Simulations with XGBoost

Objective function outcomes for XGBoost models optimized by each algorithm included in the comparative analysis are showcased in Table 7. In terms of worst, mean, and median outcomes, the proposed optimizer outperforms both the original algorithm and the other optimizers in the comparison analysis. Performance in best-case execution is comparable between the new optimization and the BOA.

The stability rates in terms of the objective function are also presented in Fig. 10. The introduced algorithm's strong performance and rate of stability suggest that the enhancements enable it to overcome the local minimum and focus on promising locations inside the search space more effectively.

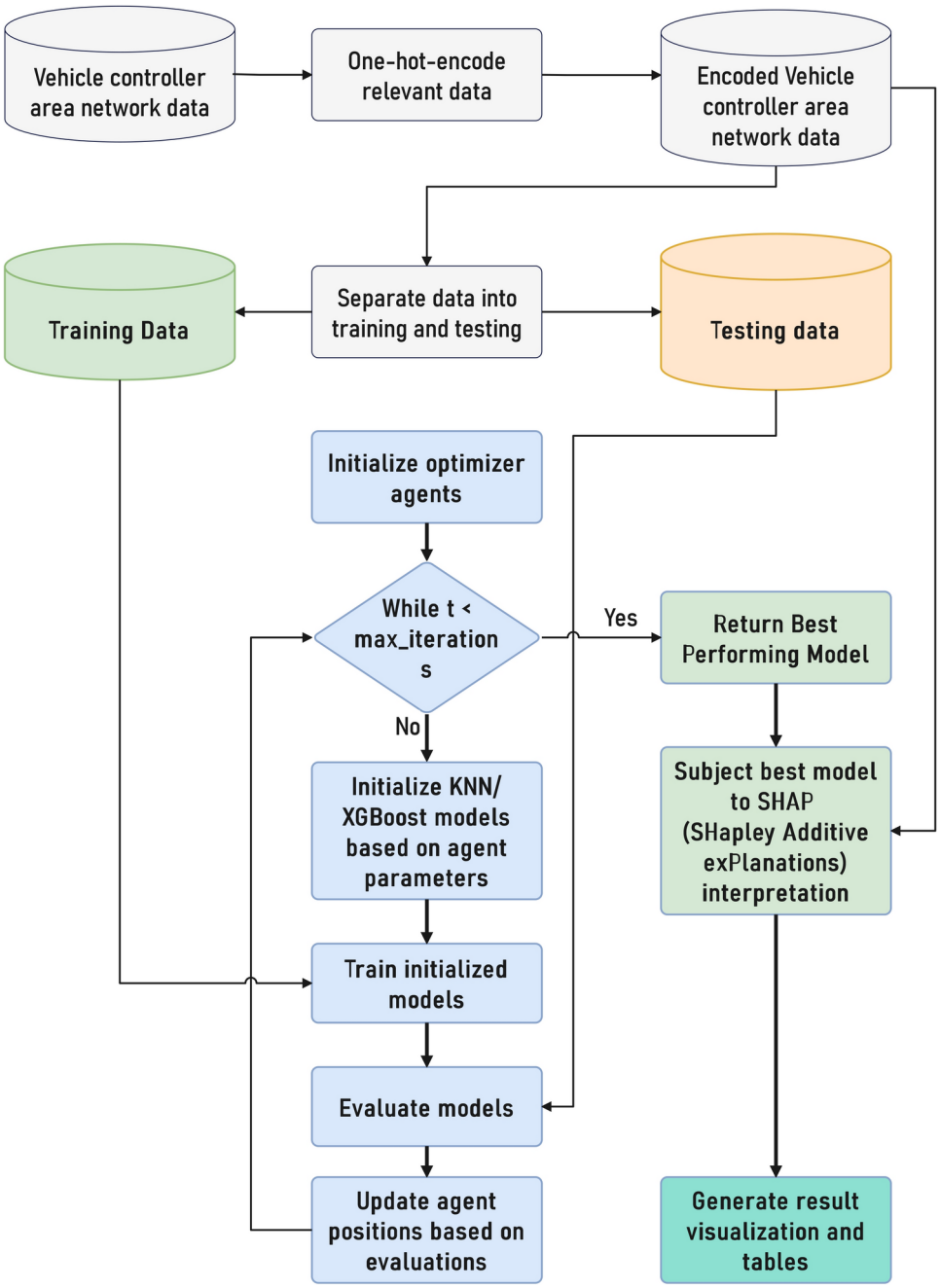

Figure 5. Proposed optimization framework flowchart.

Algorithm	Best	Worst	Mean	Median	Std	Var
KNN-GSAPSO	0.208868	**0.209686**	0.209345	0.209413	0.000313	9.77E−08
KNN-PSO	0.209413	0.210641	0.210232	0.210437	0.000482	2.33E−07
KNN-GA	0.210232	0.210505	0.210402	0.210437	**0.000113**	**1.28E−08**
KNN-VNS	0.209004	0.210914	0.209993	0.210027	0.000922	8.50E−07
KNN-ABC	0.209823	0.210914	0.210641	0.210914	0.000473	2.23E−07
KNN-FA	0.208868	0.210095	**0.209175**	**0.208868**	0.000532	2.83E−07
KNN-BOA	0.208868	0.210641	0.209379	0.209004	0.000737	5.43E−07

Table 3. Objective function outcomes for KNN simulations. Bold is used to higlight the best attained outcomes.

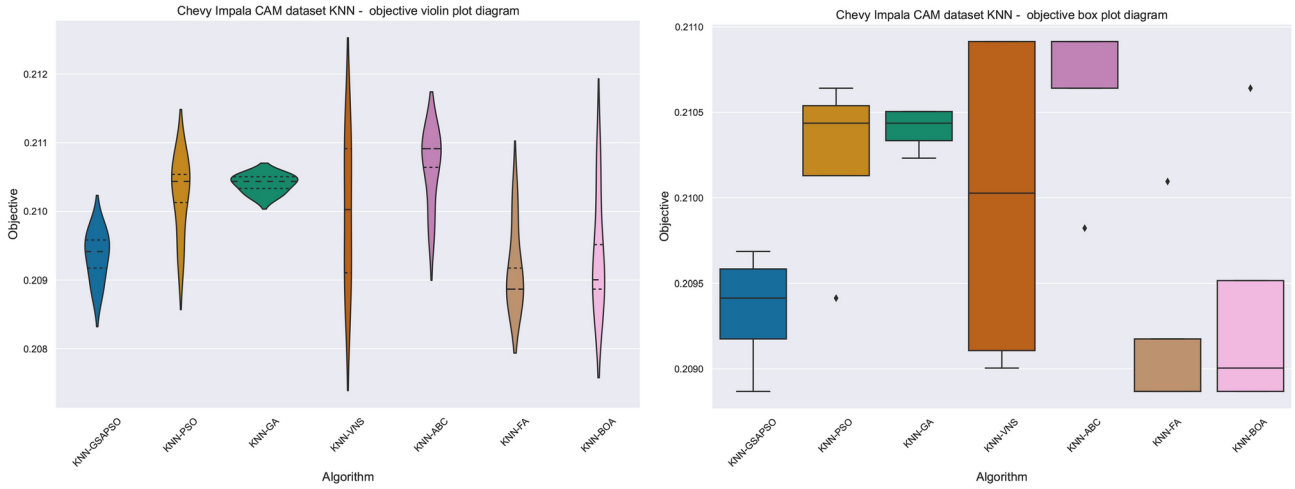

Figure 6. Objective function distribution plots for KNN simulations.

Algorithm	Best	Worst	Mean	Median	Std	Var
KNN-GSAPSO	0.076177	**0.069407**	0.072020	0.071247	0.002744	7.53E−06
KNN-PSO	0.073136	0.061663	0.065386	0.063372	0.004562	2.08E−05
KNN-GA	0.064443	0.062590	0.063285	0.063053	**0.000768**	**5.90E−07**
KNN-VNS	0.074007	0.059806	0.066519	0.066132	0.006736	4.54E−05
KNN-ABC	0.070380	0.059806	0.062450	0.059806	0.004579	2.10E−05
KNN-FA	0.076177	0.066004	**0.073634**	**0.076177**	0.004405	1.94E−05
KNN-BOA	0.076177	0.061663	0.071776	0.074633	0.005974	3.57E−05

Table 4. Indicator function outcomes for KKN simulations. Bold is used to higlight the best attained outcomes.

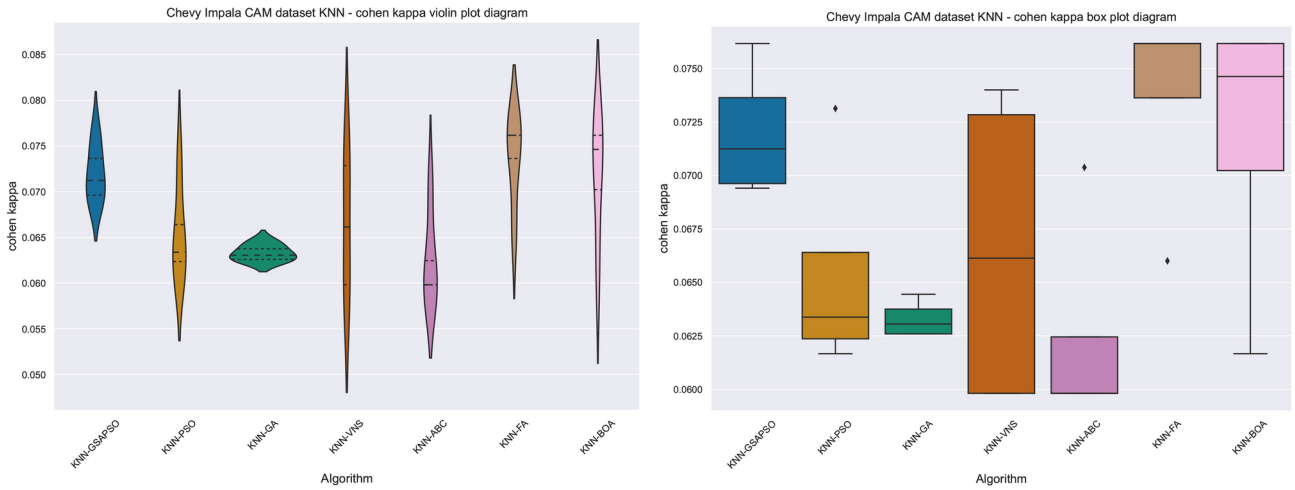

Figure 7. Indicator function distribution plots for KNN simulations.

The results of the indicator function for the XGBoost models optimized by each algorithm included in the comparative analysis are showcased in Table 8. The introduced optimizer improves on the outcomes showcased by the original algorithm as well as the outcomes showcased by other optimizers included in the comparative analysis in terms of worst, mean, and median outcomes. The BOA and the new optimizer have similar performance in the best-case scenario of execution. The results obtained in terms of the objective function are reflected in the results obtained in terms of the indicator function.

Approach	metric	non-attack	attack	accuracy	macro avg.	weighted avg.
KNN-GSAPSO	precision	0.975309	0.788936	0.790996	0.882123	0.829847
	sensitivity	0.049099	0.999650	0.790996	0.524375	0.790996
	f1-score	0.093491	0.881881	0.790996	0.487686	0.708823
KNN-PSO	precision	0.940476	0.788849	0.790587	0.864663	0.822133
	sensitivity	0.049099	0.999126	0.790587	0.524112	0.790587
	f1-score	0.093325	0.881623	0.790587	0.487474	0.708584
KNN-GA	precision	0.218700	0.766067	0.247749	0.492384	0.645915
	sensitivity	0.943443	0.052089	0.247749	0.497766	0.247749
	f1-score	0.355088	0.097545	0.247749	0.226316	0.154078
KNN-VNS	precision	0.975309	0.788936	0.790996	0.882123	0.829847
	sensitivity	0.049099	0.999650	0.790996	0.524375	0.790996
	f1-score	0.093491	0.881881	0.790996	0.487686	0.708823
KNN-ABC	precision	0.938272	0.788523	0.790177	0.863397	0.821394
	sensitivity	0.047234	0.999126	0.790177	0.523180	0.790177
	f1-score	0.089941	0.881419	0.790177	0.485680	0.707682
KNN-FA	precision	0.953488	0.789205	0.791132	0.871347	0.825267
	sensitivity	0.050963	0.999301	0.791132	0.525132	0.791132
	f1-score	0.096755	0.881913	0.791132	0.489334	0.709564
KNN-BOA	precision	0.953488	0.789205	0.791132	0.871347	0.825267
	sensitivity	0.050963	0.999301	0.791132	0.525132	0.791132
	f1-score	0.096755	0.881913	0.791132	0.489334	0.709564
	support	1609	5721			

Table 5. Detailed outcomes for KNN simulations.

The stability rates in terms of the objective function are also presented in Fig. 11. The high performance and high rate of stability of the introduced algorithm suggests that the introduced improvements help it overcome local minimum and focus on promising areas within the search space in a more efficient manner. Similar outcomes in terms of indicator function are attained to those attained in terms of objective function outcomes.

Table 9 contains details metrics for the best-constructed models optimized by each of the methods used in the comparison study.

Further details on algorithm performance, particularly details on an algorithms ability to avoid local optima and converge towards more promising outcomes, can be discerned form optimizer convergence graphs. Convergence rates in terms of the objective and indicator functions for each optimizer included in the comparative analysis are provided in Figs. 12 and 13 respectively.

Table 10 presents the parameter choices made by each optimizer for the corresponding best-performing XGBoost classifiers, to facilitate experimental reproducibility.

Statistical outcome validation

The simulation results were analyzed statistically. For this reason, the best results from each of the 30 distinct runs of each metaheuristic algorithm were collected and analyzed as data series. Deciding to employ a parametric or non-parametric statistical tests from the beginning is critical[77]. Before conducting the tests, the applicability of parametric tests must be evaluated. This involves examining the conditions of independence, normality, and homoscedasticity[78]. The independence criteria is fulfilled since the metaheuristics method always starts by creating a population of pseudo-random solutions. The homoscedasticity condition was assessed using Levene's test[79], which resulted in a $p-values$ of 0.68 for each case, indicating that this condition is also met.

Lastly, the Shapiro-Wilk's test was used for each individual optimization issue to evaluate the normality condition[80]. For every observable technique, the Shapiro-Wilk $p-values$ were calculated independently. Given that every $p-value$ found is less than 0.05, it is possible to reject the null hypothesis (H0). This suggests that there is no normal distribution for the results of any metaheuristic. The KDE diagrams shown in Fig. 14 provide more credence to this idea. Table 11 provides the Shapiro-Wilk test findings in detail. Therefore, it is inappropriate to continue with parametric testing, as the normalcy requirement was not fulfilled.

The authors chose to use non-parametric tests since the requirements for the safe use of parametric tests were not satisfied. In order to evaluate the performance of the suggested approach in relation to other algorithms in terms of the objective function results, the Wilcoxon signed-rank test was carried out. Here, the same data sets that were used to determine normality in the preceding Shapiro-Wilk test were also used. Table 12 displays the results of the Wilcoxon test. The results of the suggested GSAPSO method are statistically significantly better than those of the other algorithms taken into consideration in the comparisons, with the exception of FA for the KNN experiments, since the $p-values$ in each scenario is lower than the threshold bound of $\alpha = 0.05$.

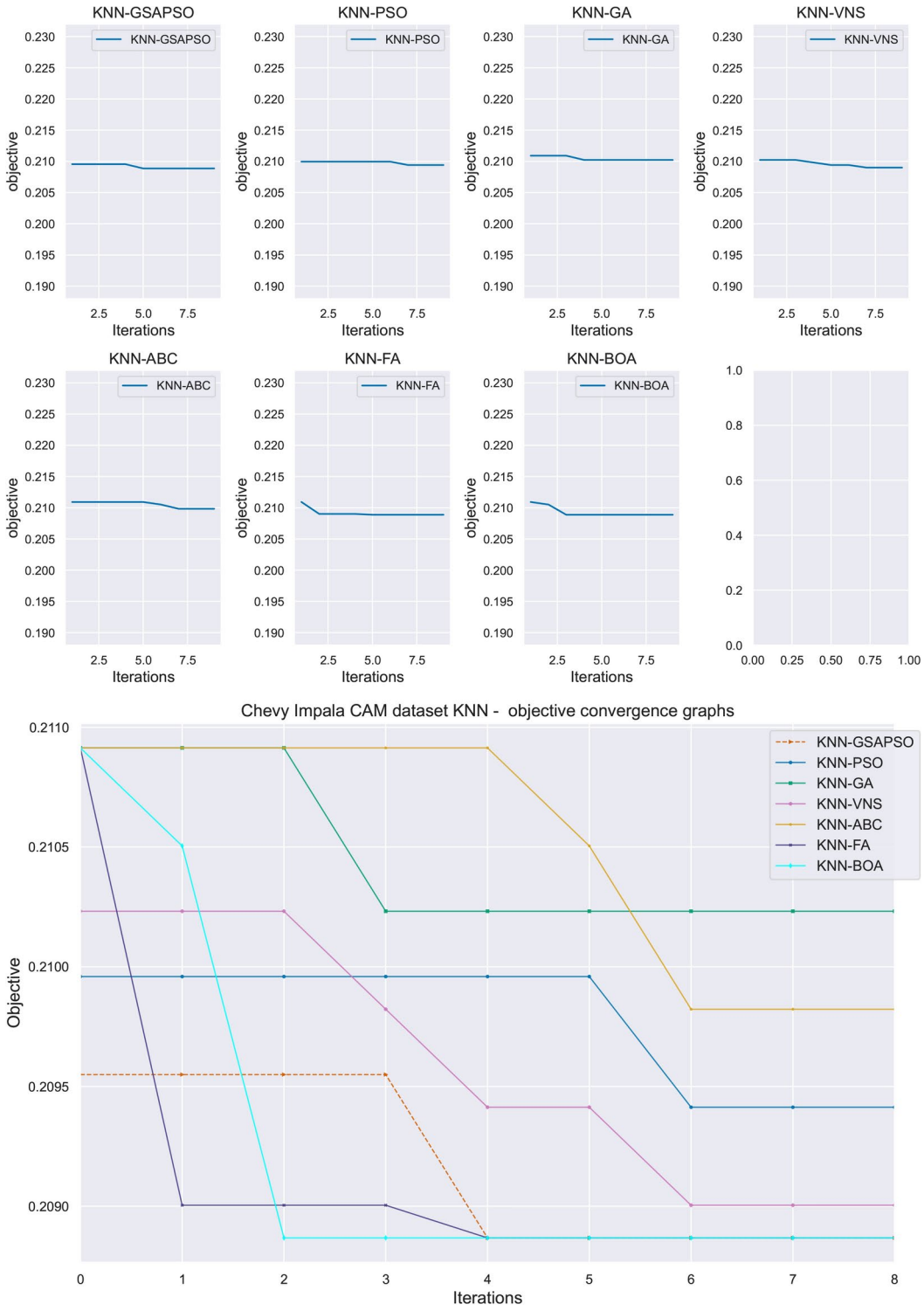

Figure 8. Objective function convergence plots for KNN simulations.

Comparisons to the baseline models

Aiming to demonstrate the proposed models' performance, in this section, comparative analysis against baseline machine learning models is provided. Baseline models were used with their default settings, without applying metaheuristics algorithms for determining the most suitable hyperparameters' values. Beside baseline KNN and XGBoost models, other evaluated models were random forest classifier[81], support vector machine[82], and a pair of deep neural networks (DNNs) structures, having one or two hidden layers, where the count of neural cells was identical to the count of features within the observed dataset. It may be noted that both proposed models

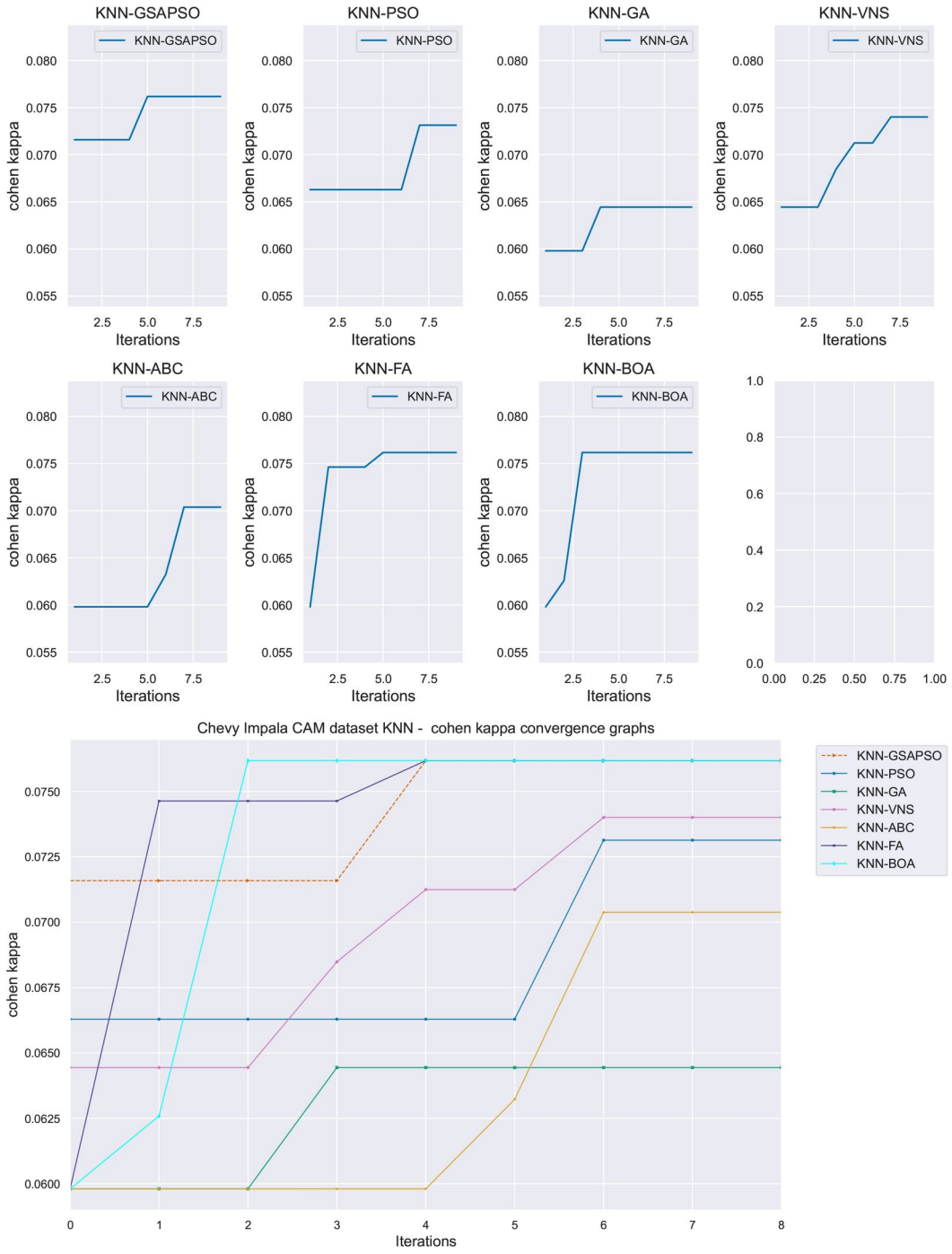

Figure 9. Indicator function convergence plots for KNN simulations.

considerably outperformed other models, both in terms of achieved accuracy, and Cohen's kappa values (Table 13).

Best model SHAP interpretation

When considering model classification and performance, it is often times just as important to understand why certain decisions are made by a model. With simpler models, such as those based on decision trees, these interpretations can be done intuitively and relatively simply, and with more complex models these interpretations can be challenging. Several approaches for model interpretation have been proposed in the literature. This work focuses on SHAP analysis to interpret the decisions made by the best performing XGBoost model[83]. The SHAP approach is based on game theory and can assist understand the importance of each characteristic and its impact on the model's decisions. This technology has already been used in the automotive sector to detect damage[84],

Method	k parameter	weights	distance
KNN-GSAPSO	20	1	2
KNN-PSO	19	0	1
KNN-GA	15	1	2
KNN-VNS	18	0	0
KNN-ABC	16	1	0
KNN-FA	20	0	2
KNN-BOA	16	1	1

Table 6. Parameter selections for KNN simulations.

Algorithm	Best	Worst	Mean	Median	Std	Var
XG-GSAPSO	0.208868	**0.208868**	**0.208868**	**0.208868**	**0.000000**	**0.00E−00**
XG-PSO	0.209550	0.210368	0.209993	0.210027	0.000378	1.43E−07
XG-GA	0.209686	0.210505	0.210095	0.210095	0.000348	1.21E−07
XG-VNS	0.209277	0.210368	0.210027	0.210232	0.000437	1.91E−07
XG-ABC	0.210095	0.210914	0.210607	0.210709	0.000339	1.15E−07
XG-FA	0.209277	0.210914	0.210437	0.210778	0.000679	4.61E−07
XG-BOA	0.208868	0.210505	0.209789	0.209891	0.000590	3.48E−07

Table 7. Objective function outcomes XG simulations.

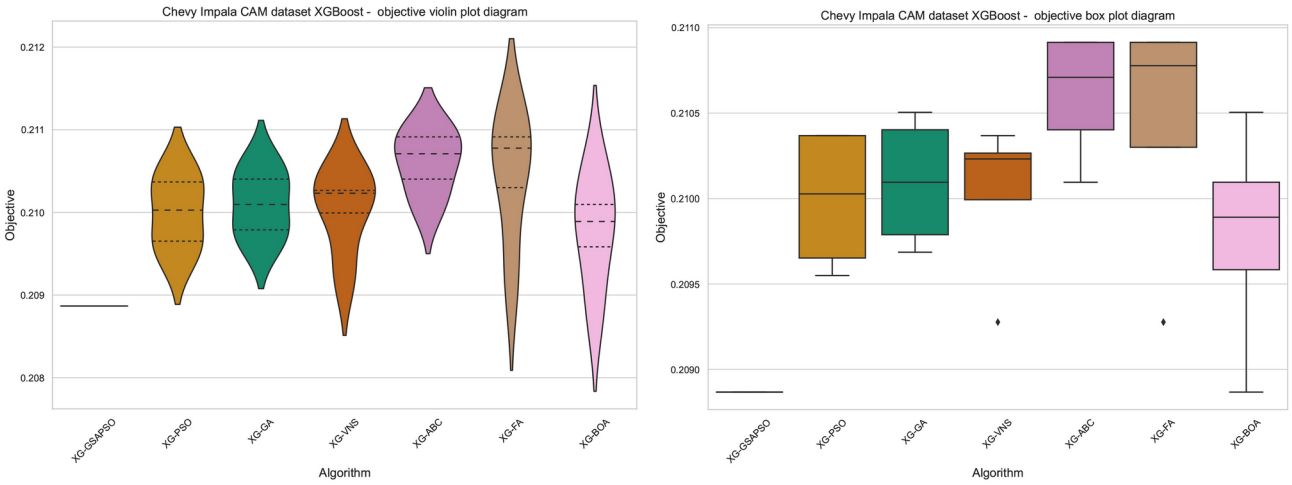

Figure 10. Objective function distributions XG simulations.

Algorithm	Best	Worst	Mean	Median	Std	Var
XG-GSAPSO	0.076177	**0.076177**	**0.076177**	**0.076177**	**0.000000**	**0.00E-00**
XG-PSO	0.072218	0.064154	0.067643	0.067101	0.003134	9.82E-06
XG-GA	0.068141	0.062590	0.065841	0.065829	0.002828	8.00E-06
XG-VNS	0.072798	0.063517	0.066618	0.065080	0.003624	1.31E-05
XG-ABC	0.066004	0.059806	0.062211	0.061517	0.002598	6.75E-06
XG-FA	0.070908	0.059806	0.063046	0.060734	0.004602	2.12E-05
XG-BOA	0.076177	0.063229	0.068705	0.067707	0.004718	2.23E-05

Table 8. Indicator function outcomes XG simulations. Bold is used to higlight the best attained outcomes.

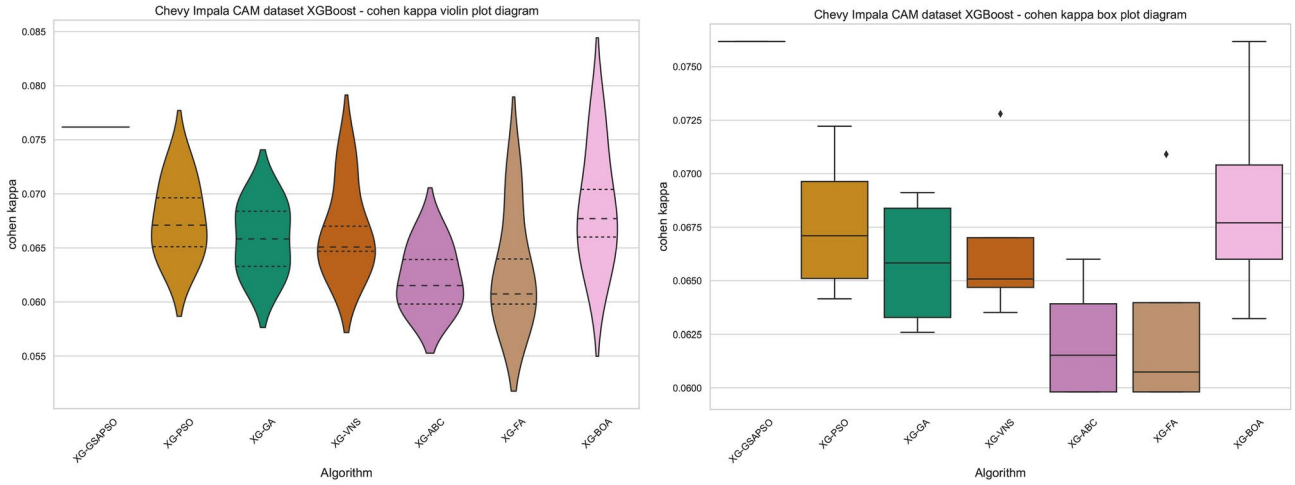

Figure 11. Indicator function distributions.

Approach	metric	non-attack	attack	accuracy	macro avg.	weighted avg.
XG-GSAPSO	precision	0.953488	0.789205	0.791132	0.871347	0.825267
	sensitivity	0.050963	0.999301	0.791132	0.525132	0.791132
	f1-score	0.096755	0.881913	0.791132	0.489334	0.709564
XG-PSO	precision	0.939759	0.788740	0.790450	0.864250	0.821890
	sensitivity	0.048477	0.999126	0.790450	0.523802	0.790450
	f1-score	0.092199	0.881555	0.790450	0.486877	0.708284
XG-GA	precision	1.000000	0.788234	0.790314	0.894117	0.834718
	sensitivity	0.044748	1.000000	0.790314	0.522374	0.790314
	f1-score	0.085663	0.881578	0.790314	0.483621	0.706868
XG-VNS	precision	0.962963	0.788798	0.790723	0.875881	0.827029
	sensitivity	0.048477	0.999476	0.790723	0.523976	0.790723
	f1-score	0.092308	0.881727	0.790723	0.487017	0.708443
XG-ABC	precision	0.985915	0.787987	0.789905	0.886951	0.831434
	sensitivity	0.043505	0.999825	0.789905	0.521665	0.789905
	f1-score	0.083333	0.881356	0.789905	0.482345	0.706183
XG-FA	precision	1.000000	0.788560	0.790723	0.894280	0.834973
	sensitivity	0.046613	1.000000	0.790723	0.523306	0.790723
	f1-score	0.089074	0.881782	0.790723	0.485428	0.707775
XG-BOA	precision	0.953488	0.789205	0.791132	0.871347	0.825267
	sensitivity	0.050963	0.999301	0.791132	0.525132	0.791132
	f1-score	0.096755	0.881913	0.791132	0.489334	0.709564
	support	1609	5721			

Table 9. Detailed outcomes XG simulations.

safety analysis[85] and crash injury severity identification[86,87]. Feature importance determined though SHAP analysis are presented in Fig. 15.

CAN code 1C7 shows a high feature importance for classification. High impact features suggest that the presence of 1C7 codes lowers the probability of a input being a threat. The codes 0688f97500003F and 0688797700003F are the second and third most important features. In contrast, these codes have an enormous positive influence on classification. Suggesting that the presence of these codes hits and is an instance of an attack. The code 0F0F00010000300 is the third most important feature and suggests a negative threat classification. Feature 0F9 also shows high importance, however, values are centered around 0. While these codes and their specific importance remain vehicle and model specific, understanding feature importance helps further data collection, by focusing on relevant features and reducing noise. When considering feature importance, expert opinion plays a crucial role.

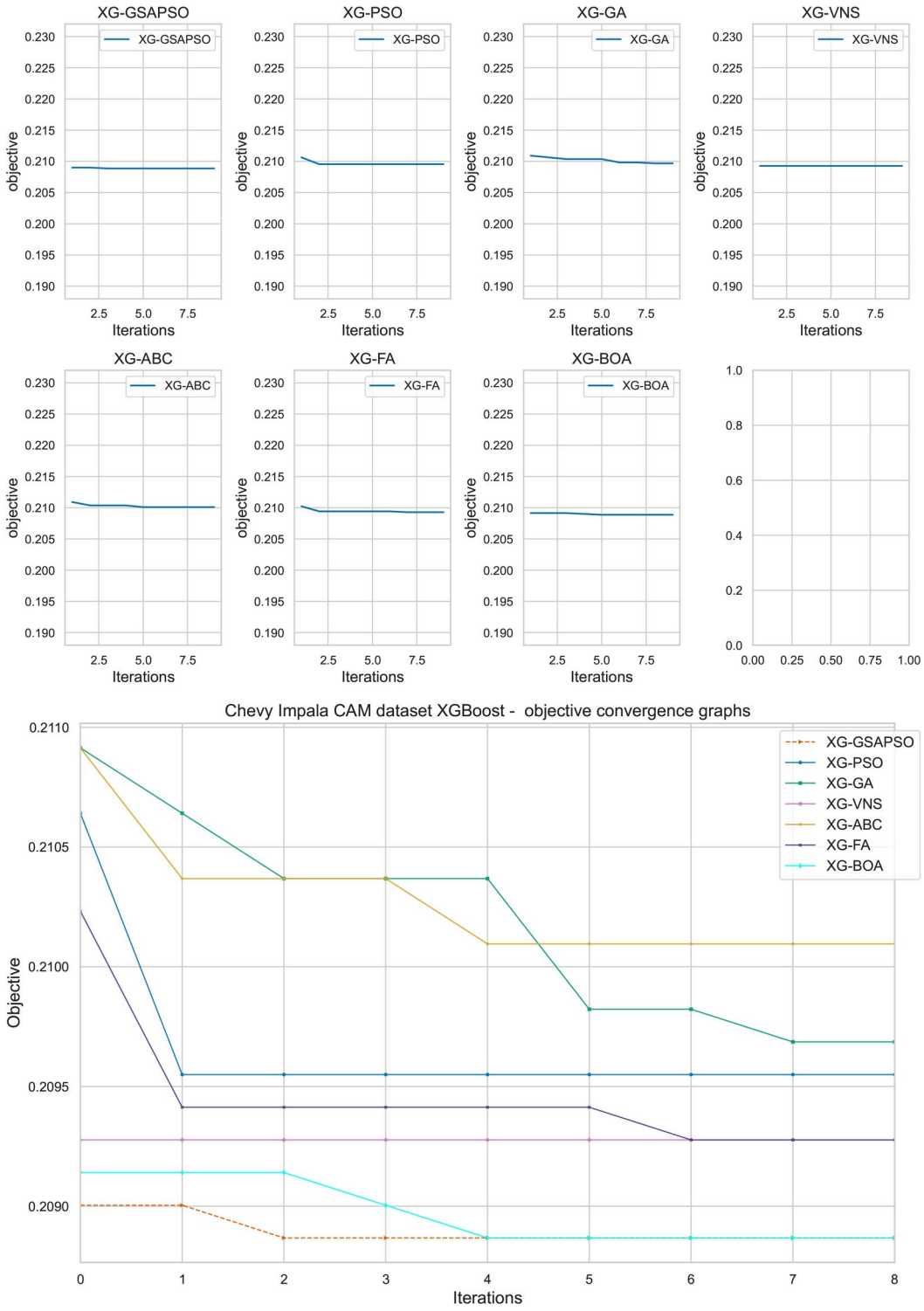

Figure 12. Objective function convergence diagrams XG simulations.

Conclusions

Technological advancement and more automation continue to accelerate, with an increasingly lucrative drive toward fully autonomous transportation. However, challenges related to vehicle and self-driving system security must be addressed with appropriate solutions prior to full deployment. The reliance of self-driving systems on computer systems concerns associated with computer security translate into road safety as well. Network and cyber attacks have the potential to severely impact road safety should self-driving systems see video scale application. This work seeks to tackle cyber security issues in automotive networks through the use of AI classifiers.

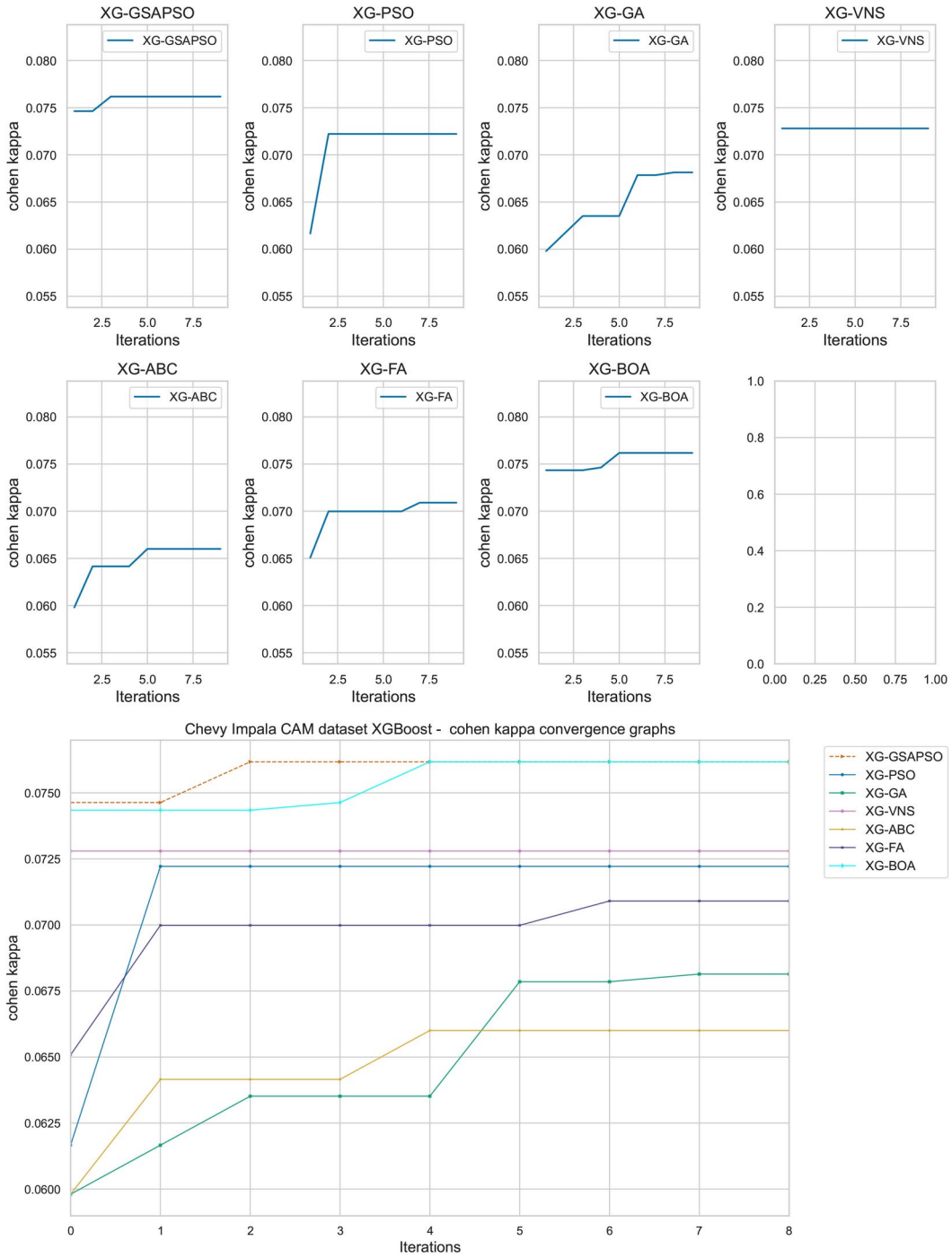

Figure 13. Indicator function convergence diagrams XG simulations.

To achieve good results, and because classifier performance is heavily reliant on hyperparameter selection, a modified version of the PSO method is presented to address the difficult problem of optimization. Two classifiers are evaluated, the KNN and XGBoost algorithms yield promising results, with the best-optimized models exceeding 89% accuracy. Understanding these approaches and tweaking processes can significantly improve their performance in a variety of situations. Key parameters, such as the *k* value in KNN or the learning rate and tree depth in XGBoost, can be carefully adjusted to improve both models' performance further.

This study has several limitations like with any other research, the number of experiments that can be conducted is limited by the availability of data and their up-to-date availability time. Moreover, the number of optimizers that are included in the comparative study is limited by the high computing needs of optimization. As more computing resources become available, future efforts aim to alleviate some of the restrictions. In addition, the next research endeavors aim to investigate the possibilities of the included altered optimizer in addressing other difficult and urgent issues related to forecasting, medicine, and cybersecurity.

Method	Learning Rate	Min Child W.	Subsample	Col by Tree	Max depth	Gamma
XG-GSAPSO	0.900000	1	1.000000	1.000000	9	0.800000
XG-PSO	0.877722	1	0.481578	0.994386	10	0.000000
XG-GA	0.785170	1	0.805085	0.563175	9	0.125654
XG-VNS	0.900000	1	0.564536	1.000000	10	0.000000
XG-ABC	0.824597	1	0.909147	0.681879	5	0.000000
XG-FA	0.766749	1	0.925916	0.346707	9	0.544823
XG-BOA	0.900000	1	1.000000	1.000000	10	0.725957

Table 10. Parameter selections XG simulations.

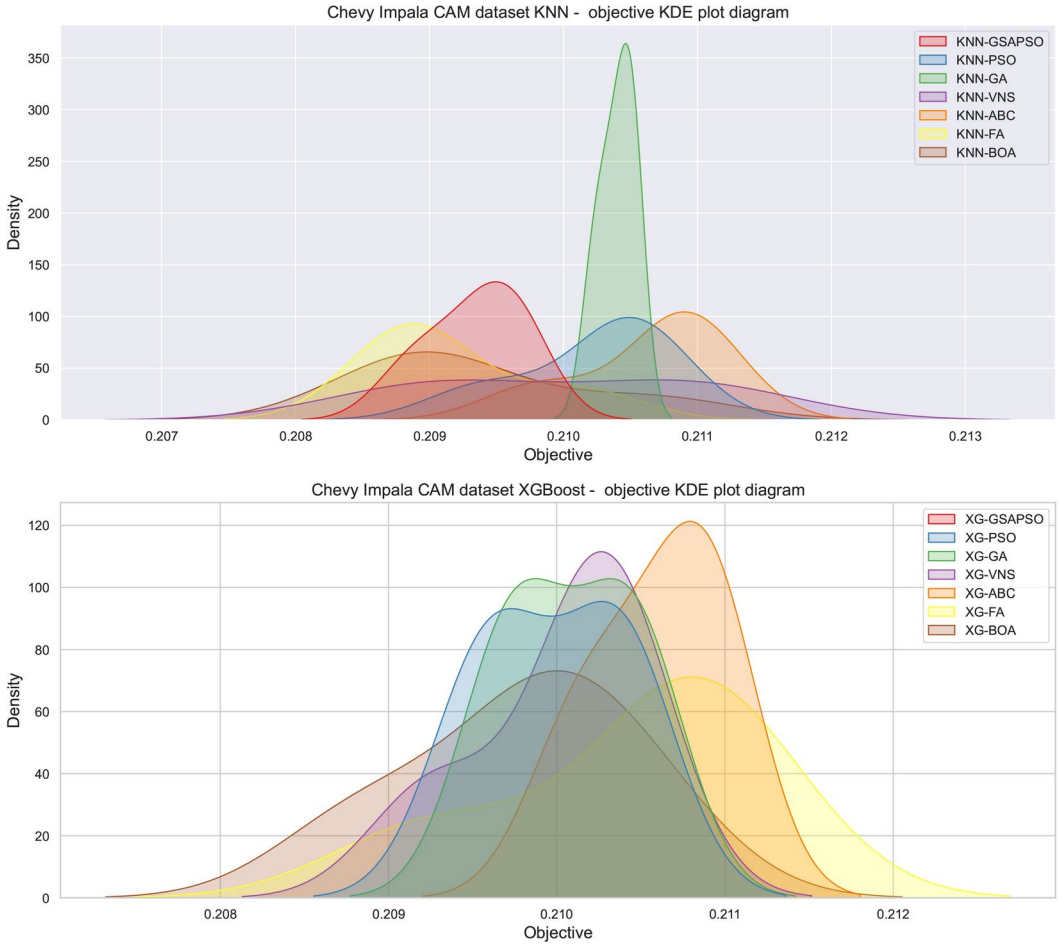

Figure 14. Objective function KDE plots for both experiments.

Model	GSAPSO	PSO	GA	VNS	ABC	FA	BOA
KNN	0.034	0.022	0.043	0.021	0.017	0.025	0.023
XGBoost	0.029	0.035	0.032	0.028	0.037	0.031	0.039

Table 11. Shapiro-Wilk outcomes for KNN and XGBoost experiments for verification of the normality condition for safe utilization of parametric tests.

GSAPSO versus others	PSO	GA	VNS	ABC	FA	BOA
KNN	0.033	0.031	0.039	0.027	0.068	0.044
XGBoost	0.042	0.031	0.039	0.028	0.032	0.046

Table 12. Wilcoxon signed-rank test scores for KNN and XGBoost experiments.

Models	Accuracy	Error rate	Cohen's kappa
KNN-GSAPSO	79.10%	0.208868	0.076177
XG-GSAPSO	79.11%	0.208868	0.076177
KNN	73.28%	0.267233	0.063284
XGBoost	74.36%	0.256417	0.064525
Random forest	72.44%	0.275603	0.056824
SVM	71.74%	0.282571	0.055789
DNN 1 hidden layer	75.86%	0.241393	0.069583
DNN 2 hidden layers	77.53%	0.224712	0.070537

Table 13. Comparisons of KNN-GSAPSO and XGBoost-GSAPSO to other baseline models.

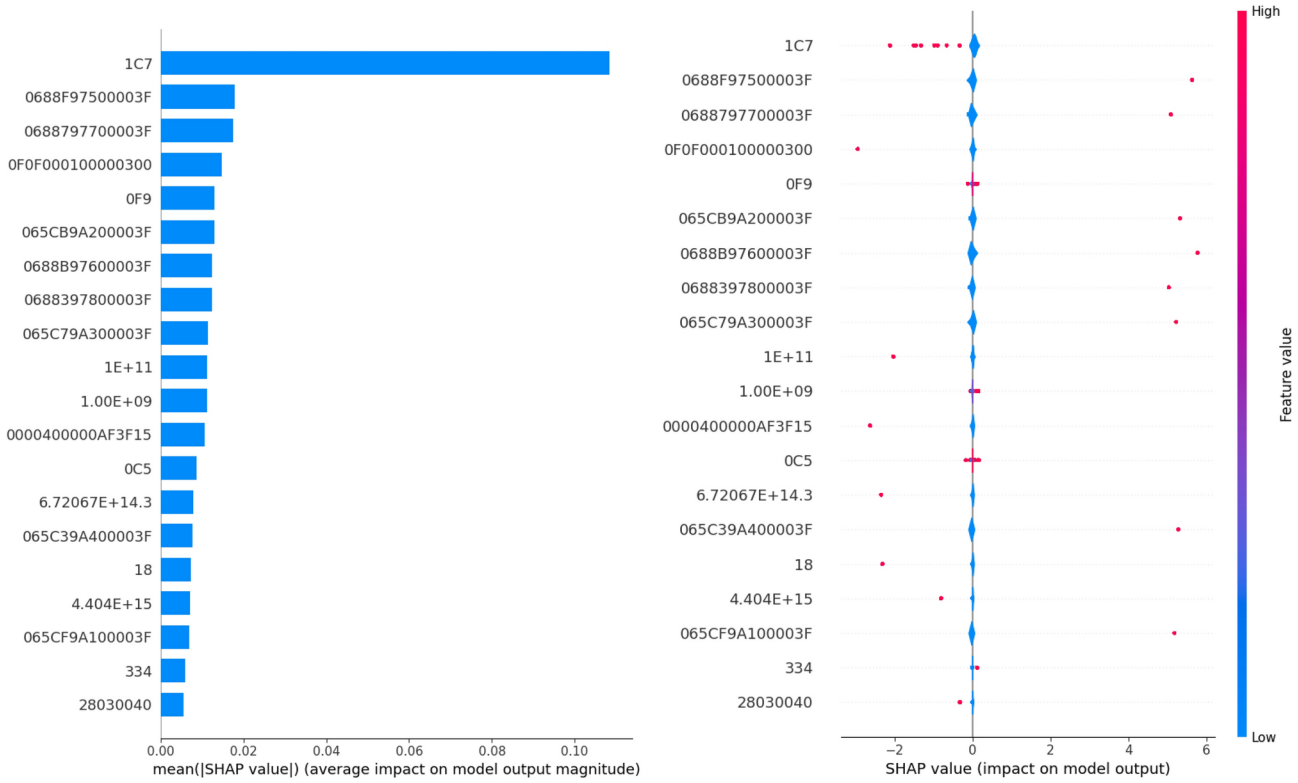

Figure 15. Feature importance determined via SHAP analysis.

Data availibility

During the research, we employed open-access CAN intrusion detection data sets, which were released under a public license and are freely available to anyone. The datasets generated during and/or analyzed during the current study are available from the corresponding author upon a reasonable request.

Received: 28 June 2024; Accepted: 23 September 2024
Published online: 02 October 2024

References

1. Zivkovic, M. et al. Xgboost hyperparameters tuning by fitness-dependent optimizer for network intrusion detection. In: *Communication and Intelligent Systems: Proceedings of ICCIS 2021*, 947–962 (Springer, 2022).
2. Duan, Y., Edwards, J. S. & Dwivedi, Y. K. Artificial intelligence for decision making in the era of big data - evolution, challenges and research agenda. *Int. J. Inf. Manage.* **48**, 63–71. https://doi.org/10.1016/j.ijinfomgt.2019.01.021 (2019).
3. Yang, L. & Shami, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* **415**, 295–316 (2020).
4. Savanović, N. *et al.* Intrusion detection in healthcare 4.0 internet of things systems via metaheuristics optimized machine learning. *Sustainability* **15**, 12563 (2023).
5. Cák, F. & Dakić, P. Creating Feature Model for YAML Generator in CI/CD Pipelines with React Web Application, 529–539 (Springer Nature Singapore, 2024).
6. Dakić, P. Software compliance in various industries using ci/cd, dynamic microservices, and containers. *Open Comput. Sci.* https://doi.org/10.1515/comp-2024-0013 (2024).
7. Dakić, P., Stupavský, I. & Todorović, V. The effects of global market changes on automotive manufacturing and embedded software. *Sustainability* **16**, 4926. https://doi.org/10.3390/su16124926 (2024).
8. Dakić, P. Importance of knowledge management for CI/CD and security in autonomous vehicles systems https://doi.org/10.7251/jit2401007d (2024).
9. Peterson, L. E. K-nearest neighbor. *Scholarpedia* **4**, 1883 (2009).
10. xgboost developers. Xgboost documentation - xgboost 2.0.3 documentation (2024).
11. Kennedy, J. & Eberhart, R. Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks, vol. 4, 1942–1948 (IEEE, 1995).
12. Wolpert, D. H. & Macready, W. G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1**, 67–82 (1997).
13. Brooke Lampe. can-dataset (2023).
14. Lampe, B. & Meng, W. can-train-and-test: A curated can dataset for automotive intrusion detection. *Comput. Secur.* https://doi.org/10.1016/j.cose.2024.103777 (2024).
15. Rajyalakshmi, V. & Lakshmanna, K. Detection of car parking space by using hybrid deep densenet optimization algorithm. *Int. J. Network Manage* **34**, e2228 (2024).
16. Chen, C.-M. *et al.* A provably secure key transfer protocol for the fog-enabled social internet of vehicles based on a confidential computing environment. *Vehicular Commun.* **39**, 100567 (2023).
17. Huang, M.-H. & Rust, R. T. Engaged to a robot? the role of AI in service. *J. Serv. Res.* **24**, 30–41. https://doi.org/10.1177/1094670520902266 (2020).
18. Tyukin, I. Y., Higham, D. J., Bastounis, A., Woldegeorgis, E. & Gorban, A. N. The feasibility and inevitability of stealth attacks. *IMA J. Appl. Math.* **89**(1), 44–84 (2024).
19. Vasconcelos, H. et al. Explanations can reduce overreliance on AI systems during decision-making. *Proc. ACM Human-Comput. Interact.* **7**, 1–38 (2023).
20. Solmaz, G. et al. Learn from IoT. In: Proceedings of the 1st ACM Workshop on Emerging Smart Technologies and Infrastructures for Smart Mobility and Sustainability, https://doi.org/10.1145/3349622.3355446 (ACM, 2019).
21. Luntovskyy, A. & Globa, L. Performance, reliability and scalability for IoT. In: 2019 International Conference on Information and Digital Technologies (IDT), https://doi.org/10.1109/dt.2019.8813679 (IEEE, 2019).
22. Kain, T. et al. FDIRO: A general approach for a fail-operational system design. In: Proceedings of the 30th European Safety and Reliability Conference and 15th Probabilistic Safety Assessment and Management Conference, https://doi.org/10.3850/978-981-14-8593-0_4204-cd (Research Publishing Services, 2020).
23. Suresh, G. M. & Madhavu, M. L. AI based intrusion detection system using self-adaptive energy efficient BAT algorithm for software defined IoT networks. In: 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), https://doi.org/10.1109/icccnt49239.2020.9225415 (IEEE, 2020).
24. Ujjan, R. M. A. *et al.* Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN. *Futur. Gener. Comput. Syst.* **111**, 763–779. https://doi.org/10.1016/j.future.2019.10.015 (2020).
25. Roopak, M., Tian, G. Y. & Chambers, J. An intrusion detection system against DDoS attacks in IoT networks. In: 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), https://doi.org/10.1109/ccwc47524.2020.9031206 (IEEE, 2020).
26. Al-Haija, Q. A. & Zein-Sabatto, S. An efficient deep-learning-based detection and classification system for cyber-attacks in IoT communication networks. *Electronics* **9**, 2152. https://doi.org/10.3390/electronics9122152 (2020).
27. Zekry, A., Sayed, A., Moussa, M. & Elhabiby, M. Anomaly detection using IoT sensor-assisted ConvLSTM models for connected vehicles. In: 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), https://doi.org/10.1109/vtc2021-spring51267.2021.9449086 (IEEE, 2021).
28. Zivkovic, M. *et al.* Novel hybrid firefly algorithm: An application to enhance xgboost tuning for intrusion detection classification. *PeerJ Comput. Sci.* **8**, e956 (2022).
29. Zivkovic, M. et al. Xgboost tuned by hybridized sca metaheuristics for intrusion detection in healthcare 4.0 iot systems. In: International Conference on Engineering, Applied Sciences and System Modeling, 1–16 (Springer, 2023).
30. Salb, M. *et al.* Enhancing internet of things network security using hybrid cnn and xgboost model tuned via modified reptile search algorithm. *Appl. Sci.* **13**, 12687 (2023).
31. Musleh, D., Alotaibi, M., Alhaidari, F., Rahman, A. & Mohammad, R. M. Intrusion detection system using feature extraction with machine learning algorithms in iot. *J. Sens. Actuator Netw.* **12**, 29 (2023).
32. Turukmane, A. V. & Devendiran, R. M-multisvm: An efficient feature selection assisted network intrusion detection system using machine learning. *Comput. Secur.* **137**, 103587 (2024).
33. Tuncali, C. E., Fainekos, G., Ito, H. & Kapinski, J. Simulation-based adversarial test generation for autonomous vehicles with machine learning components, https://doi.org/10.48550/ARXIV.1804.06760 (2018).
34. Stupavský, I. & Vranić, V. Analysing the controversial social media community. In: 2022 IEEE 16th International Scientific Conference on Informatics (Informatics), https://doi.org/10.1109/informatics57926.2022.10083476 (IEEE, 2022).
35. Awad, E. *et al.* The moral machine experiment. *Nature* **563**, 59–64. https://doi.org/10.1038/s41586-018-0637-6 (2018).
36. Stupavsky, I., Dakić, P., Todorovic, V. & Aleksic, M. Historical aspect and impact of fake news on business in various industries (2023). Copyright - Copyright Varazdin Development and Entrepreneurship Agency (VADEA) May 18/May 19, 2023; Last updated - 2023-06-06.
37. Chalás, F., Stupavský, I. & Vranić, V. Discussion manipulation, language and domain dependent models: An overview. In: 2023 Zooming Innovation in Consumer Technologies Conference (ZINC), https://doi.org/10.1109/zinc58345.2023.10174128 (IEEE, 2023).
38. Hbaieb, A., Rezgui, J. & Chaari, L. Pedestrian detection for autonomous driving within cooperative communication system. In: 2019 IEEE Wireless Communications and Networking Conference (WCNC), https://doi.org/10.1109/wcnc.2019.8886037 (IEEE, 2019).
39. Petričko, A., Dakić, P. & Vranić, V. Comparison of visual occupancy detection approaches for parking lots and dedicated containerized rest-api server application. In 9th Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, SQAMIA 2022, vol. 3237 (2022).

40. Golis, T., Dakić, P. & Vranić, V. Creating microservices and using infrastructure as code within the CI/CD for dynamic container creation. In: 2022 IEEE 16th International Scientific Conference on Informatics (Informatics), https://doi.org/10.1109/informatics57926.2022.10083442 (IEEE, 2022).
41. Nguyen, H. T. et al. A deep hierarchical reinforcement learner for aerial shepherding of ground swarms. In Neural Information Processing, 658–669, https://doi.org/10.1007/978-3-030-36708-4_54 (Springer International Publishing, 2019).
42. Kročka, M., Dakić, P. & Vranić, V. Extending parking occupancy detection model for night lighting and snowy weather conditions. In: 2022 IEEE Zooming Innovation in Consumer Technologies Conference (ZINC), 203–208, https://doi.org/10.1109/ZINC55034.2022.9840556 (2022).
43. Dennis, L. A. & Fisher, M. Verifiable self-aware agent-based autonomous systems. *Proc. IEEE* **108**, 1011–1026. https://doi.org/10.1109/jproc.2020.2991262 (2020).
44. Cunneen, M. *et al.* Autonomous vehicles and avoiding the trolley (dilemma): Vehicle perception, classification, and the challenges of framing decision ethics. *Cybern. Syst.* **51**, 59–80. https://doi.org/10.1080/01969722.2019.1660541 (2019).
45. Mirjalili, S. Evolutionary algorithms and neural networks. *Studies Comput. Intell.* **780**, 43–53 (2019).
46. Karaboga, D. & Akay, B. A comparative study of artificial bee colony algorithm. *Appl. Math. Comput.* **214**, 108–132 (2009).
47. Yang, X.-S. & Slowik, A. Firefly algorithm. In Swarm intelligence algorithms, 163–174 (CRC Press, 2020).
48. Hubálovská, M., Hubálovskỳ, Š & Trojovskỳ, P. Botox optimization algorithm: A new human-based metaheuristic algorithm for solving optimization problems. *Biomimetics* **9**, 137 (2024).
49. Hansen, P., Mladenović, N., Brimberg, J. & Pérez, J. A. M. Variable neighborhood search (Springer, 2019).
50. Navazi, F., Yuan, Y. & Archer, N. An examination of the hybrid meta-heuristic machine learning algorithms for early diagnosis of type ii diabetes using big data feature selection. *Healthcare Anal.* **4**, 100227 (2023).
51. Cuk, A. *et al.* Tuning attention based long-short term memory neural networks for parkinson's disease detection using modified metaheuristics. *Sci. Rep.* **14**, 4309 (2024).
52. Bacanin, N., Simic, V., Zivkovic, M., Alrasheedi, M. & Petrovic, A. Cloud computing load prediction by decomposition reinforced attention long short-term memory network optimized by modified particle swarm optimization algorithm. *Ann. Operat. Res.* https://doi.org/10.1007/s10479-023-05745-0 (2023).
53. Predić, B. *et al.* Cloud-load forecasting via decomposition-aided attention recurrent neural network tuned by modified particle swarm optimization. *Complex Intell. Syst.* **10**, 2249–2269 (2024).
54. Zivkovic, T., Nikolic, B., Simic, V., Pamucar, D. & Bacanin, N. Software defects prediction by metaheuristics tuned extreme gradient boosting and analysis based on shapley additive explanations. *Appl. Soft Comput.* **146**, 110659 (2023).
55. Khoshniat, N., Jamarani, A., Ahmadzadeh, A., Haghi Kashani, M. & Mahdipour, E. Nature-inspired metaheuristic methods in software testing. *Soft Comput.* **28**(2), 1503–44 (2024).
56. Aziz, R. M. *et al.* Modified genetic algorithm with deep learning for fraud transactions of ethereum smart contract. *Appl. Sci.* **13**, 697 (2023).
57. Mizdrakovic, V. et al. Forecasting bitcoin: Decomposition aided long short-term memory based time series modelling and its explanation with shapley values. *Knowledge-Based Syst.* **299**, 112026 (2024).
58. Pilcevic, D. *et al.* Performance evaluation of metaheuristics-tuned recurrent neural networks for electroencephalography anomaly detection. *Front. Physiol.* **14**, 1267011 (2023).
59. Pavlov-Kagadejev, M. *et al.* Optimizing long-short-term memory models via metaheuristics for decomposition aided wind energy generation forecasting. *Artif. Intell. Rev.* **57**, 45 (2024).
60. Basha, J. *et al.* Chaotic harris hawks optimization with quasi-reflection-based learning: An application to enhance cnn design. *Sensors* **21**, 6654 (2021).
61. Kumpf, K. et al. Insider threat detection using bidirectional encoder representations from transformers and optimized adaboost classifier. In: 2024 International Conference on Circuit, Systems and Communication (ICCSC), 1–6 (IEEE, 2024).
62. Bacanin, N. et al. Improving performance of extreme learning machine for classification challenges by modified firefly algorithm and validation on medical benchmark datasets. *Multimed. Tools Appl.* https://doi.org/10.1007/s11042-024-18295-9 (2024).
63. Song, X. Student performance prediction employing k-nearest neighbor classification model and meta-heuristic algorithms. *Multiscale Multidis. Model. Exp. Des.* **7**, 4397–4412. https://doi.org/10.1007/s41939-024-00481-9 (2024).
64. Dorji, Y., Rafsanjani, A. K. & AsadAmraji, M. Evaluation model for equipping urban regions with intelligent transportation based on the combination of euclidean and manhattan distances. Iran. J. Sci. Technol., Trans. Civil Eng. https://doi.org/10.21203/rs.3.rs-4593542/v1 (2024).
65. Vidhya, A. XGBoost: Introduction to XGBoost Algorithm in Machine Learning (2018).
66. Galitsky, B. Obtaining supported decision trees from text for health system applications, 71–111 (Elsevier, 2022).
67. de Albuquerque; Paolo Barsocchi;, A. K. B. S. N. S. H. C. (ed.) 5G IoT and Edge Computing for Smart Healthcare (Elsevier, 2022).
68. Eledkawy, A., Hamza, T. & El-Metwally, S. Precision cancer classification using liquid biopsy and advanced machine learning techniques. *Sci. Rep* https://doi.org/10.1038/s41598-024-56419-1 (2024).
69. El-Sofany, H., El-Seoud, S. A., Karam, O. H. & Bouallegue, B. Using machine learning algorithms to enhance iot system security. *Sci. Rep.* https://doi.org/10.1038/s41598-024-62861-y (2024).
70. Karthikeyan, M., Manimegalai, D. & RajaGopal, K. Firefly algorithm based wsn-iot security enhancement with machine learning for intrusion detection. *Sci. Rep.* https://doi.org/10.1038/s41598-023-50554-x (2024).
71. Varzaneh, Z. A. & Hosseini, S. An improved equilibrium optimization algorithm for feature selection problem in network intrusion detection. *Sci. Rep.* https://doi.org/10.1038/s41598-024-67488-7 (2024).
72. Bacanin, N. *et al.* Energy efficient offloading mechanism using particle swarm optimization in 5g enabled edge nodes. *Clust. Comput.* **26**, 587–598 (2023).
73. Cuong-Le, T. *et al.* An efficient approach for damage identification based on improved machine learning using pso-svm. *Eng. Comput.* **38**, 1–16 (2022).
74. Esfandyari, M., Delouei, A. A. & Jalai, A. Optimization of ultrasonic-excited double-pipe heat exchanger with machine learning and pso. *Int. Commun. Heat Mass Transfer* **147**, 106985 (2023).
75. Rahnamayan, S., Tizhoosh, H. R. & Salama, M. M. A. Quasi-oppositional differential evolution. In: 2007 IEEE Congress on Evolutionary Computation, 2229–2236, https://doi.org/10.1109/CEC.2007.4424748 (2007).
76. Cheng, S. & Shi, Y. Diversity control in particle swarm optimization. In: 2011 IEEE Symposium on Swarm Intelligence, 1–9 (IEEE, 2011).
77. Popović, M., Milosavljević, M. & Dakić, P. Twitter data analytics in education using ibm infosphere biginsights. In: Sinteza 2016 - International Scientific Conference on ICT and E-Business Related Research, 74–80, https://doi.org/10.15308/Sinteza-2016-74-80 (2016).
78. LaTorre, A. *et al.* A prescription of methodological guidelines for comparing bio-inspired optimization algorithms. *Swarm Evol. Comput.* **67**, 100973 (2021).
79. Glass, G. V. Testing homogeneity of variances. *Am. Educ. Res. J.* **3**, 187–190 (1966).
80. Shapiro, S. S. & Francia, R. An approximate analysis of variance test for normality. *J. Am. Stat. Assoc.* **67**, 215–216 (1972).
81. Breiman, L. Random forests. *Machine Learn.* **45**, 5–32 (2001).
82. Vapnik, V. Estimation of dependences based on empirical data (Springer Science & Business Media, 2006).

83. Merrick, L. & Taly, A. The explanation game: Explaining machine learning models using shapley values. In: Machine Learning and Knowledge Extraction: 4th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2020, Dublin, Ireland, August 25–28, 2020, Proceedings 4, 17–38 (Springer, 2020).
84. Movsessian, A., Cava, D. G. & Tcherniak, D. Interpretable machine learning in damage detection using shapley additive explanations. *ASCE-ASME J. Risk Uncertainty Eng. Syst. Part B: Mech. Eng.* **8**, 021101 (2022).
85. Yuan, C. *et al.* Application of explainable machine learning for real-time safety analysis toward a connected vehicle environment. *Accident Anal. Prevent.* **171**, 106681 (2022).
86. Kang, Y. & Khattak, A. J. Deep learning model for crash injury severity analysis using shapley additive explanation values. *Transp. Res. Rec.* **2676**, 242–254 (2022).
87. Dong, S., Khattak, A., Ullah, I., Zhou, J. & Hussain, A. Predicting and analyzing road traffic injury severity using boosting-based ensemble learning models with shapley additive explanations. *Int. J. Environ. Res. Public Health* **19**, 2925 (2022).

Acknowledgements

The work reported here was supported by Erasmus+ ICM 2023 No. 2023-1-SK01-KA171-HED-000148295 and Model-based explication support for personalized education (Podpora personalizovaného vzdelávania explikovaná modelom) - KEGA (014STU-4/2024), the Slovak national project Increasing Slovakia's Resilience Against Hybrid Threats by Strengthening Public Administration Capacities (Zvýšenie odolnosti Slovenska voči hybridným hrozbám pomocou posilnenia kapacít verejnej správy) (ITMS code: 314011CDW7), co-funded by the European Regional Development Fund (ERDF), the Operational Programme Integrated Infrastructure for the project: Research in the SANET network and possibilities of its further use and development (ITMS code: 313011W988), co-funded by the ERDF, rurALLURE project - European Union's Horizon 2020 Research and Innovation program under grant agreement number: 101004887 H2020-SC6-TRANSFORMATIONS-2018-2019-2020/H2020-SC6-TRANSFORMATIONS-2020, the Science Fund of the Republic of Serbia, Grant No. 7373, Characterizing crises-caused air pollution alternations using an artificial intelligence-based framework - crAIRsis and Grant No. 7502, Intelligent Multi-Agent Control and Optimization applied to Green Buildings and Environmental Monitoring Drone Swarms - ECOSwarm.

Author contributions

Conceptualization, P.D., M.Z., N.B., V.S.; writing-original draft preparation, P.D., M.Z, L.J., J.K., M.A.; writing-review and editing, M.Z, P.D., L.J., N.B., V.S., M.A.; methodology, P.D., V.S.; formal analysis, P.D. M.Z, L.J., N.B., M.A.; funding acquisition, P.D., J.K., N.B.; All authors have read and agreed to the published version of the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to N.B.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.