scientific reports

Check for updates

OPEN

AutoML-ID: automated machine learning model for intrusion detection using wireless sensor network

Abhilash Singh[1], J. Amutha[2], Jaiprakash Nagar[3], Sandeep Sharma[4]✉ & Cheng-Chi Lee[5,6]✉

Momentous increase in the popularity of explainable machine learning models coupled with the dramatic increase in the use of synthetic data facilitates us to develop a cost-efficient machine learning model for fast intrusion detection and prevention at frontier areas using Wireless Sensor Networks (WSNs). The performance of any explainable machine learning model is driven by its hyperparameters. Several approaches have been developed and implemented successfully for optimising or tuning these hyperparameters for skillful predictions. However, the major drawback of these techniques, including the manual selection of the optimal hyperparameters, is that they depend highly on the problem and demand application-specific expertise. In this paper, we introduced Automated Machine Learning (AutoML) model to automatically select the machine learning model (among support vector regression, Gaussian process regression, binary decision tree, bagging ensemble learning, boosting ensemble learning, kernel regression, and linear regression model) and to automate the hyperparameters optimisation for accurate prediction of numbers of *k*-barriers for fast intrusion detection and prevention using Bayesian optimisation. To do so, we extracted four synthetic predictors, namely, area of the region, sensing range of the sensor, transmission range of the sensor, and the number of sensors using Monte Carlo simulation. We used 80% of the datasets to train the models and the remaining 20% for testing the performance of the trained model. We found that the Gaussian process regression performs prodigiously and outperforms all the other considered explainable machine learning models with correlation coefficient (R = 1), root mean square error (RMSE = 0.007), and bias = − 0.006. Further, we also tested the AutoML performance on a publicly available intrusion dataset, and we observed a similar performance. This study will help the researchers accurately predict the required number of *k*-barriers for fast intrusion detection and prevention.

Intrusion detection at border areas is of utmost importance and demands a high level of accuracy. Any failure in intrusion detection may result in havoc on the nation's security[1]. Each country shares international boundaries with its neighboring countries, extending to thousands of kilometers. Continuous monitoring of such a colossal borderline through occasional patrolling is a crucial problem. To overcome this problem, WSNs are generally used and deployed along the borderline for surveillance and monitoring[2,3]. WSNs are a widely adopted technology that consists of a group of sensors capable of sensing, processing, and transmitting processed information. It can be easily installed anywhere, even in hard-to-reach areas, because it does not require pre-installed infrastructure. The capability of detecting any event or environmental condition makes it more prudent for intrusion detection applications[4,5]. Apart from intrusion detection, WSNs found applications in precision agriculture, health monitoring, environment monitoring, hazards monitoring, and many more[6–9].

[1]Indian Institute of Science Education and Research Bhopal, Fluvial Geomorphology and Remote Sensing Laboratory, Bhopal 462066, India. [2]Gautam Buddha University, School of ICT, Greater Noida 201312, India. [3]Indian Institute of Technology Kharagpur, Subir Chowdhury School of Quality and Reliability, Kharagpur 721302, India. [4]Department of Electronics Engineering, Madhav Institute of Technology and Science, Gwalior 474005, India. [5]Department of Library and Information Science, Research and Development, Center for Physical Education, Health, and Information Technology, Fu Jen Catholic University, New Taipei 242, Taiwan. [6]Department of Computer Science and Information Engineering, Asia University, Taichung 41354, Taiwan. ✉email: sandeepsvce@gmail.com; cclee@mail.fju.edu.tw

Border surveillance, intrusion detection, and prevention problems are addressed with two different approaches. Researchers propose various algorithms and Internet of Things (IoT) solutions for intrusion detection and surveillance in border areas in the first approach. In the second approach, they develop analytical models to estimate the intrusion detection probability in terms of *k*-coverage, *k*-barrier coverage, number of *k*-barriers, and many other performance metrics. Yang et al.[10] have proposed an energy-efficient intrusion detection method that is capable of identifying weak zones of the network deployment region that need to be repaired. After identifying the weak zones, they are repaired to achieve the desired quality of barrier coverage. Specifically, their proposed method focuses on one-directional coverage only for single and multiple intruder scenarios. The authors have claimed that their proposed method and algorithms could enhance the network lifetime. In another work presented in[11], Raza et al. have analysed the impact of heterogeneous WSNs deployed following either uniform or Gaussian distribution scenario. They have studied the impact of sensor density and sensing range of sensor nodes on the intrusion detection probability. They found that the heterogeneous WSNs provide better intrusion detection performance than the homogeneous WSNs at a given sensing range and sensor node density. Similarly, Arfaoui et al.[12] have rendered an analytical model that considers the notion of possible paths that an intruder can follow to cross a belt region in border areas. They have developed a model considering border area characteristics and the intrusion paths to estimate the time taken by an intruder to cross the border area. The authors conclude that their proposed model can detect the intrusion as soon as an intruder enters the restricted border area.

Further, Singh and Singh[13] have presented a smart border surveillance system that uses a WSN which is able to identify and detect the intrusion and then alerts the control center about the presence of an intruder. The proposed system is capable in differentiating between animals and persons. Further, the system uses Raspberry Pi boards integrated with infra-red, ultrasonic and camera sensors and is found to be very effective and accurate to identify any possible intruder. Again, Sharma and Kumar[14] have proposed a ML-based smart surveillance and intrusion detection system for border regions. The proposed system is capable in detection intruders during day time and at night along with the kind of weapon carried by the intruder. The proposed system is made of a high-resolution camera with IR capabilities for day and night vision, a GPS module interfaced with Raspberry Pi to extract the accurate location of the intruder, and a bluetooth scanner to detect the bluetooth signature of the intruder device. The entire module is put into a climate protected box that can be mounted on a high platform. Further, Mishra et al. in[15] have provided a detailed literature review on various ML techniques for intrusion detection. They have also provided a comprehensive discussion on various types of attacks along with their respective features and security threats. With the help of a specific feature, ML techniques can identify and detect the intrusion quickly and accurately. Sun et al.[16] have proposed a three-level intrusion detection model to minimise the memory consumption, computational time, and cost. The proposed model is claimed to decrease memory consumption, time, and cost up to a great extend. Further, in[17], Ghosh et al. have proposed two routing schemes, namely KPS and Loop-Free (LP)-KPS, to enhance the lifetime of a WSN deployed for intrusion detection in border areas or surveillance of some crucial military establishments. On comparing the proposed algorithms with LEACH and TEEN routing algorithms, they found that the proposed algorithms provide enhanced network lifetime. In[18], Benahmed and Benahmed have proposed an optimal approach to achieve a fault-tolerant network for the surveillance of critical areas using WSNs. The proposed approach identifies the faulty sensors and replaces them with active sensors to fill the coverage gap. The proposed approach can provide a sufficient minimum number of sensors to cover the area under surveillance. Another work presented by Arfaoui and Boudriga in[19] provided an efficient surveillance system that can rapidly detect any intruder crossing border areas. In this work, the authors have incorporated the impact of obstacles present in the environment and the terrain of the border areas to derive the expression for intrusion detection probability.

Further, Sharma and Nagar[20] have obtained an analytical expression of *k*-barrier coverage probability for intrusion detection in a rectangular belt region. They have considered all the possible paths an intruder may follow to cross the region. Further, they have also analysed the impact of various parameters such as the number of sensors, sensing range, sensor to intruder velocity ratio, and the intrusion path angle.

The analytical approaches discussed above effectively solve the intrusion detection problem. However, these approaches need validation through the simulation approach, which is time-consuming. For example, a single iteration requires approximately 15 hours for a particular set of network parameters, increasing significantly as the network complexity increases. Various machine learning methods have been proposed to overcome the time-complexity issue associated with the simulations. Recently, Singh et al.[21] proposed three machine learning methods based on GPR to map the *k*-barrier coverage probability for accurate and fast intrusion detection using WSNs. These methods are based on scaling the predictors; scale-GPR (S-GPR), center-mean-GPR (C-GPR), and GPR. They have used synthetic predictors derived from Monte Carlo simulations. They selected many sensors, sensing range of the sensor, sensor to intruder velocity ratio, mobile to static node ratio, angle of the intrusion path, and the required *k*-barriers as potential predictors. They found that the non-standardise methods accurately map the *k*-barrier coverage probability using the synthetic variables with R = 0.85 and RMSE = 0.095. More recently, Singh et al.[22] proposed a logarithmic predictor transformation and scaling-based algorithm coupled with SVR (i.e., LT-FS-ID) to map the number of required *k*-barriers for fast intrusion detection and prevention over a rectangular Region of Interest (RoI) considering uniform sensor distribution. The dimension of the dataset LT-FS-ID is 182×5. They used four predictors to accurately predict the required *k*-barriers. They reported that the proposed approach accurately predicts the *k*-barriers with R = 0.98 and RMSE = 6.47. The feasibility of deep learning algorithms for the intrusion detection has been investigated by Otoum et al. in[23]. They have presented a restricted Boltzmann machine-based clustered IDS (RBC-IDS) for monitoring critical infrastructures using WSNs. Further, they have compared the performance of RBC-IDS with the adaptively supervised and clustered hybrid IDS (ASCH-IDS) and found that both provides same detection and accuracy rates, but, detection time of RBC-IDS is approximately twice that of ASCH-IDS.

Parameters	Values
Network simulator	NS-2.35
Network region	Rectangular RoI
Network area (m^2)	100×50–250×200
Sensor nodes (N)	100–400
Sensing range (R_S)	15–40 m
Transmission range (R_{tx})	30–80 m
Node distribution	Gaussian distribution
Sensing model	Binary sensing model (BSM)

Table 1. Simulation parameters.

The machine learning methods discussed above involve manual selection of the best performing algorithm, which may lead to bias results if the results are not compared with the benchmark algorithm. In addition, the optimisation of the hyperparameter associated with each algorithm is treated differently. To solve this problem, in this paper, we introduced an automated machine learning (AutoML) model to automate the model selection and hyperparameter optimisation task. In doing so, we synthetically extracted potential predictors (i.e., area of the region, sensing range of the sensor, transmission range of the sensor, and the number of sensors) through Monte Carlo simulation. We then evaluated the predictor importance and predictor sensitivity through the regression tree ensemble approach. Subsequently, we applied AutoML on the training datasets to get the best optimised model. We evaluated the performance of the best performing algorithm over the testing data using R, RMSE, and bias as performance metrics.

Material and methods

Predictor generation. The quality of the prediction of a machine learning model depends on the quality of predictors and the model hyperparameters[24]. These predictors can be categorised into real and synthetic-based upon the dataset acquiring process. The real data can be obtained through direct measurements through instruments or sensors. However, the generation of real data involves intensive cost and labor. In contrast to real data, synthetic data can be obtained through mathematical rules, statistical models, and simulations[25]. In comparison to real data, acquiring synthetic data is efficient and cost-effective. Due to this, the use of synthetic datasets to train machine learning models is increased in the past lustrum[21,26–29].

We adopted the synthetic method to extract the predictor datasets using Monte Carlo simulations. In doing so, we have used network simulator NS-2.35 to generate the entire dataset. A finite number of homogeneous (i.e., sensing, transmission, and computational capabilities are identical for each sensor) sensor nodes are deployed according to Gaussian distribution, also known as a normal distribution in a rectangular RoI to achieve this. Gaussian distribution is considered in this study since it can improve intrusion detection capability and is preferred for realistic applications. In a Gaussian distributed network, the probability that a sensor node is located at a point *(x, y)* in reference to the deployed location (x_0, y_0)[30,31] is given by:

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\left(\frac{(x-x_0)^2}{2\sigma_x^2} + \frac{(y-y_0)^2}{2\sigma_y^2}\right)} \tag{1}$$

where σ_x and σ_y are the standard deviations of x and y location coordinates, respectively.

To evaluate the performance of WSNs, we have considered the Binary Sensing Model (BSM)[32], which is the most extensively used sensing range model. Each sensor (S_i) is assumed with the sensing range (R_s) and is deployed at an arbitrary point ($P(x_i, y_i)$). As per BSM, the target can be detected by any random sensor with 100% probability if the target lies with in the sensing range of the sensor. Otherwise, the target detection probability will be equal to zero and is represented mathematically as:

$$P(S_i) = \begin{cases} 1, & if\ d(S_i,P) \le R_s \\ 0, & otherwise \end{cases} \tag{2}$$

where $d(S_i, P) = \sqrt{(x_i - x)^2 + (y_i - y)^2}$, the Euclidean distance between S_i and target point P. In addition, we have considered that any two sensors can communicate if they satisfy the criteria, $R_{tx} \geq 2R_s$, where R_{tx} and R_s represents the transmission range and sensing range, respectively. A barrier is constructed by joining a cluster of sensor nodes across the RoI to detect the presence of intruders. Furthermore, to assure barrier coverage, it is required to identify a Barrier Path (BP) in the RoI. The sensor nodes detect each intruder in the path in this scenario. Thus, to ensure guaranteed k-barrier coverage in the rectangular RoI, the number of required nodes is computed as : $k = \lceil \frac{L}{2R_s} \rceil$ and maximum number of BPs can be computed as $BP_{max} = \lfloor \frac{N}{k} \rfloor$[33], where L is the length of the rectangular RoI, R_s is the sensing range of nodes, and N is the number of sensor nodes. Table 1 lists the various network parameters and their values that have been used to obtain the simulation results.

Relative predictor importance. In machine learning, the choice of input predictors has a substantial control on its performance[28]. Predictor importance analysis is not restricted to any particular representations, tech-

niques, or measures and can be used in any situation where predictive models are required. It is used to express how significant the predictor was for the model's predictive performance, irrespective of the structure (linear or nonlinear) or the direction of the predictor effect. We calculated the relevancy of the selected predictors in estimating the *k*-barriers by estimating each predictor's relative predictor importance score. To do so, we have used the regression tree ensemble technique[21,34]. It is an inbuilt class with a tree-based classifier that assigns a relative score for every predictor or attribute of the data. The higher the score, the more important the predictor.

Initially, we trained a regression tree ensemble model by boosting hundred regression trees (i.e., $t = 100$) with a learning rate of one (i.e., $\delta = 1$) each using the Least Squares gradient Boosting (LSBoost) ensemble aggregation method. Boosting an ensemble of regression algorithms seems to have several advantages, like, handling missing data, representing nonlinear patterns, and yielding better generalisation if weak learners were combined into a single meta learner. In addition, the LSBoost ensemble minimises the mean square error by combining individual regression trees, often known as weak learners. The LSBoost technique successfully trains weak learners on the testing data set, fitting residual errors, and detecting its weak points. Based on such weak points, it generates a new weak learner (l_i) during every iteration. It evaluates its weight (ω_i) in order to enhance the difference between the response value and the aggregated predicted value, hence increasing prediction accuracy. Finally, the algorithm updates the current model (M_i) by emphasising on the prior weak learner's (M_i-1) weak point according to Eq. (3). It then integrates the weak learner into the existing model after training and iteratively generates a single strong learner (M_n, i.e., ensemble of weak learners).

$$M_i = M_{i-1} + \delta \cdot \omega_i \cdot l_i \quad (i = 1, 2, 3, \ldots, n) \tag{3}$$

To explore further the predictor importance, we estimated the coefficients indicating the relative importance of each predictor within the trained model by computing the total variations in the node risk (ΔR) due to split among each predictor, and then normalising it by the total number of branch nodes (R_{BN}) and is mathematically represented as:

$$\Delta R = \frac{R_P - (R_{CH1} + R_{CH2})}{R_{BN}} \tag{4}$$

where R_P indicates the node risk of the parent and R_{CH1} & R_{CH2} indicates the node risk of two children. The node risk at individual node (R_i) is mathematically represented as in Eq. (5);

$$R_i = P_i \cdot E_i \tag{5}$$

where P_i denotes the probability of node i and E_i denotes the node i mean square error.

Predictor sensitivity. We have performed the sensitivity analysis of the predictors using Partial Dependence Plot (PDP)[21,35]. PDP depicts whether a model's predicted response (outcome) changes as a single explanatory variable varies. These plots have the advantage of exhibiting the form of relationship that exists between the variable and the response[36]. Moreover, it depicts the marginal effect of one or more variables on the predicted response of the model[37]. In this study, we have considered the combined impact of two predictors simultaneously from the input predictor set (i.e., υ) on the predictand by marginalising the impact of the remaining predictors. To accomplish this, a subset υ^S and a complimentary set (υ^C) of υ^S is extracted from the predictor set ($\upsilon = \{z_1, z_2, \ldots, z_n\}$) where n represents the total number of predictors. Any prediction on υ is determined by Eq. (6) and the partial dependence of the predictor in υ^S is inferred by computing the expectation (E_c) of Eq. (6):

$$f(\upsilon) = f(\upsilon^s, \upsilon^c) \tag{6}$$

$$\begin{aligned} f^s(\upsilon^s) &= E_c[f(\upsilon^s, \upsilon^c)] \\ &= \int f(\upsilon^s, \upsilon^c) \cdot \rho_c(\upsilon^c) \cdot d\upsilon^c \end{aligned} \tag{7}$$

where $\rho_c(\upsilon^c)$ indicates the marginal probability of υ^c, which is represented in Eq. (8).

$$\rho_c(\upsilon^c) \approx \int p(\upsilon^s, \upsilon^c) \cdot d\upsilon^s \tag{8}$$

Then, the partial dependency of the predictor in υ^s can be determined by :

$$f^s(\upsilon^s) \approx \frac{1}{U} \sum_{i=1}^{U} f(\upsilon^s, \upsilon_i^c) \tag{9}$$

where U represents the total number of observations.

Automated machine learning model. AutoML is used to automate the machine learning process such as data pre-processing, predictor or feature engineering, best algorithm selection, and hyperparameter optimisation[38–40]. For past few years, it has been widely used in industry and academia to solve real and near real-time problems[41–43]. In this study, firstly, we have performed the predictor standardisation using Z-score scaling[44]. Afterward, we divided the complete dataset randomly using Mersenne Twister (MT) random generator in an 80:20 ratio for training and testing the AutoML model. The dimension of the complete dataset is 182×5, where

182 is the number of observations and 5 is the number of predictors (i.e., area of the region, sensing range of the sensor, transmission range of the sensor, and the number of sensors) and the response variable (i.e., k-barrier). The dimension of the training dataset is 145×5, and the dimension of the testing dataset is 37×5. After data division, we have automated the algorithms selection and hyperparameter optimisation step and investigated its performance. Various explainable machine learning models participate in the algorithm selection process, which is discussed next in the upcoming subsections.

Support vector regression model. The Support Vector Regression (SVR) model was introduced by Vapnik et al.[45], and it was developed primarily using the Support Vector Machine (SVM) classifiers. The SVR model has the benefit of being able to optimise the nominal margin using regression task analysis and is a popular choice for prediction and curve-fitting both for linear and nonlinear regression types[46]. The relationship among input and output variables for nonlinear mapping[47] is determined by:

$$y_i = w\phi(p) + q \tag{10}$$

where $p = (p^1, p^2, \ldots, p^n)$ indicates the input, $y_i \in \mathrm{Rl}$ indicates the output, $w \in \mathrm{R_n}$ indicates the weight vector, $q \in \mathrm{R}$ indicates the constant, n indicates the number of training datasets and $\phi(p)$ indicates an irregular function that is used to assign the input to the predictor. To determine w and q, Eq. (11) is used, where χ_i, χ_i^* indicates the slack variable.

$$\begin{aligned} Minimise &: \frac{1}{2}||w^2|| + C\sum_{i=1}^{n}(\chi_i - \chi_i^*) \\ Subject\ to &: \begin{cases} y_i - (w\phi(p_i) + q_i) \le \epsilon + \chi_i \\ (w\phi(p_i) + q_i) - y_i \le \epsilon + \chi_i^* \\ \chi_i, \chi_i^* \ge 0 \end{cases} \end{aligned} \tag{11}$$

In the SVR model, the three basic hyperparameters used are the insensitive loss function (ϵ) that specifies the tolerance margin; the capacity parameter or penalty coefficient or box constraint (C) that specifies the error weight; and the Gaussian width parameter or kernel scale (γ)[48,49]. A high value of C lets SVR reminisce the training data. The smaller ϵ value implies noiseless data. However, the γ value is equally responsible for the under-adjustment or over-adjustment of prediction. Mathematically, it is represented as:

$$K(p_i, p) = e^{(-\gamma||p_i - p||^2)} \tag{12}$$

where K represents the kernel function, γ represents the kernel scale that manages the influence of predictors variation on kernel variation.

Gaussian process regression model. Gaussian Process Regression (GPR), also known as kriging[50] is based on Bayesian theory[51] and is used to solve complex regression problems (high dimension, nonlinearity), facilitates the hyper-parameter adaptive acquisition, easy to implement, and is used with no loss of performance. The fundamental and extensively used GPR is mainly comprised of a simple zero mean and squared exponential covariance function[52] as represented in Eq. (13).

$$K(x, x\prime) = \varpi_f^2 exp\left[\frac{-r}{2}\right] \tag{13}$$

where

$$r = \frac{|x - x\prime|^2}{g^2} \tag{14}$$

where $k(x, x\prime)$ represents the covariance function or kernels that provide the expected correlation among several observations. In the GPR model, there are two hyperparameters used, such as the model noise (ϖ_f) and the length scale (g) that regulates the vertical scale and the horizontal scale of the function change, respectively.

Binary decision tree regression. A Binary Decision Tree (BDT) regression is formed by performing consecutive recursive binary splits on variables, that is of the form $y_i \le v$, $y_i \ge v$, where $v \in \mathbb{R}$ are observed values in a binary regression tree[53], which is represented as:

$$T(y) = \sum_{m=1}^{M} m \cdot B_m(y) \tag{15}$$

where $T(y)$ indicates the regression tree, M indicates the number of tree's terminal nodes, and $B_m\,(y)$ indicates the base function which is determined by:

$$B_m(y) = \prod_{i=1}^{L_m} [y_i(m) - v_{im}] \tag{16}$$

where L_m indicates the total splits, y_i indicates the involved variable, and v_{im} indicates the splitting value. Moreover, the decision tree establishes the rule till the samples in a leaf fall under a specified size, i.e., the minimum leaf (min-leaf) size[54]. Since the min-leaf size defines when splitting must be terminated, it is considered a vital parameter that must be fine-tuned.

Ensemble regression model. Perrone and Cooper[55] proposed a general conceptual framework for obtaining considerably better regression estimates using ensemble methods. Ensemble Learning (EL) enhances performance by building and combining several base learners with specific approaches. It is mainly used when there is a limited amount of training data. It is challenging to choose a suitable classifier with this limited available data. Ensemble algorithms minimise the risk of selecting a poor classifier by averaging the votes of individual classifiers. This study has applied bagging and boosting EL methods due to their widespread usage and effectiveness for building ensemble learning algorithms.

Bagging (Breiman[56,57]), also known as bootstrap aggregation or Random Forest (RF), is one of the most prominent approach for building ensembles, that uses a bootstrap sampling technique to generate multiple different training sets. Subsequently, the base learners are trained on every training set, and then combining those base learners to create the final model. Hence, bagging works for a regression problem as follows: Consider a training set, S that comprises of data $\{(X_i, Y_i), i = 1, 2, \ldots, m\}$, where X_i and Y_i represents the realisation of a multi-dimensional estimator and a real valued variable respectively. A predictor $P(Y|X = x) = f(x)$[58] is represented as:

$$\zeta_m(x) = h_m(S_1, S_2, \ldots S_m)(x) \tag{17}$$

At first, create a bootstrapped sample Eq. (18) based on the empirical distribution of the pairs $S_i = (X_i, Y_i)$, next, using the plug-in concept, estimate the bootstrapped predictor as shown in Eq. (19). Finally, the bagged estimator is represented by Eq. (20).

$$S_i^* = (Y_i^*, X_i^*) \tag{18}$$

$$\zeta_m^*(x) = h_m(S_1^*, S_2^*, \ldots S_m^*)(x) \tag{19}$$

$$\zeta_{m;B}(x) = P|S_m^*(x)| \tag{20}$$

Moreover, the three hyperparameters used in bagging are the MinLeafSize (minimum number of observations per leaf), NumVariablesToSample (number of predictors to sample at every node), and the NumLearningCycles (number of trees). The first two parameters determine the tree's structure, while tuning the final parameter helps balance efficiency and accuracy.

Boosting (Freund[59]) is another ensemble method that aims to boost the efficiency of a given learning algorithm. The Least-Squares Boosting (LSBoost) ensemble method is used in this study because it is suited for regression and forecasting problems. LSBoost aims to reduce the Mean Squared Error (MSE) between the target variable (Y) and the weak learners' aggregated prediction (Y_p). At first, median of (Y), represented as $(\widetilde{Y})$ is computed. Next, to enhance the model accuracy, several regression trees $(r_1, r_2, \cdots, r_m)$ are integrated in a weighted manner. Individual regression trees are determined by the following predictor variables (X)[60]:

$$Y_p(X) = \widetilde{Y}(X) + \eta \sum_{m=1}^{d} w_m \times r_m(X) \tag{21}$$

where (w_m) represents the weight for the m model, d represents the weak learners, and η with $0 < \eta \leq 1$ represents the learning rate.

Kernel regression model. Kernel regression (Nadaraya[61]) is the most used non-parametric method on account of the virtue of kernel and is undoubtedly known as univariate kernel smoother. In order to achieve a kernel regression, a collection of kernels are locally placed at every observational point. The kernel is set a weight to every location depending on its distance from the observational point. A multivariate kernel regression[62] determines how the response parameter, y_i is dependent on the explanatory parameter, x_i, as in Eqs. (22) and (23).

$$E(y_i|x_i) = m(x_i) + \psi_i \tag{22}$$

and

$$y_i = m(x_i) + \psi_i \tag{23}$$

where $E[\psi_i] = Cov[m(x_i), \psi_i] = 0$, $m(.)$ represents a non-linear function, and ψ_i is random with mean zero and variance σ^2. It describes the way that y_i varies around its mean, $m(x_i)$. The mean can be represented as the probability density function f:

$$m(x_i) = E[Y_i|x_i = x] = \frac{\int y.f(x,y)dy}{\int f(x,y)dy} = \frac{\int y.f(x,y)dy}{\int f(x)} \tag{24}$$

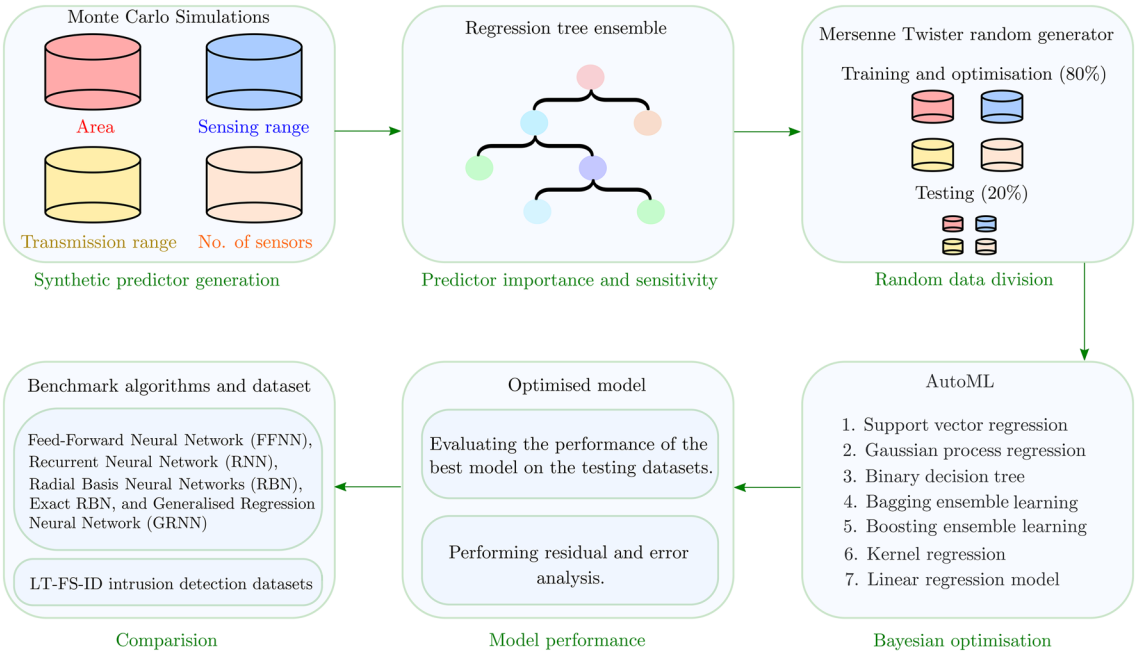

Figure 1. Flowchart of the proposed methodology.

Linear regression model. A linear regression model[63] examines the relationship among different influential predictors and an outcome variable. The basic linear regression model, which represents the universal set of two-variable and multiple regression as complementary subsets, can be expressed as:

$$Y = a + \sum_{i=1}^{n} b_i X_i + u \tag{25}$$

where Y represents the dependent variable, $X_1, X_2, \ldots, X_n$ represents the n independent variables, a and b represents the regression coefficients and u represents the stochastic disturbance-term that could be caused by an undefined independent variable.

Bayesian optimisation. Bayesian Optimisation (BO)[64,65] is an efficient approach for addressing optimisation problems characterised by expensive experiments. It keeps track of the previous observations and forms a probabilistic mapping (or model) between the hyperparameter and a probabilistic score on the objective function that is to be optimised. The probabilistic model is known as a surrogate of the objective function. The surrogate function is much easy to optimise, and with the help of the acquisition function, the next set of hyperparameters is selected for evaluation on the actual objective function based on its best performance on the surrogate function. Hence, it comprises a surrogate function for determining the objective function and an acquisition function for sampling the next observation. In BO, the objective function *(f)* is obtained from the Gaussian Process (GP) as described in Eq. (26).

$$f(x) \sim GP(\mu(x), \vartheta(x_i, x_j)) \tag{26}$$

where μ and ϑ are calculated from the observations of x[66].

We select the best performing algorithm among the above-discussed models with the optimised hyperparameter. Lastly, we evaluated the performance of the best-performing algorithm using the test dataset. A flowchart of the detailed methodology is illustrated in Fig. 1.

Results

Predictor importance and sensitivity. We plotted the relative predictor importance score of each predictor along with their respective box plot for a better visual representation of the datasets (Fig. 2). We found that the relative predictor importance score ranges approximately from 9 to 152. The higher the value of the relative estimate, the more relevant is the predictor in estimating the response variable (i.e., k-barriers). We found that out of these four predictors, the transmission range of the sensor emerges as the most relevant predictor in predicting the required number of k-barriers for fast intrusion detection and prevention considering Gaussian node distribution over a rectangular region. The number of sensors also shows good relevancy in predicting the response variable and ranked second. The area of the region of interest and the sensing range of the sensor shows fair relevancy and ranked third and fourth, respectively.

We also evaluated the impact of each predictor on the response variable. We plotted the partial dependence plot for each possible pair of predictors (Fig. 3a–f). For a better visual inspection, we also plotted the

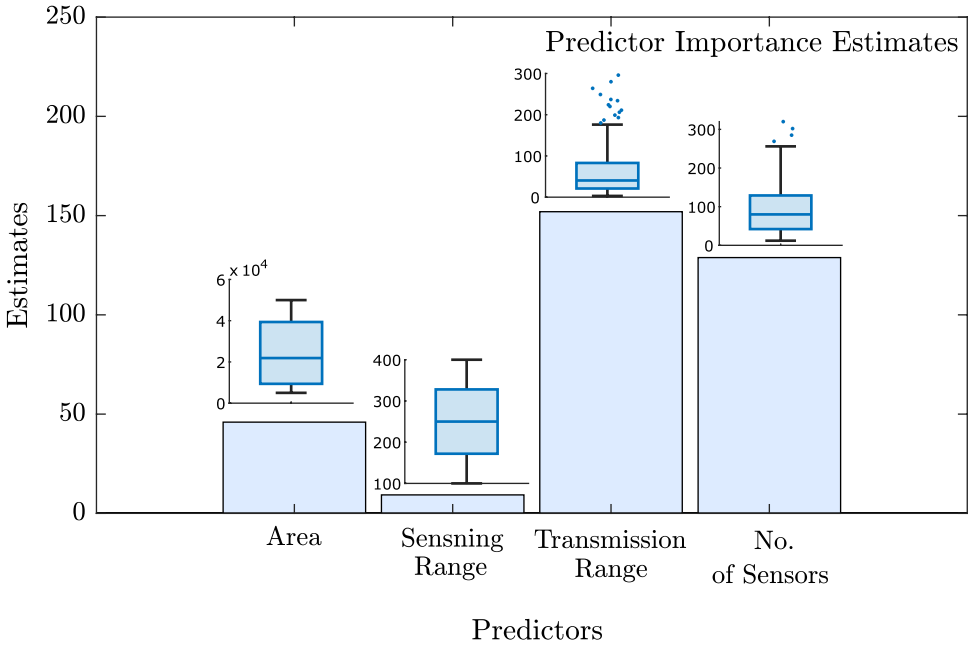

Figure 2. Graph showing the relative predictor importance score for all four predictors. The estimates for the area of the RoI, sensing range of the sensor, transmission range of the sensor, and the number of sensors are 46.0, 9.3, 152.0, and 128.9, respectively.

three-dimensional plot and its two-dimensional illustration. We observed that the area of the RoI has a slightly negative impact on the target variable i.e., the response variable decreases with an increase in the area of the RoI. However, an inverse relationship is observed with all other predictors. The sensing range of the sensor, the transmission range of the sensor, and the number of the sensors have a positive impact on the response variable i.e., the response variable increases with an increase in these predictors.

Model performance. We iteratively selected the best machine learning model with optimised hyperparameters value using the Bayesian optimisation[67–69] on the 80% of the datasets (Fig. 4). We used Eq. (27) as the objective function (*Obj*) to select the best machine learning model with optimised hyperparameters.

$$Obj = log(1 + valLoss) \tag{27}$$

where *valLoss* is the cross-validation mean square error (CV-MSE). At each iteration, the value of the objective function is computed for any one of the participating models. The model (with optimised hyperparameters), which returns the minimum observed loss (i.e., the smallest value of the objective function so far), is considered as the best model. After iterating for 120 iterations, the AutoML algorithm returned the GPR model as the best model along with the optimal hyperparameters (i.e., for the GPR model; sigma = 0.98). Before returning the model, the AutoML algorithm retrains the GPR model on the entire training dataset.

Once we get the trained GPR model, we evaluate its performance on the training datasets to estimate the training accuracy. We found that the model performed well on the training datasets with a correlation coefficient (R = 1), root mean square error (RMSE = 0.003), and bias = 0. However, for an unbiased evaluation, we evaluated the performance of the trained model on the test datasets (i.e., 20% of the total datasets). In doing so, we fed the testing predictors into the trained GPR model and obtained the predicted response. We then compared the GPR predicted *k*-barriers with the observed values (Fig. 5a). We found that the GRP model performs prodigiously with a R = 1, RMSE = 0.007, and bias = − 0.006. All the data points are aligned along the regression line and lie well inside the 95% Confidence Interval (C.I).

Further, to assess the appropriateness of the plotted linear regression plot, we performed residual analysis. We plotted the time series of the observed and the predicted values along with the corresponding residual values (Fig. 5b). We found that the residuals are significantly low and do not follow any pattern, which indicates a good linear fit.

To understand the distribution of the error (i.e., difference of predicted and observed values), we performed error analysis using error histogram (Fig. 6). To do so, we plotted the error histogram using ten bins. The error ranges from −0.00997 from the left to 0.00356 on the right of the histogram plot. We found that the error follows a right-skewed Gaussian distribution. The peak of the distribution lies in the underestimated region. Lastly, we presented the results of the remaining algorithms of the AutoML (i.e., SVR, BDT, Bagging ensemble learning, Boosting ensemble learning, kernel, and linear regression) in Table 2. We found that the best performing AutoML algorithm (i.e., GPR) outperforms all the other algorithms.

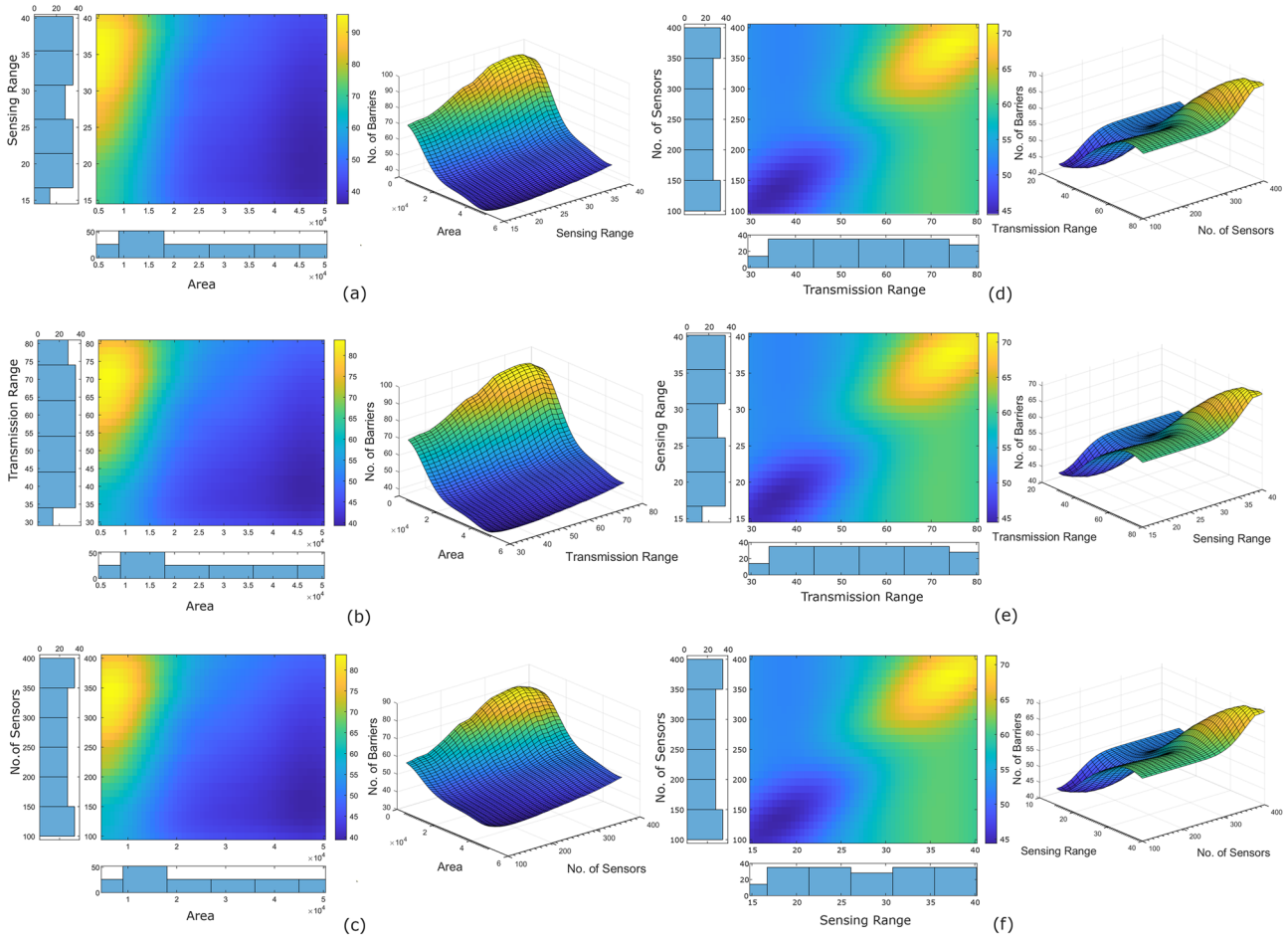

Figure 3. Two-dimensional and three-dimensional partial dependency plots show the predictor sensitivity of all possible predictor pairs. The histogram along the x and y-axis of the two-dimensional plot shows the distribution of the predictor and the response variable, respectively.

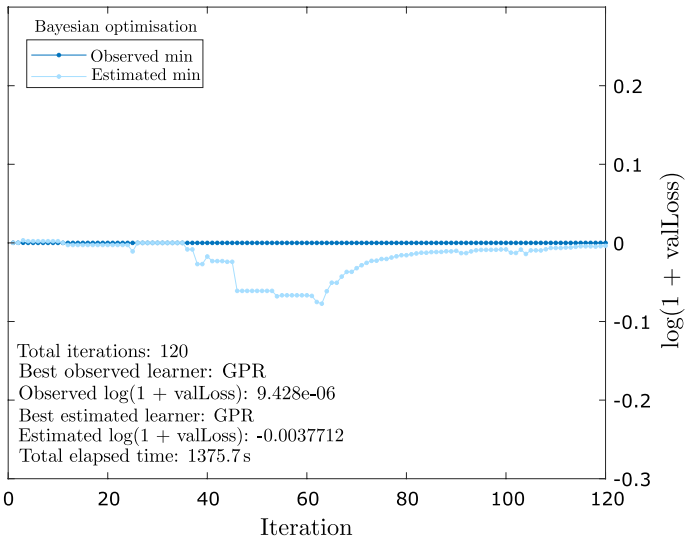

Figure 4. Curve illustrating the Bayesian optimisation process for the selection of the best machine learning model with optimal hyperparameters.

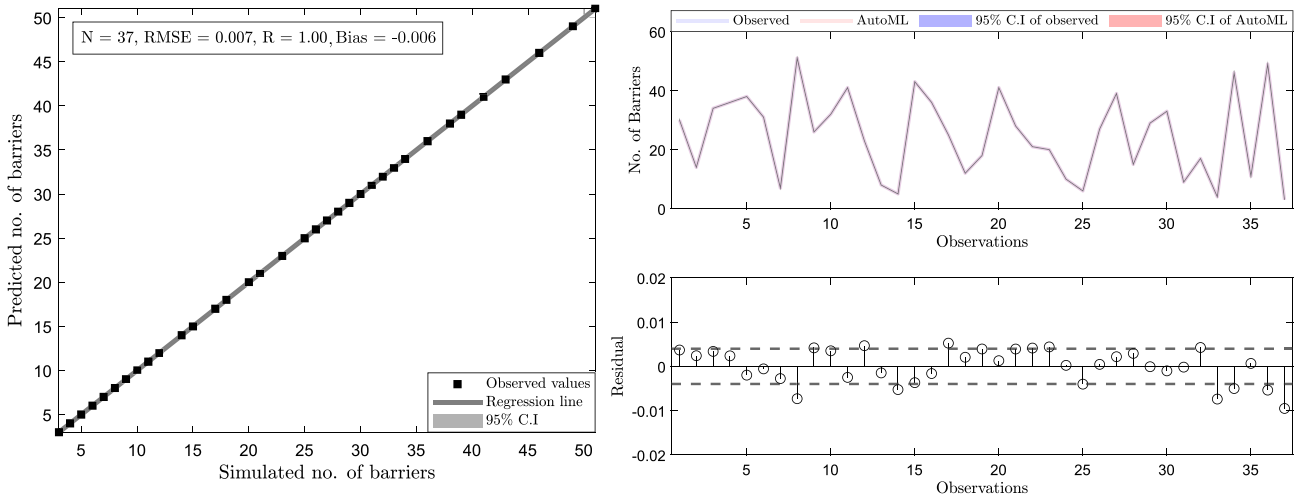

Figure 5. The left panel shows the linear regression plot between the predicted and observed responses. The top plot on the right panel shows the time series plot of the predicted and observed. The bottom panel shows the corresponding residuals. The dashed line in the residual plot shows the RMSE value.

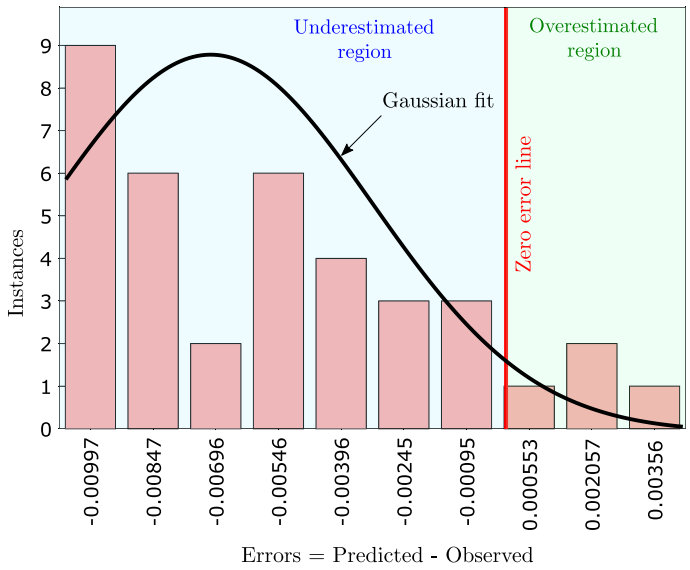

Figure 6. Error analysis using error histogram of 10 bins. The line in red shows the zero error line. The area to the left of the zero error line shows the underestimated region, and the area right to the zero error line shows the overestimated region.

Performance metrics	Algorithms					
	SVR	BDT	Bagging EL (random forest)	Boosting EL (LSBoost)	Kernel regression	Linear regression
R	0.93	0.81	0.93	0.73	0.91	0.94
RMSE	63.61	73.07	81.84	118.03	32.29	33.68
Bias	53.59	56.99	67.31	89.81	31.28	31.81
t (s)	95.3	111.3	103.4	107.7	43.01	36.7

Table 2. Performance of the other AutoML algorithms.

Performance metrics	FFNN	RNN	Exact RBN	RBN	GRNN
R	0.47	0.95	0.30	0.41	0.97
RMSE	36.96	14.92	107.95	161.11	64.61
Bias	21.47	71.06	86.21	139.23	60.18
t (s)	2.5	13.51	2.90	3.98	2.23

Table 3. Comparing the performance of the AutoML with the deep learning models.

	k-barriers		*k*-barrier coverage probability[21]		
Performance metrics	AutoML (This study)	LT-FS-ID[22]	GPR	S-GPR	C-GPR
R	1	0.98	0.85	0.64	0.79
RMSE	0.007	6.47	0.095	0.137	0.108
t (s)	0.73	0.65	8.16	7.79	9.51

Table 4. Comparing the results of AutoML with previous studies.

Discussion

We observed that the AutoML approach successfully selects the best machine learning model among a group of explainable machine learning algorithms (i.e., among SVR, GPR, BDT, bagging ensemble learning, boosting ensemble learning, kernel regression, and linear regression model) and optimised its hyperparameters. However, we have compared the AutoML derived results with the benchmark algorithms for an unbiased and fair evaluation of the proposed approach. We selected Feed-Forward Neural Network (FFNN)[70], Recurrent Neural Network (RNN)[71], Radial Basis Neural Networks (RBN)[72], Exact RBN[73], and Generalised Regression Neural Network (GRNN)[74] as the benchmark algorithms. We selected these algorithms because they are frequently used in diverse applications such as remote sensing, blockchain, cancer diagnosis, precision medicine, decease prediction, self-driving cars, streamflow forecasting, and speech recognition; hence have high generalisation capabilities[37,75–77]. In doing so, we trained these algorithms over the same datasets. We found that the AutoML outperforms all the deep learning benchmark algorithms (Table 3). Among the benchmark algorithms GRNN performs the best (with R = 0.97, RMSE = 64.61, Bias = 60.18, and computational time complexity, t = 2.23 s). Surprisingly, all the benchmark algorithms have a high positive bias value. It indicates that these models highly overestimate the number of required *k*-barriers. We have also compared the performance of the AutoML with previous studies[21,22] for the prediction of *k*-barriers and *k*-barrier coverage probability (Table 4).

Further, we also tested the performance of the AutoML approach over the publicly available intrusion detection dataset[22]. In a recent study, Singh et al.[22] have proposed a log-transformed feature scaling based algorithm (i.e., LT-FS-ID) for intrusion detection considering uniform node distribution scenario. We downloaded the datasets and applied the proposed AutoML approach to them. In doing so, we iterated the AutoML for 120 iterations using the Bayesian optimisation to obtain the best optimised machine learning model. We found that AutoML approach perform well over the dataset (with R = 0.92, RMSE = 30.59, and Bias = 18.13). Interestingly, the same GPR algorithms emerges as the best learner algorithms with a optimised sigma = 0.33. It highlights the potential of the GPR algorithm for intrusion detection, which becomes more apparent from the recently published literature's[21,78].

The proposed AutoML approach for estimating the *k*-barriers for fast intrusion detection and prevention is highly user-friendly and provides a fast solution. It reduces the confusion of selecting the best-performing algorithm by automating the process. Further, it also overcomes the limitation of the LT-FS-ID algorithm[22]. LT-FS-ID algorithm only works if the input predictors are a positive real number. It will not work if any input predictors contain zero (or negative values). Although the AutoML approach gives the best result, its performance will hamper with the sensor aging. In other words, with the aging effect in the sensors, the quality of the data recorded by the sensor may change drastically (i.e., datasets become dynamic), resulting in performance degradation. In such a situation, retraining the proposed model will solve the problem.

Conclusion

In this study, we proposed a robust AutoML approach to estimate the accurate number of *k*-barriers required for fast intrusion detection and prevention using WSNs over a rectangular RoI considering the Gaussian distribution of the node deployment. We found that the synthetic predictors (i.e., the area of the RoI, sensing range of the sensor node, transmission range of the sensor node, and the number of sensors) extracted through Monte Carlo simulations successfully mapped with the *k*-barriers. Among these predictors, the transmission range of the sensor emerges as the most relevant predictor, and the sensing range of the sensor emerges as the least relevant predictor. In addition to this, we observed that only the area of the RoI has a slightly negative impact on the response variable. We then iteratively run the AutoML algorithms to obtain the best machine learning model among the explainable machine learning model using Bayesian optimisation techniques. We found that

the AutoML algorithm selects the GPR algorithm as the best machine learning model to map the required *k*-barriers accurately. We evaluated the potential of the GPR algorithm over unseen test datasets. We found that the AutoML elected algorithm performs exceptionally well on the test datasets.

We further compared the AutoML results with the benchmark algorithms for a more reliable and robust conclusion. We found that AutoML outperforms all the benchmark algorithms in terms of accuracy. For more generalisation of this approach, we tested the efficacy of the AutoML over the publicly available datasets on intrusion detection using WSNs, and we found a similar performance. This study is a step towards a cost-efficient approach for fast intrusion detection and prevention using explainable machine learning models.

Data availability

The datasets generated during and/or analysed during the current study can be made available from the corresponding author on a reasonable request.

Code availability

The computer algorithms originated during the current study can be made available from the corresponding author on a reasonable request.

Received: 30 January 2022; Accepted: 18 May 2022
Published online: 31 May 2022

References

1. Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C. & Faruki, P. Network intrusion detection for iot security based on learning techniques. *IEEE Commun. Surv. Tutor.* **21**, 2671–2701 (2019).
2. Wang, Y., Wang, X., Xie, B., Wang, D. & Agrawal, D. P. Intrusion detection in homogeneous and heterogeneous wireless sensor networks. *IEEE Trans. Mob. Comput.* **7**, 698–711 (2008).
3. Abduvaliyev, A., Pathan, A.-S.K., Zhou, J., Roman, R. & Wong, W.-C. On the vital areas of intrusion detection systems in wireless sensor networks. *IEEE Commun. Surv. Tutor.* **15**, 1223–1237 (2013).
4. Butun, I., Morgera, S. D. & Sankar, R. A survey of intrusion detection systems in wireless sensor networks. *IEEE Commun. Surv. Tutor.* **16**, 266–282 (2013).
5. Resende, P. A. A. & Drummond, A. C. A survey of random forest based methods for intrusion detection systems. *ACM Comput. Surv.* **51**, 1–36 (2018).
6. Ali, A., Ming, Y., Chakraborty, S. & Iram, S. A comprehensive survey on real-time applications of wsn. *Future Internet* **9**, 77 (2017).
7. Singh, A., Sharma, S. & Singh, J. Nature-inspired algorithms for wireless sensor networks: A comprehensive survey. *Comput. Sci. Rev.* **39**, 100342 (2021).
8. Amutha, J., Sharma, S. & Nagar, J. Wsn strategies based on sensors, deployment, sensing models, coverage and energy efficiency: Review, approaches and open issues. *Wirel. Pers. Commun.* **111**, 1089–1115 (2020).
9. Nagar, J., Chaturvedi, S. K. & Soh, S. An analytical model to estimate the performance metrics of a finite multihop network deployed in a rectangular region. *J. Netw. Comput. Appl.* **149**, 102466 (2020).
10. Yang, T., Mu, D., Hu, W. & Zhang, H. Energy-efficient border intrusion detection using wireless sensors network. *EURASIP J. Wirel. Commun. Netw.* **2014**, 1–12 (2014).
11. Raza, F., Bashir, S., Tauseef, K. & Shah, S. Optimizing nodes proportion for intrusion detection in uniform and gaussian distributed heterogeneous wsn. In *2015 12th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, 623–628 (IEEE, 2015).
12. Arfaoui, I., Boudriga, N., Trimeche, K. & Abdallah, W. Wsn-based border surveillance systems using estimated known crossing paths. In *Proceedings of the 15th International Conference on Advances in Mobile Computing and Multimedia*, 182–190 (2017).
13. Singh, R. & Singh, S. Smart border surveillance system using wireless sensor networks. *Int. J. Syst. Assur. Eng. Manage.* **20**, 1–15 (2021).
14. Sharma, M. & Kumar, C. Machine learning-based smart surveillance and intrusion detection system for national geographic borders. In *Artificial Intelligence and Technologies* 165–176 (Springer, 2022).
15. Mishra, P., Varadharajan, V., Tupakula, U. & Pilli, E. S. A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE Commun. Surv. Tutor.* **21**, 686–728 (2018).
16. Sun, Z., Xu, Y., Liang, G. & Zhou, Z. An intrusion detection model for wireless sensor networks with an improved v-detector algorithm. *IEEE Sens. J.* **18**, 1971–1984 (2017).
17. Ghosh, K., Neogy, S., Das, P. K. & Mehta, M. Intrusion detection at international borders and large military barracks with multi-sink wireless sensor networks: An energy efficient solution. *Wirel. Pers. Commun.* **98**, 1083–1101 (2018).
18. Benahmed, T. & Benahmed, K. Optimal barrier coverage for critical area surveillance using wireless sensor networks. *Int. J. Commun. Syst.* **32**, e3955 (2019).
19. Arfaoui, I. & Boudriga, N. A border surveillance system using wsn under various environment characteristics. *Int. J. Sens. Netw.* **30**, 263–278 (2019).
20. Sharma, S. & Nagar, J. Intrusion detection in mobile sensor networks: A case study for different intrusion paths. *Wirel. Pers. Commun.* **115**, 2569–2589 (2020).
21. Singh, A., Nagar, J., Sharma, S. & Kotiyal, V. A gaussian process regression approach to predict the k-barrier coverage probability for intrusion detection in wireless sensor networks. *Expert Syst. Appl.* **172**, 114603 (2021).
22. Singh, A., Amutha, J., Nagar, J., Sharma, S. & Lee, C.-C. Lt-fs-id: Log-transformed feature learning and feature-scaling-based machine learning algorithms to predict the k-barriers for intrusion detection using wireless sensor network. *Sensors*https://doi.org/10.3390/s22031070 *(2022).*
23. Otoum, S., Kantarci, B. & Mouftah, H. T. On the feasibility of deep learning in sensor network intrusion detection. *IEEE Netw. Lett.* **1**, 68–71 (2019).
24. Schmidt, J., Marques, M. R., Botti, S. & Marques, M. A. Recent advances and applications of machine learning in solid-state materials science. *NPJ Comput. Mater.* **5**, 1–36 (2019).
25. Nikolenko, S. I. *et al.* Synthetic data for deep learning. arXiv:1909.11512 (arXiv preprint) (2019).
26. Chen, R. J., Lu, M. Y., Chen, T. Y., Williamson, D. F. & Mahmood, F. Synthetic data in machine learning for medicine and healthcare. *Nat. Biomed. Eng.* **20**, 201–5 (2021).
27. Rankin, D. *et al.* Reliability of supervised machine learning using synthetic data in health care: Model to preserve privacy for data sharing. *JMIR Med. Inform.* **8**, e18910 (2020).
28. Singh, A., Kotiyal, V., Sharma, S., Nagar, J. & Lee, C.-C. A machine learning approach to predict the average localization error with applications to wireless sensor networks. *IEEE Access* **8**, 208253–208263 (2020).

29. Abay, N. C., Zhou, Y., Kantarcioglu, M., Thuraisingham, B. & Sweeney, L. Privacy preserving synthetic data release using deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 510–526 (Springer, 2018).
30. Wang, D., Xie, B. & Agrawal, D. P. Coverage and lifetime optimization of wireless sensor networks with gaussian distribution. *IEEE Trans. Mob. Comput.* **7**, 1444–1458 (2008).
31. Wang, Y., Fu, W. & Agrawal, D. P. Gaussian versus uniform distribution for intrusion detection in wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.* **24**, 342–355 (2012).
32. Zou, Y. & Chakrabarty, K. Sensor deployment and target localization in distributed sensor networks. *ACM Trans. Embedd. Comput. Syst.* **3**, 61–91 (2004).
33. Mostafaei, H., Chowdhury, M. U. & Obaidat, M. S. Border surveillance with wsn systems in a distributed manner. *IEEE Syst. J.* **12**, 3703–3712 (2018).
34. Torres-Barrán, A., Alonso, Á. & Dorronsoro, J. R. Regression tree ensembles for wind energy and solar radiation prediction. *Neurocomputing* **326**, 151–160 (2019).
35. Friedman, J. H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **20**, 1189–1232 (2001).
36. Goldstein, A., Kapelner, A., Bleich, J. & Pitkin, E. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *J. Comput. Graph. Stat.* **24**, 44–65 (2015).
37. Singh, A., Gaurav, K., Rai, A. K. & Beg, Z. Machine learning to estimate surface roughness from satellite images. *Remote Sens.* **13**, 3794 (2021).
38. Guyon, I. *et al.* Design of the 2015 chalearn automl challenge. In *2015 International Joint Conference on Neural Networks (IJCNN)*, 1–8 (IEEE, 2015).
39. Guyon, I. *et al.* Automl challenge 2015: Design and first results. In *Proceedings of of AutoML* (2015).
40. Guyon, I. *et al.* A brief review of the chalearn automl challenge: Any-time any-dataset learning without human intervention. In *Workshop on Automatic Machine Learning*, 21–30 (PMLR, 2016).
41. He, Y. *et al.* Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 784–800 (2018).
42. Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M. & Hutter, F. Practical automated machine learning for the automl challenge 2018. In *International Workshop on Automatic Machine Learning at ICML*, 1189–1232 (2018).
43. He, X., Zhao, K. & Chu, X. Automl: A survey of the state-of-the-art. *Knowl.-Based Syst.* **212**, 106622 (2021).
44. Townes, F. W., Hicks, S. C., Aryee, M. J. & Irizarry, R. A. Feature selection and dimension reduction for single-cell rna-seq based on a multinomial model. *Genome Biol.* **20**, 1–16 (2019).
45. Vapnik, V. *et al.* Support vector method for function approximation, regression estimation, and signal processing. *Adv. Neural Inf. Process. Syst.* **20**, 281–287 (1997).
46. Saha, A. *et al.* Flood susceptibility assessment using novel ensemble of hyperpipes and support vector regression algorithms. *Water* **13**, 241 (2021).
47. Arifuzzaman, M., Aniq Gul, M., Khan, K. & Hossain, S. Application of artificial intelligence (ai) for sustainable highway and road system. *Symmetry* **13**, 60 (2021).
48. da Silva Santos, C. E., dos Santos Coelho, L. & Llanos, C. H. Nature inspired optimization tools for svms-niots. *MethodsX* **8**, 101574 (2021).
49. Zaghloul, M. S., Hamza, R. A., Iorhemen, O. T. & Tay, J. H. Comparison of adaptive neuro-fuzzy inference systems (anfis) and support vector regression (svr) for data-driven modelling of aerobic granular sludge reactors. *J. Environ. Chem. Eng.* **8**, 103742 (2020).
50. César de Sá, N., Baratchi, M., Hauser, L. T. & van Bodegom, P. Exploring the impact of noise on hybrid inversion of prosail rtm on sentinel-2 data. *Remote Sens.* **13**, 648 (2021).
51. Rasmussen, C. E. Gaussian processes in machine learning. In *Summer School on Machine Learning* 63–71 (Springer, 2003).
52. Asante-Okyere, S., Shen, C., Yevenyo Ziggah, Y., Moses Rulegeya, M. & Zhu, X. Investigating the predictive performance of gaussian process regression in evaluating reservoir porosity and permeability. *Energies* **11**, 3261 (2018).
53. Artime Ríos, E. M., Sánchez Lasheras, F., Suárez Sánchez, A., Iglesias-Rodríguez, F. J. & Seguí Crespo, M. D. M. Prediction of computer vision syndrome in health personnel by means of genetic algorithms and binary regression trees. *Sensors* **19**, 2800 (2019).
54. Kim, S.-H., Moon, I.-J., Won, S.-H., Kang, H.-W. & Kang, S. K. Decision-tree-based classification of lifetime maximum intensity of tropical cyclones in the tropical western north pacific. *Atmosphere* **12**, 802 (2021).
55. Perrone, M. P. & Cooper, L. N. *When networks disagree: Ensemble methods for hybrid neural networks*. Tech. Rep., Brown Univ Providence RI Inst for Brain and Neural Systems (1992).
56. Breiman, L. *Bagging Predictors (Technical Report 421)* (University of California, 1994).
57. Breiman, L. Stacked regressions. *Mach. Learn.* **24**, 49–64 (1996).
58. Erdal, H. & Karahanoğlu, İ. Bagging ensemble models for bank profitability: An emprical research on Turkish development and investment banks. *Appl. Soft Comput.* **49**, 861–867 (2016).
59. Freund, Y. *et al.* Experiments with a new boosting algorithm. In *icml* Vol. 96 148–156 (Citeseer, 1996).
60. Jung, C. High spatial resolution simulation of annual wind energy yield using near-surface wind speed time series. *Energies* **9**, 344 (2016).
61. Watson, G. S. Smooth regression analysis. *Sankhyā Indian J. Stat. Ser. A* **20**, 359–372 (1964).
62. Heo, G.-Y. Condition monitoring using empirical models: Technical review and prospects for nuclear applications. *Nucl. Eng. Technol.* **40**, 49–68 (2008).
63. Poole, M. A. & O'Farrell, P. N. The assumptions of the linear regression model. *Trans. Inst. Brit. Geograph.* **20**, 145–158 (1971).
64. Močkus, J. On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference* 400–404 (Springer, 1975).
65. Feurer, M. *et al.* Methods for improving Bayesian optimization for automl. In *Proceedings of the International Conference on Machine Learning* (2015).
66. Savaia, G. *et al.* Experimental automatic calibration of a semi-active suspension controller via Bayesian optimization. *Control. Eng. Pract.* **112**, 104826 (2021).
67. Pelikan, M., Goldberg, D. E., Cantú-Paz, E. *et al.* Boa: The Bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, vol. 1, 525–532 (Citeseer, 1999).
68. Shahriari, B., Swersky, K., Wang, Z., Adams, R. P. & De Freitas, N. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* **104**, 148–175 (2015).
69. Frazier, P. I. A tutorial on Bayesian optimization. arXiv:1807.02811 (arXiv preprint) (2018).
70. Fine, T. L. *Feedforward Neural Network Methodology* (Springer Science & Business Media, 2006).
71. Zaremba, W., Sutskever, I. & Vinyals, O. Recurrent neural network regularization. arXiv:1409.2329 (arXiv preprint) (2014).
72. Karayiannis, N. B. Reformulated radial basis neural networks trained by gradient descent. *IEEE Trans. Neural Netw.* **10**, 657–671 (1999).
73. Çivicioğlu, P., Alçı, M. & Bedok, E. Using an exact radial basis function artificial neural network for impulsive noise suppression from highly distorted image databases. In *International Conference on Advances in Information Systems*, 383–391 (Springer, 2004).
74. Specht, D. F. *et al.* A general regression neural network. *IEEE Trans. Neural Netw.* **2**, 568–576 (1991).
75. Guidotti, R. *et al.* A survey of methods for explaining black box models. *ACM Comput. Surv.* **51**, 1–42 (2018).

76. Xie, M., Li, H. & Zhao, Y. Blockchain financial investment based on deep learning network algorithm. *J. Comput. Appl. Math.* **372**, 112723 (2020).
77. Shrestha, A. & Mahmood, A. Review of deep learning algorithms and architectures. *IEEE Access* **7**, 53040–53065 (2019).
78. Nwakanma, C. I., Ahakonye, L. A. C., Lee, J.-M. & Kim, D.-S. Selecting gaussian process regression kernels for iot intrusion detection and classification. In *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, 462–465 (IEEE, 2021).

Acknowledgements
We want to acknowledge IISER Bhopal, Madhya Pradesh, India; Gautam Buddha University, Uttar Pradesh, India; IIT Kharagpur, West Bengal, India; MITS Gwalior, Madhya Pradesh, India; Fu Jen Catholic University, Taiwan; and Asia University, Taiwan, for providing institutional support.

Author contributions
A.S. developed the models, J.N. and J.A. extracted the datasets, S.S. and C.C.L. analysed the results. All the authors contributed to the writing and reviewed the manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to S.S. or C.-C.L.

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.