PENERBIT PRESS
UNIVERSITI TEKNOLOGI MARA

e-ISSN: 2785-924X

Available online at
https://jsst.uitm.edu.my/index.php/jsst

Journal of Smart Science and Technology 5(1) 2025, 17-39.

Journal of Smart Science and Technology

Indoor Occupancy Detection Using Machine Learning and Environmental Sensors

Akindele Segun Afolabi[1*], Olubunmi Adewale Akinola[2], Oyinlolu Ayomidotun Odetoye[3], Emmanuel Adetiba[4,5,6]

[1]*Department of Electrical and Electronics Engineering, University of Ilorin, Ilorin 240003, Nigeria*
[2]*Department of Electrical and Electronic Engineering, Federal University of Agriculture, Abeokuta 110111, Nigeria*
[3]*Department of Electrical and Information Engineering, Landmark University, Omu-Aran 251103, Nigeria*
[4]*Department of Electrical and Information Engineering, Covenant University, Ota, Ogun State 112233, Nigeria*
[5]*Covenant Applied Informatics and Communication Africa Center of Excellence, Covenant University, Ota Ogun 112233, Nigeria*
[6]*HRA Institute of System Science, Durban University of Technology, Durban 4000, South Africa*

Citation:
Afolabi, A. S., Akinola, O. A., Odetoye, O. A., & Adetiba, E. (2025). Indoor occupancy detection using machine learning and environmental sensors. *Journal of Smart Science and Technology*, 5(1), 17-39.

ARTICLE INFO

Article history:
Received 13 November 2024
Revised 02 February 2025
Accepted 20 February 2025
Published 31 March 2025

Keywords:
indoor occupancy detection
machine learning
data leakage
target leakage
random forest classifier
decision trees classifier

DOI:
10.24191/jsst.v5i1.101

ABSTRACT

Detecting the occupancy status of enclosed spaces has been immensely beneficial in the automated control of HVACs (heating, ventilation, and cooling systems), providing assistance to the elderly, healthcare provisioning, recognition of human activity, and others. As a result of these benefits, a plethora of machine learning-based solutions for occupancy detection has been developed in the literature. However, many of these solutions have poor prediction accuracies. Furthermore, it is necessary to develop models that are robust enough to achieve acceptable performance in situations where partial data from sensors are available. In this paper, we experimentally determined the Machine Learning (ML) models that are most robust for use in indoor occupancy detection. This is important because the activities of human subjects in an ML environment are capable of disrupting the data available to some deployed ML models, which might cause the performance of such models to drop. Hence, it is crucial to determine ML models that are robust against such disruptions. In this paper, three algorithms were developed: the first was for outlier removal from features, the second was for feature selection, and the third was for partial-features-availability-aware ML model selection. These algorithms were applied to data from environmental sensors such as temperature, humidity, carbon dioxide (CO_2), and light sensors, and afterward. The resulting data was used to train six different ML-based classifiers. The classifiers

[1*] Corresponding author. *E-mail address*: afolabisegun@unilorin.edu.ng
https://doi.org/10.24191/jsst.v5i1.101

considered in this paper were Logistic Regression (LR), Random Forest (RF), Decision Tree (DT), K-Nearest Neighbours (KNN), Support Vector Machines (SVM), and Gradient Boosting Machines (GBM). Simulation experiments revealed that only the RF and DT models are robust against the partial features availability problem, achieving at least 90% performance scores across all the considered metrics.

1 INTRODUCTION

Human occupancy detection, which involves detecting the presence of people in a building, is a vital task that has, in recent times, been shown to have numerous benefits. For example, it has encouraged the creation of several smart applications in a variety of relevant contexts such as automated control of heating, ventilation, and cooling systems (HVACs)[1–3] providing assistance to the elderly[4], healthcare provision[5,6], recognition of human activity[7,8] to name a few. Occupancy detection has also accelerated the development of smart buildings[9,10]. Technologies for detecting occupancy and estimating the number of individuals in a space are roughly classified as: a) room installation devices[11,12], and b) body-worn devices[13]. In practice, constantly wearing gadgets is inconvenient, therefore, researchers have recently focused on systems that rely solely on devices installed within the room infrastructure[10,14].

In the built environment, there are two major techniques for determining occupants' presence and count[15]. The first method is the physics-based model, which predicts occupancy using estimated internal heat gain in rooms, considering factors such as size, layout, and building envelope. The second is the data-driven model which detects user presence in buildings and is less complex than the physics-based models. However, generating the data-driven model requires a lengthy training process and a large amount of tagged occupancy data, which may be unavailable for diverse rooms and buildings.

One of the most important applications of occupancy detection systems is their use in smart buildings for the automatic control of appliances. Some occupancy characteristics that might be employed as key control inputs in a smart home include[16]: a) the presence of occupants can be utilized to switch active equipment on or off when the part of the evaluated building is inhabited or empty; b) the number of occupants can be utilized to accurately manage independent ventilation systems.

There have been many initiatives to monitor building occupancy using wired and wireless sensor networks. For example, some research studies combined motion sensor data with information from magnetic reed switches[17,18], while others used temperature sensor arrays[19], passive infrared (PIR) sensors[20], cameras[21], or ambient sensors[22] to estimate the room-level occupancy status (i.e., occupant presence or count). Also, non-environmental sensors like RFID (radio frequency identifier) sensors[23], ultrasonic sensors[24], pressure-based sensors[23], Doppler radar-based sensors[25], depth sensors[23], and radio frequency (Wi-Fi signal)[26–28] are used in applications for occupancy detection. Due to the randomness and complexity of occupancy, it can be quantified using a mathematical model. Existing work on occupancy models can be classified into two categories, that is, Mathematical model-based (probability) and Machine Learning (ML) model-based.

Mathematical model-based: The Mathematical model-based approach can be further classified into deterministic models and stochastic models. In this approach, deterministic scheduling uses long-term monitoring and occupancy statistics to generate probability distributions, which aid in estimating group activity patterns and developing probabilistic models[29]. Researchers explore the relationship between occupant behaviour and environmental stimulus using stochastic processes, and treat room occupancy as a random variable with the probability of occupancy state computed at each time point[30,31].

https://doi.org/10.24191/jsst.v5i1.101

Machine Learning model-based: This is a more realistic and accurate approach that employs data mining. It can predict future development paths using artificial intelligence methods such as ANNs (Artificial Neural Networks), decision trees, Support Vector Machines (SVM), logistic regression, and Bayesian networks[26,32,33].

2 RELATED WORKS

Odetoye et al.[34] considered a PIR sensor-based occupancy detection strategy for energy management. However, PIR sensors struggle to accurately identify stationary humans, leading to false negatives. To improve accuracy, artificial intelligence methods can be applied to sensor data[35,36]. In recent times, machine learning and artificial intelligence have been applied to problems in diverse sectors including health[37], agriculture[38], and banking[39] just to name a few. It has also been extended to occupancy detection as several studies[26,32,33] on occupancy detection have explored the use of artificial neural networks.

Feng et al.[40] proposed an ANN-based occupancy detection approach in which a Convolutional Neural Network (CNN) and a Long Short-Term Memory (LSTM) network were employed to create a deep learning model. Praise et al.[41] used SVM to develop an occupancy detection model using temperature, humidity, carbon dioxide (CO_2), sound, pressure, and PIR motion sensors. In their work, two sensor nodes delivered data in real-time to a Raspberry pi gateway and cloud for analytics, and an SVM model classified data into "occupied" and "not occupied" states using a hyper-plane.

Many studies have considered the use of decision trees[27,31] in occupancy detection. For example, Mahmud et al.[36] used a decision tree classifier for occupancy detection. Data from two sensors (PIR and active infrared sensors) were used to train their model. However, this classifier exhibited poor accuracy and high variance, leading to incorrect data categorisation. Particularly, the accuracy achieved by the classifier was 67.6%. In an attempt to boost this accuracy, a decision tree classifier was hybridised with a K-Nearest Neighbour classifier, but this did not yield any improvement.

An LSTM model was employed for detecting occupancy by Khalil et al.[42]. The model was applied to three office spaces in an educational institution case study. The final findings showed that stacked LSTM with a transfer learning architecture was able to predict occupancy, but it still needed accuracy improvement. Specifically, the accuracy achieved was 71%.

Fayed et al.[43] used the Neutrosophy method to improve the accuracy of different occupancy classifiers which include Linear Discriminant Analysis (LDA), K-Nearest Neighbours (KNN), Naive Bayes (NB), Support Vector Machine (SVM), and Random Forest (RF). Interestingly, their approach successfully increased the accuracies of each of the classifiers. However, several existing machine learning-based studies on occupancy suffer from performance issues, with low accuracy being a prevalent shortcoming. This is evidenced in the summary of some existing works in Table 1.

Apart from issues related to low accuracy, another challenge that may arise in ML-based occupancy detection models is the issue of data leakage. This factor is sometimes ignored in machine learning-based experiments. Data leakage can lead to inflated results for trained ML models. Leakage is not a novel issue in machine learning. However, according to Kapoor and Narayanan[48], there is no thorough investigation on leakage in ML-based science, therefore, mitigation solutions for data leakage in scientific ML applications are still not well understood.

https://doi.org/10.24191/jsst.v5i1.101

Table 1. A summary of existing works on occupancy detection

Author(s)	Methodology	Type of enclosed space	Sensor(s) used	Performance	Limitation(s)
Mutis et al.[44]	Occupants were detected and counted using deep Convolutional Neural Network	Office	Surveillance camera	Accuracy = 84%	Relatively low accuracy
Kim et al.[45]	Activities of occupants were detected using Long Short Term Memory Networks and Convolutional Neural Network	Residential	Acoustic sensor	Precision = 78% F1-score =83.9% Recall = 90.8%	Relatively low precision and F1-scores
Elkhoukhi et al.[16]	Occupants were detected and counted using Long Short Term Memory Networks	Laboratory	CO_2 sensor	Accuracy = 70%	Relatively low accuracy
Ng et al.[46]	Occupants were detected using RF fingerprinting	Laboratory	Bluetooth low energy	Accuracy = 90%	Required Cloud processing
Mahmud et al.[36]	Occupants were detected using Decision Tree and K-Nearest Neighbours	Office	Passive infrared sensor and active infrared sensor	Accuracy = 67%	Relatively low accuracy
Parise et al.[41]	Occupants detected using support vector machines classifier	Classroom	Temperature, humidity, pressure, PIR, acoustic sensors	Accuracy = 96%	Cloud Processing required
Wang et al.[47]	Occupants detected and counted using the random forest classifier	Office	Wi-Fi device	Accuracy = 84%	Relatively low accuracy

Kapoor and Narayanan[48] described data leakage as a flaw in ML that leads to overoptimistic results. One type of data leakage is target leakage[49], that occurs when a feature used in developing an ML model is unavailable when making predictions. Due to the human activities in spaces where environmental sensors are deployed, sensor data may be disrupted, leading to the non-availability of certain features which have been previously used for training prior to model deployment, therefore, causing a condition which we refer to as "partial feature availability" problem, a condition similar to target leakage. Consequently, this could lead to a decline in the model's performance. To address this, a potential partial feature availability problem in this paper is anticipated and thus, the three algorithms aimed at ensuring acceptable model performance are developed.

3 MATERIALS AND METHODS

The general framework of the proposed occupancy detection system is shown in Fig. 1. The framework consists of three blocks, namely, a) an array of sensors, b) the prediction models, and c) the controller hardware. The input into the system comprises environmental parameters that jointly represent the occupancy status of the room. These parameters include the concentration of CO_2 in the air, environmental temperature, and environmental humidity. It is important to highlight that the presence of a human in a room alters these environmental parameters, and if the pattern of this alteration is correctly explored, an accurate detection of human presence can be achieved.

https://doi.org/10.24191/jsst.v5i1.101

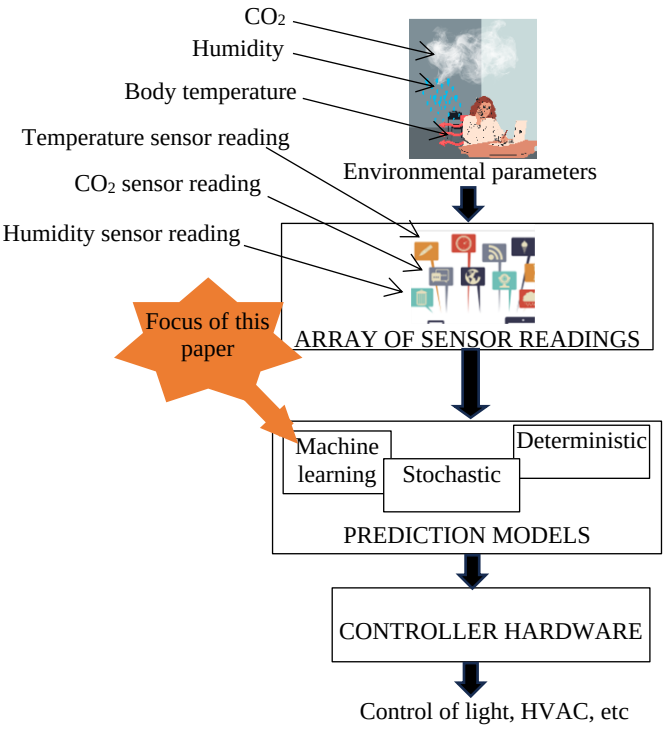


Fig. 1. Framework diagram.

The first block of Fig. 1 is where environmental parameters are sensed by a few environmental sensors such as CO_2, temperature, and humidity sensors. The readings of the sensors are compiled into a data bank for processing. The readings are taken periodically from all the sensors and they are grouped to form an input vector. Each vector is annotated with the actual occupancy status of the room which falls into two categories, that is occupied (the events when a person was in the room) and not occupied (the events when nobody was in the room). The second block (prediction models) is where the measured data is processed to determine if the room is occupied or not. As mentioned in the introduction, different types of prediction models for occupancy exist, but the focus of this paper is on the ML models. Hence, the data is fed into ML algorithms to train ML models that could subsequently be used in occupancy prediction when unannotated new data are supplied to them. The third block is the controller hardware block which controls the operation of attached electrical appliances based on the input it receives from the prediction models block.

3.1 Model formulation for sensor data

Consider a vector of sensors *S*, written as in Equation 1,

$$S = [s_1, s_2, s_3, \cdots, s_I] \tag{1}$$

that are used for environmental status measurements, for example, temperature, humidity level, CO_2 level, etc. Let the measurement recorded by one sensor $s \in S$ at instance k be denoted by $m_i^k \in M^k$ and the total number of sensors, which is the cardinality of *S* be denoted by *I*. In addition, let the set of measurements recorded by one sensor s_i for all the instances *k* be denoted by M_i. Then, the vector of measurement readings from all the sensors at one instance *k* is written as in Equation 2, and the vector of measurement readings from one sensor for all instances *k* is written as in Equation 3.

$$M^k = [m_1^k, m_2^k, m_3^k, \cdots, m_I^k], \ k = 1,2, \cdots, K \tag{2}$$

https://doi.org/10.24191/jsst.v5i1.101

$$M_i = [m_i^1, m_i^2, m_i^3, \cdots, m_i^K], \quad i = 1,2, \cdots, I \tag{3}$$

Also, the concatenation of the measurements read by all sensors for all time instances is given as M and is written as in Equation 4, where $\|$ is a vector concatenation operation and $M^k \subset M$. The task is to build a binary classifier that maps each M^k to y_k as shown in Equation 5, where $y_k \in \{0,1\}$ and y_k is the occupancy status of an indoor environment at instance *k*.

$$M = \Big\| _{k=1}^{K} M^k \tag{4}$$

$$c: M^k \rightarrow y_k \tag{5}$$

3.2 Removal of outliers

The presence of outliers in the measurement data introduces points of noisy large gradients which negatively affects the performance of some ML algorithms. There is, therefore, a need to remove them from the rest of the data. Let the mean of the measurement of a sensor s_i taken over all the time instances be given as in Equation 6.

$$\mu_{s_i} = \frac{1}{K} \sum_{k=1}^{K} m_i^k , \quad \forall s_i \epsilon S \tag{6}$$

To eliminate outliers from the measurement data, a 99.7 percentile clipper is applied. The clipper discards all outliers, such that all values of measurements m_k^i from sensor s_i are restricted to the range as shown in Equation 7, where δ_{s_i} is the standard deviation of measurements read by sensor s_i written as in Equation 8.

$$\mu_{s_i} - 3\delta_{s_i} \leq m_i^k \leq \mu_{s_i} + 3\delta_{s_i} \quad \forall s_i \in S, \forall m_i^k \in M^k, k = 1,2,3, \cdots, K \tag{7}$$

$$\delta_{s_i} = \sqrt{\frac{1}{K} \sum_{k=1}^{K} \left(m_i^k - \mu_{s_i}\right)^2}, \qquad \forall s_i \in S \tag{8}$$

Algorithm 1 describes the 99.7 percentile clipper.

Algorithm 1: The 99.7 percentile clipper algorithm

Input: M which contains outliers
Output: M that is void of outliers
Initialize: X = empty vector ϕ
for k = 1 to total number of instances K do
  for i = 1 to total number of sensors I do
    If (measurement $m_i^k <$ (mean $\mu_{s_i} - 3 \times$ standard deviation δ_{s_i})) **then**
      update $X \leftarrow X \| M^k$;
    else if (measurement $m_i^k >$ (mean $\mu_{s_i} + 3 \times$ standard deviation δ_{s_i})) **then**
      update $X \leftarrow X \| M^k$;
    end if
  end for
end for
$M \leftarrow M \backslash X$;
return M

https://doi.org/10.24191/jsst.v5i1.101

3.3 Feature selection

In this study, a correlation threshold ρ_{th} is applied to determine the features set that is used to train ML models. Let $\rho_{i,j}$ denote the correlation between the measurement data read by two sensors $s_i, s_j \in S$ which is computed as[50]:

$$\rho_{i,j} = \frac{K\sum_{k=1}^{K} m_i^k m_j^k - \left(\sum_{k=1}^{K} m_i^k\right)\left(\sum_{k=1}^{K} m_j^k\right)}{\sqrt{\left[K\sum_{k=1}^{K}(m_i^k)^2 - \left(\sum_{k=1}^{K} m_j^k\right)^2\right]\left[K\sum_{k=1}^{K}\left(m_j^k\right)^2 - \left(\sum_{k=1}^{K} m_i^k\right)^2\right]}} \tag{9}$$

If the value of $\rho_{i,j}$ for any two features exceeds a threshold ρ_{th}, one of the features is removed from the features set while the other is retained. In this context, a feature implies the data read by a single sensor. Algorithm 2 shows how feature correlation is used to select features for ML model training.

Algorithm 2: The feature selection algorithm

Input: Features M and correlation threshold ρ_{th}
Output: Features F_ρ which contains only low-correlation features
Initialize: F_ρ = empty vector ϕ
1: **for** i = 1 to total number of sensors I - 1 **do**
2: threshold exceeded = **FALSE**
3: **for** $j = i + 1$ to total number of sensors I **do**
4: $\rho_{i,j} = \frac{K\sum_{k=1}^{K} m_i^k m_j^k - \left(\sum_{k=1}^{K} m_i^k\right)\left(\sum_{k=1}^{K} m_j^k\right)}{\sqrt{\left[K\sum_{k=1}^{K}(m_i^k)^2 - \left(\sum_{k=1}^{K} m_j^k\right)^2\right]\left[K\sum_{k=1}^{K}\left(m_j^k\right)^2 - \left(\sum_{k=1}^{K} m_i^k\right)^2\right]}}$
5: **if** correlation coefficient $\rho_{i,j} > \rho_{\text{th}}$ **then**
6: threshold exceeded = **TRUE**
7: **end if**
8: **end for**
9: **if** threshold exceeded = **FALSE** **then**
10: $F_\rho \leftarrow F_\rho \,||\, M_i$
11: **end if**
12: **end for**
13: **return** F_ρ

The initialization step of Algorithm 2 is used to create an empty vector F_ρ which will eventually be populated with selected features to be used to train machine learning algorithms. Line 1 of the algorithm contains counter i that selects a feature while Line 3 contains another counter j that selects another feature. Line 2 contains a Boolean variable (threshold exceeded) which is used to determine whether a feature should be included among the features to be used for training a machine learning model or not. Line 4 computes the Pearson correlation coefficient of the selected features which involves looping through all the K instances in each of the two selected features. The result of Line 4 is $\rho_{i,j}$ which is the correlation between the two features which is compared with a threshold value ρ_{th} in Line 5. If the computed value of $\rho_{i,j}$ is greater than the set threshold ρ_{th}, then the Boolean variable, threshold exceeded, is set to TRUE in Line 6. In Line 9, the Boolean variable threshold exceeded is examined, and if it is FALSE, then vector M_i, which contains all the instances of the i^{th} feature is included in F_ρ in Line 10, otherwise it is not included. This is repeated until the correlation between each feature and all other features has been computed and any time the correlation threshold is not exceeded, the content of F_ρ is updated in Line 10. Finally, the selected features stored in vector F_ρ are returned in Line 13.

In this paper, we preempt a potential partial feature availability and develop a method to address it. Particularly, our approach is to ameliorate the detrimental impact of a potential partial availability of

https://doi.org/10.24191/jsst.v5i1.101

features on the performance of a model. To achieve this, we consider that features that are strongly correlated with the target will have the most detrimental impact on model performance if they become unavailable. This is due to the behaviour of the machine learning optimization process which prioritizes features that are highly correlated with the target as stronger predictors[51]. This happens because features that are highly correlated with the target are relatively easier to learn for predicting the target[51]. Therefore, the proactive exclusion of features that are strongly correlated with the target from the features set would reduce the detrimental impact of the partial availability of features. Hence, Algorithm 2 considers the correlation between each feature and the target variable (ground truth) such that, if the correlation exceeds a threshold value, the feature is removed from the feature set.

3.4 Partial feature availability-aware ML model selection

Consider a set of ML models such as Logistic Regression (LR), Random Forest (RF), Decision Tree (DT), K-Nearest Neighbours (KNN), Support Vector Machines (SVM), and Gradient Boosting Machines (GBM) models; these models can be grouped into a row vector, *Mod*, having *W* elements as shown in Equation 10.

$$Mod = [\text{LR, RF, DT, KNN, SVM, GBM}]. \tag{10}$$

For each of the models in the vector *Mod*, the corresponding performances in terms of accuracy, precision, recall, and F1-score[52] are computed and combined to form vectors such as, $[Acc_w]_{w=1}^{W}$, $[Pre_w]_{w=1}^{W}$, $[Rec_w]_{w=1}^{W}$, and$[F1_w]_{w=1}^{W}$, respectively. Here, w = 1,2,3, …, W denote ML models LR, RF, DT, KNN, SVM, and GBM as indicated in Equation 10. For example, Acc_1 is the accuracy of the LR model, Acc_2 is the accuracy of the RF model, and so on. Likewise, Pre_1 is the precision of the LR model, Pre_2 is the precision of the RF model, and so on. As will be shown in Algorithm 3, the four performance model vectors $[Acc_w]_{w=1}^{W}$, $[Pre_w]_{w=1}^{W}$, $[Rec_w]_{w=1}^{W}$, and $[F1_w]_{w=1}^{W}$, are used to determine the elements of partial features availability candidacy binary vectors $[Mod_w^{\text{Acc}}]_{w=1}^{W}$, $[Mod_w^{\text{Pre}}]_{w=1}^{W}$, $[Mod_w^{\text{Rec}}]_{w=1}^{W}$, and$[Mod_w^{\text{F1}}]_{w=1}^{W}$, respectively depending on whether or not an ML model's performance reaches a minimum threshold. Let P_{th} denotes the minimum performance threshold required of each model for such model to be considered as a candidate partial-features-robust model. The candidacy of the model is determined using Equation 11 to 14. Specifically, it is determined and binarised based on accuracy (Equation 11), precision (Equation 12), recall (Equation 13), and *F1*-score (Equation 14), where w = 1,2, …, W.

$$Mod_{\text{w}}^{\text{Acc}} = \begin{cases} 1 & \text{if } Acc_W \geq P_{\text{th}} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$Mod_{\text{w}}^{\text{Pre}} = \begin{cases} 1 & \text{if } Pre_W \geq P_{\text{th}} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

$$Mod_{\text{w}}^{\text{Rec}} = \begin{cases} 1 & \text{if } Rec_W \geq P_{\text{th}}\,1 \\ 0 & \text{otherwise} \quad 0 \end{cases} \tag{13}$$

$$Mod_{\text{w}}^{\text{F1}} = \begin{cases} 1 & \text{if } F1_W \geq P_{\text{th}} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

Let $\left[Mod_w^{\text{T\_rob}}\right]_{w=1}^{W}$ denotes a binary vector that indicates the partial features robustness status of each model; a model element with a value of "1" implies that the model is robust against partial feature availability problem, while a value of "0" implies otherwise. The partial features robustness of each model is determined as in Equation 15, where w = 1,2, …, W and notation "&" represents a bitwise AND operator.

https://doi.org/10.24191/jsst.v5i1.101

$$Mod_w^{\text{T\_rob}} = \begin{cases} 1 & \text{if } (Mod_{\text{w}}^{\text{Acc}} \,\&\, Mod_{\text{w}}^{\text{Pre}} \,\&\, Mod_{\text{w}}^{\text{Rec}} \,\&\, Mod_{\text{w}}^{\text{F1}}) = 1 \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

Equation 15 implies that if a model has all its performance values exceeding or equal to a minimum threshold P_{th}, the model is considered robust against partial feature availability problems and vice versa. The procedure for determining the partial features robustness status of an ML model is shown in Algorithm 3.

Algorithm 3: The partial feature availability-aware ML model selection algorithm

Input: $[Acc_w]_{w=1}^{W}$, $[Pre_w]_{w=1}^{W}$, $[Rec_w]_{w=1}^{W}$, $[F1_w]_{w=1}^{W}$

Output: $\left[Mod_w^{\text{T\_rob}}\right]_{w=1}^{W}$

Initialize: $\left[Mod_w^{\text{T\_rob}}\right]_{w=1}^{W} = \emptyset$, $[Mod_w^{\text{Acc}}]_{w=1}^{W} = \emptyset$, $[Mod_w^{\text{Pre}}]_{w=1}^{W} = \emptyset$, $[Mod_w^{\text{Rec}}]_{w=1}^{W} = \emptyset$, $[Mod_w^{\text{F1}}]_{w=1}^{W} = \emptyset$

Binarize model performance vectors:

1: **for** w = 1 to W **do**
2: **If** $Acc_{\text{w}} \geq P_{\text{th}}$ **then**
3: $Mod_w^{\text{Acc}} = 1$
4: **else**
5: $Mod_w^{\text{Acc}} = 0$
6: **end if**
7:
8: **If** $Pre_{\text{w}} \geq P_{\text{th}}$ **then**
9: $Mod_w^{\text{Pre}} = 1$
10: **else**
11: $Mod_w^{\text{Pre}} = 0$
12: **end if**
13:
14: **If** $Rec_{\text{w}} \geq P_{\text{th}}$ **then**
15: $Mod_w^{\text{Rec}} = 1$
16: **else**
17: $Mod_w^{\text{Rec}} = 0$
18: **end if**
19:
20: **If** $F1_{\text{w}} \geq P_{\text{th}}$ **then**
21: $Mod_w^{\text{F1}} = 1$
22: **else**
23: $Mod_w^{\text{F1}} = 0$
24: **end if**
25:
26: **Determine the partial features robustness status:**
27: **for** w = 1 to W **do**
28: **if** $Mod_w^{\text{Acc}} = 1$ **then**
29: **if** $Mod_w^{\text{Pre}} = 1$ **then**
30: **if** $Mod_w^{\text{Rec}} = 1$ **then**
31: **if** $Mod_w^{\text{F1}} = 1$ **then**
32: $Mod_w^{\text{T\_rob}}$
33: **else**

https://doi.org/10.24191/jsst.v5i1.101

$Mod_w^{T\_rob} = 0$
end if
else
$Mod_w^{T\_rob} = 0$
end if
else
$Mod_w^{T\_rob} = 0$
end if
else
$Mod_w^{T\_rob} = 0$
end if
end for
return $\left[Mod_w^{T\_rob}\right]_{w=1}^{W}$

4 PERFORMANCE EVALUATION

An experiment was set up and a publicly available occupancy dataset[53] was used.

4.1 Data collection

A publicly available occupancy dataset[53] was used in this study. Temperature, humidity, light, humidity-ratio, and CO_2 levels were measured in an office area with approximate dimensions of 5.85 m × 3.50 m × 3.53 m (W × D × H)[54]. The data was collected using a microcontroller. A ZigBee radio was linked to it, and the data was sent to a recording station. A digital camera was used to assess whether or not the room was inhabited. Every minute, the camera time-stamped an image, which was then manually examined to categorize the data as occupied-room-status data or not-occupied-room-status data. Two of the datasets were obtained (datatraining.txt and datatest.txt)[53] and merged so that the final dataset had 10,808 entries. The dataset is unbalanced such that the "not-occupied" class constitutes the majority class. The stratified splitting method was used to divide the dataset into training and test sets in the proportions of 80% and 20% respectively while retaining the original class distribution. This division was done in order to ensure that the test set is not seen by the ML model during training and validation phases so that the patterns in the test data are not learned by the model. The minority class of the training set was unsampled, and the training set was further split into training and validation sets in the proportions of 75% and 25%, respectively. Upsampling was done because most machine learning models are sensitive to class imbalance.

4.2 Model training

Six different classifiers were employed for performance evaluation. A range of supervised classification algorithms was considered which included the following: Logistic Regression (LR), Random Forest (RF), Decision Tree (DT), K-Nearest Neighbours (KNN), Support Vector Machines (SVM), and Gradient Boosting Machine (GBM).

Logistic Regression

Logistic regression[55] is a statistical method for modelling binomial outcomes. The input may include one or more features (or variables) while the output of a binary logistic regression may be either 0 or 1, allowing for the binary classification of positive and negative classes. While it may not capture complex relationships as effectively as other models, its simplicity can yield strong baseline results, especially when the dataset is linearly separable. Some strengths of LR include:

(i) Simplicity and Interpretability: Because logistic regression is a linear model, it is simple to apply, analyse, and comprehend how features relate to the target class.

https://doi.org/10.24191/jsst.v5i1.101

(ii) Good for Linearly Separable Data: It is a powerful baseline model for classification issues and works well when the data is linearly separable.

Decision Trees

Decision Trees[56,57] are intuitive models that split data based on feature values to make predictions. The leaf nodes of the tree are the set of classes being predicted. Their ease of visualization and interpretation can significantly facilitate the comprehension of patterns. Pruning approaches or their usage in ensemble methods like as RF may improve their performance, as they may suffer from overfitting when employed alone. Some strengths of DT include:

(i) Interpretability: Decision trees give clear, visual decision rules that are simple to understand and explain.

(ii) Handles Nonlinear Data: They do not need data transformation to represent complex relationships and feature interactions.

Random Forest

Random Forest[56,57] is an ensemble learning method that builds multiple decision trees and merges them to improve accuracy and control overfitting. The idea behind RF is similar to asking several experts for their opinions and then using their votes to make a decision. RF is an example of ensemble ML, where individual ML models are first evaluated and then integrated into a single model that can often produce superior predictive performance than the individual models.

(i) Handles Nonlinear Relationships: Random Forest works very well with a variety of datasets because it can capture complex, nonlinear relationships between features.

(ii) Robustness and Overfitting Mitigation: It enhances model generalisation and lowers the danger of overfitting by averaging many decision trees.

K-Nearest Neighbours

Because KNN[55,56] is a non-probabilistic and non-parametric model, it is the top option for classification studies with no previous knowledge of data distribution. The similarity measure (a distance metric) serves as the basis for classification. Any unknown sample is categorised using the majority vote of its k closest neighbours. The complexity of KNN grows as dimensionality increases, hence dimensionality reduction procedures are undertaken before utilizing KNN to reduce the impacts of the curse of dimensionality.

(i) Non-parametric Nature: KNN can adapt to different datasets since it makes no assumptions about the underlying data distribution.

(ii) Flexibility: It can directly handle multiclass classification and does well on smaller datasets with well-separated classes.

Support Vector Machines

The main goal of SVM[55–57] classification technique is to create functions that locate the best hyperplanes to separate the various classes in training data. This method could be regarded as an 'optimization' method, in which the greater margin between distinct classes in the training data yields the best hyperplanes. Since SVM may adopt generalisation qualities, overfitting is successfully prevented in the training phase.

(i) Effective in High-Dimensional Spaces: SVM works well in high-dimensional spaces when there are many features relative to the number of samples.

(ii) Kernel Trick for Nonlinear Data: By using kernel functions like the radial basis function (RBF), SVM is able to handle data that is nonlinearly separable.

https://doi.org/10.24191/jsst.v5i1.101

Gradient Boosting Machines

GBM is an ensemble technique that builds models sequentially, focusing on correcting errors made by previous models. This method generally provides superior predictive accuracy compared to simpler models due to its ability to capture complex relationships within data.

(i) Handles Complex Data Well: GBM is able to capture complex patterns because it builds models iteratively, learning from past mistakes.

(ii) Customizable and Versatile: It is versatile and customizable, allowing for the fine-tuning of parameters, such as the number of estimators and learning rate, to maximize performance for particular datasets.

Each of the discussed ML algorithms has its own strength and uniqueness. They, therefore, make a good combination from which our proposed Algorithms 3 can select based on their performances in order to establish a robust ML model for partial feature availability scenarios.

4.3 Simulation settings

The correlation threshold between features was set to 0.8. If two features had correlation values of 0.8 or above, one of them was removed from the feature set. The correlation threshold between features and the target variable was also set to 0.8. The partial features robustness threshold (P_{th}) was set to 90%. The experiment was conducted on a PC with a Core i3 processor running Windows 10 operating system and having 8 Gigabyte RAM. Python scripts were written to execute the 3 algorithms that were developed.

The hyperparameters shown in Table 2 were selected for training the machine learning models and no attempt was made to optimize them.

Table 2. Hyperparameters of machine learning models and their values

Models	Hyperparameters and values			
LR	Optimization solver = ‘lbfgs’	Maximum number of iterations = 100		
RF	N_estimators = 100	Split criteria = ‘gini’	Feature considered for splitting a node = ‘sqrt’	Use bootstrap = yes
DT	Split criteria = ‘gini’	Max_depth = grow until trees are pure	Min_samples_split = 2	Min_samples_leaf = 1
KNN	Number of neighbors = 5	Weight of neighbors = ‘uniform’	Leaf size = 30	Parameter = ‘Euclidean distance’
SVM	Kernel type = ‘rbf’	Gamma = ‘scale’	Tolerance for stopping criteria = 10^{-3}	
GBM	Number of estimators = 100	Learning rate = 0.1	Estimator max depth = 3	Min samples required to split a node = 2

4.4 Experimental Results

In this section, we first determine whether the ML models underfit or overfit the occupancy dataset. To do this, training subsets of increasing sizes were iteratively selected, and each model was trained on them. For each training size, the models' performance was assessed on both the training and validation sets by computing their respective accuracy scores. We plotted the training and validation accuracy over multiple epochs for all 6 ML models as illustrated in Figures 2(a) through (f).

https://doi.org/10.24191/jsst.v5i1.101

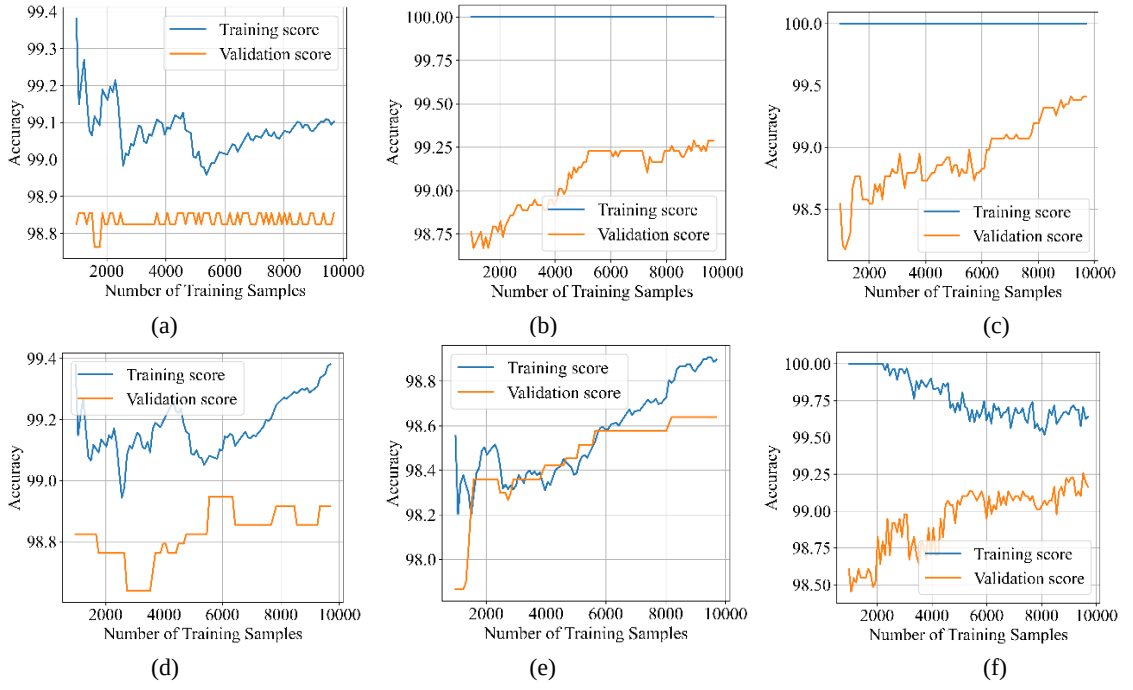
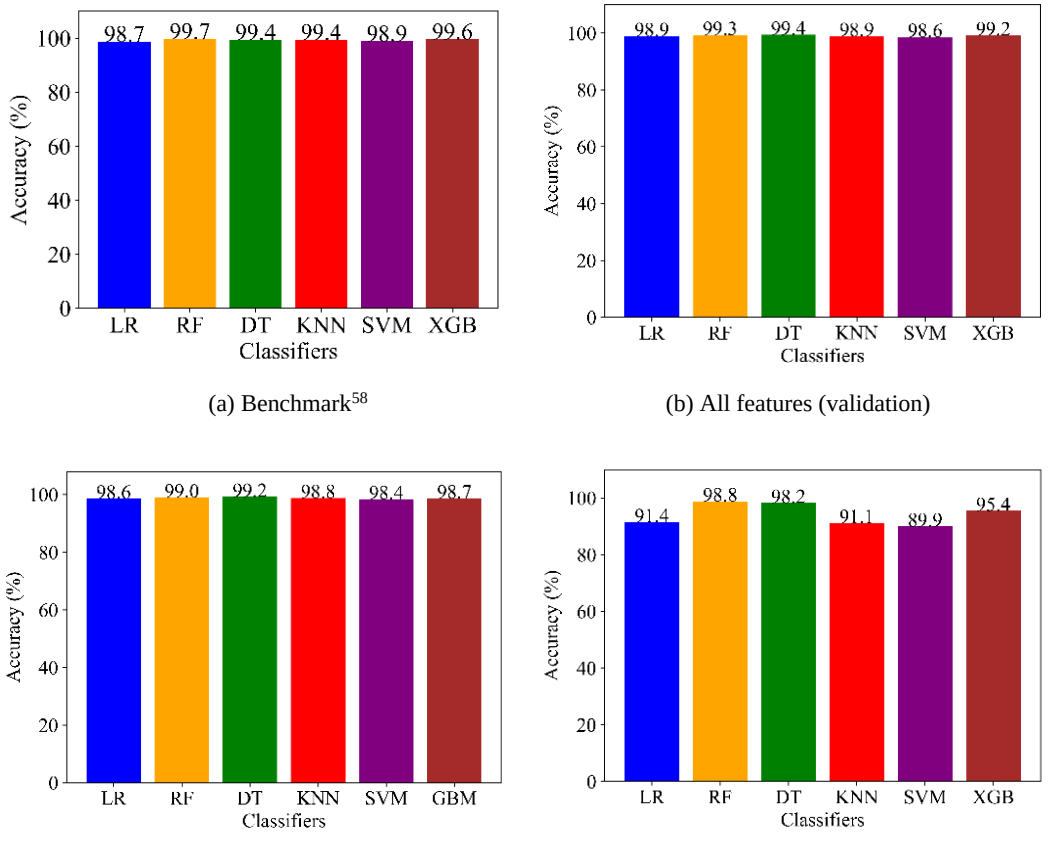


Fig. 2 Learning curves which show the training and validation scores as a function of the number of training samples used in ML models which include: (a) Logistic regression, (b) Random Forest, (c) Decision tree, (d) K-nearest neighbours, (e) Support vector machine, and (f) Gradient boosting machine.

In machine learning, if a model has low training and low validation accuracies (high bias), the model fails to capture the complexity of the data and therefore underfits it. Conversely, if a model has high training accuracy but low validation accuracy (high variance), the model memorizes the training data and overfits it, leading to poor generalisation. Figures 2(a) through (f) show that the training curves displayed high accuracy, indicating effective learning from the training data. The validation curves also reached a similarly high accuracy without showing a significant gap from the training curve. These alleviate concerns of underfitting and overfitting and affirm the high generalizability of our 6 trained ML models.

In order to observe the impact of partial feature availability problems on the performance of the ML models, we present 3 groups of results for each category of experiment. The first group, labelled as "All features (validation)", are instances where correlation-based feature selection described in Algorithm 2 was not performed on the validation experiment result. In this case, a high correlation may exist between a feature and the ground truth. The second group is similar to the first group except that the figures are labelled as "All features (test)", which denotes the results of the testing experiment. Hence, just like the first group, a high correlation between a feature and the ground truth may exist. Furthermore, the third group has the label "partial features" which represents instances where Algorithm 2 has been executed to filter off features that share a high correlation with the ground truth. Additionally, we present a set of results[58] found in the literature that are labelled as “benchmark”. They are included to provide insight regarding the performance values of a published work that used similar models and the same dataset as ours. It should be noted that these benchmark results, which were obtained by Alam et al.[58], have not considered potential partial feature availability problems and may likely perform poorly under this condition.

https://doi.org/10.24191/jsst.v5i1.101



(a) Benchmark[58]

(b) All features (validation)

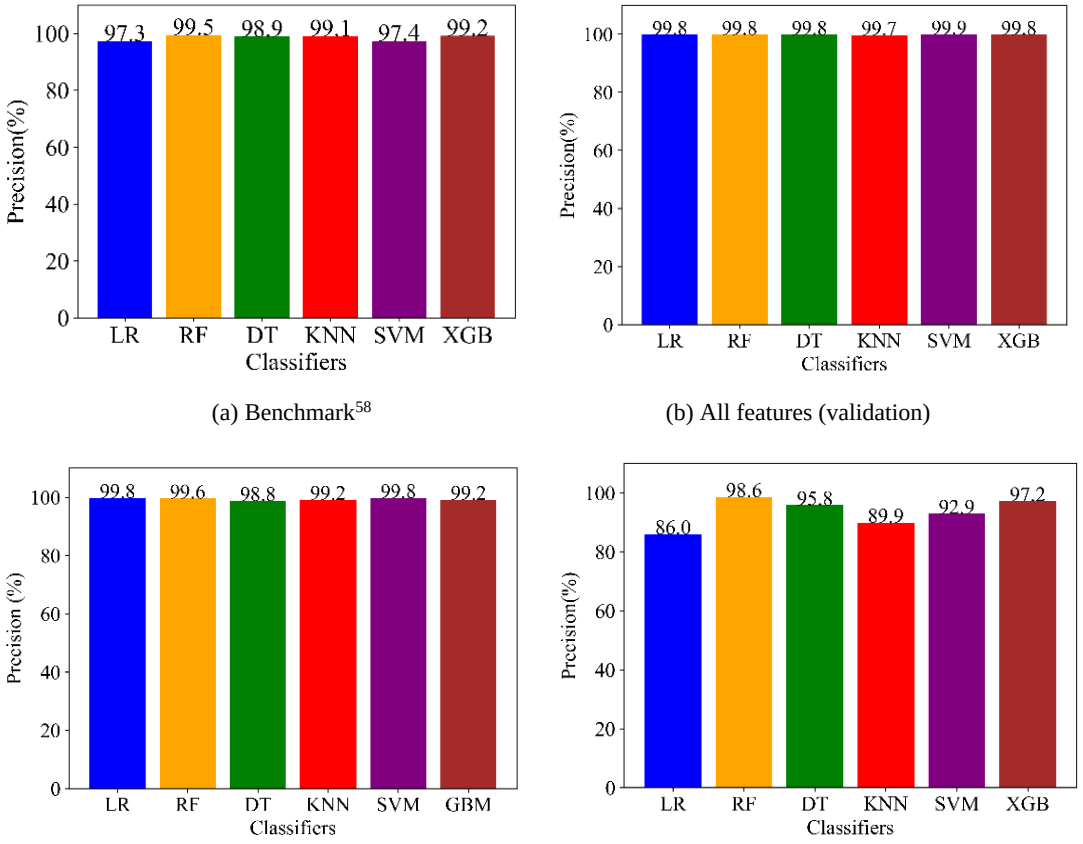
(c) All features (test)

(d) Partial features

Fig. 3 Accuracy of 6 classifiers: (a) result of benchmark algorithm which is susceptible to degradation by partial feature availability problem, (b) validation result of proposed algorithms which is susceptible to degradation by partial feature availability problem, (c) test result of proposed algorithm which is susceptible to degradation by partial feature availability problem (d) result of proposed algorithms with partial feature availability problem introduced.

Fig. 3(a), 4(a), 5(a), and 6(a) are the accuracy, precision, recall, and F1-scores of the benchmark model. It can be observed that all benchmark results exceed 97.0% performance score which is impressive. However, if subjected to the partial feature availability problem, these performance values will likely drop significantly. Fig. 3(b), 3(c), and 3(d) show the accuracies of the proposed ML models. Specifically, Fig. 3(b) shows the accuracy results of the validation experiment, Fig. 3(c) shows the testing experiment accuracy results, and Fig. 3(d) shows the results where the partial feature availability problem has been considered. It can be observed in Fig. 3(b) and (c) that all the ML models yielded high accuracy performances when the partial feature availability issue was ignored. However, in Fig. 3(d), most of the accuracy values dropped when the partial feature availability issue was considered. Specifically, the SVM model in Fig. 3(d) had the highest drop in accuracy value, while the RF model had the lowest.

https://doi.org/10.24191/jsst.v5i1.101



(a) Benchmark[58] (b) All features (validation)

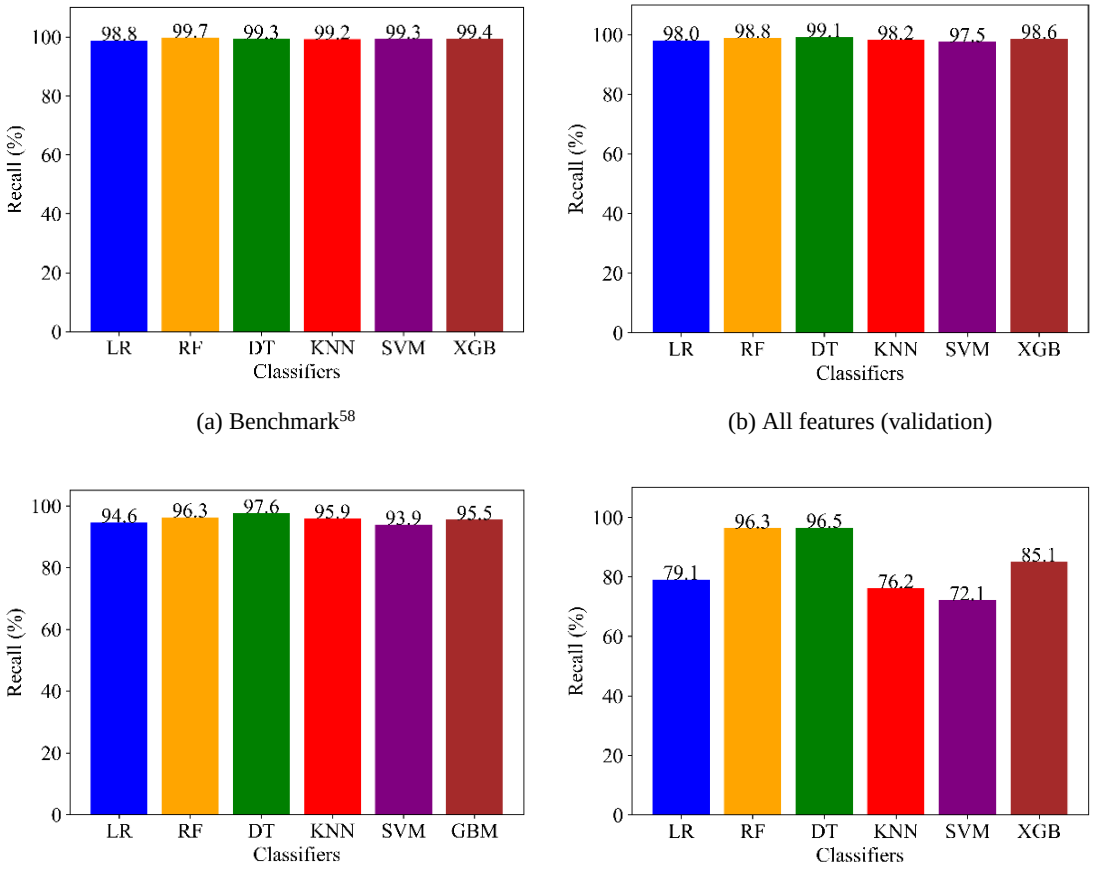
(c) All features (test) (d) Partial features

Fig. 4. Precision scores of 6 classifiers: (a) result of benchmark algorithm which is susceptible to degradation by partial feature availability problem, (b) validation result of proposed algorithms which is susceptible to degradation by partial feature availability problem, (c) test result of proposed algorithm which is susceptible to degradation by partial feature availability problem (d) result of proposed algorithms with partial feature availability problem introduced.

Fig. 4(b) and (c) show the precision scores of the ML models when the partial feature availability issue was ignored while Fig. 4(d) shows the precision scores of the ML models when the partial feature availability issue was considered. It is evident in Fig. 4(b) and (c) that all models have high precision values exceeding 98%. However, most of the precision values dropped when the partial feature availability issue is considered in Fig. 4(d). In this figure, the RF model exhibited the least drop in precision value, while the LR, KNN, and the SVM models experienced more significant drops.

Illustrated in Fig. 5(b) and (c) are the recall scores of the 6 ML models when the partial feature availability problem was ignored while Fig. 5(d) shows the recall scores of the ML models when the partial feature availability issue was considered. It can be observed in Fig. 5(b) and (c) that all models had high recall scores when the partial feature availability issue was ignored, but when it was considered in Fig. 5(d), the performances of some of the models depreciated significantly. For example, the recall scores of SVM and KNN dropped by about 20% each when compared to the results of the test experiment. The RF and DT models, on the other hand, experienced only a small decrease in their recall scores.

https://doi.org/10.24191/jsst.v5i1.101



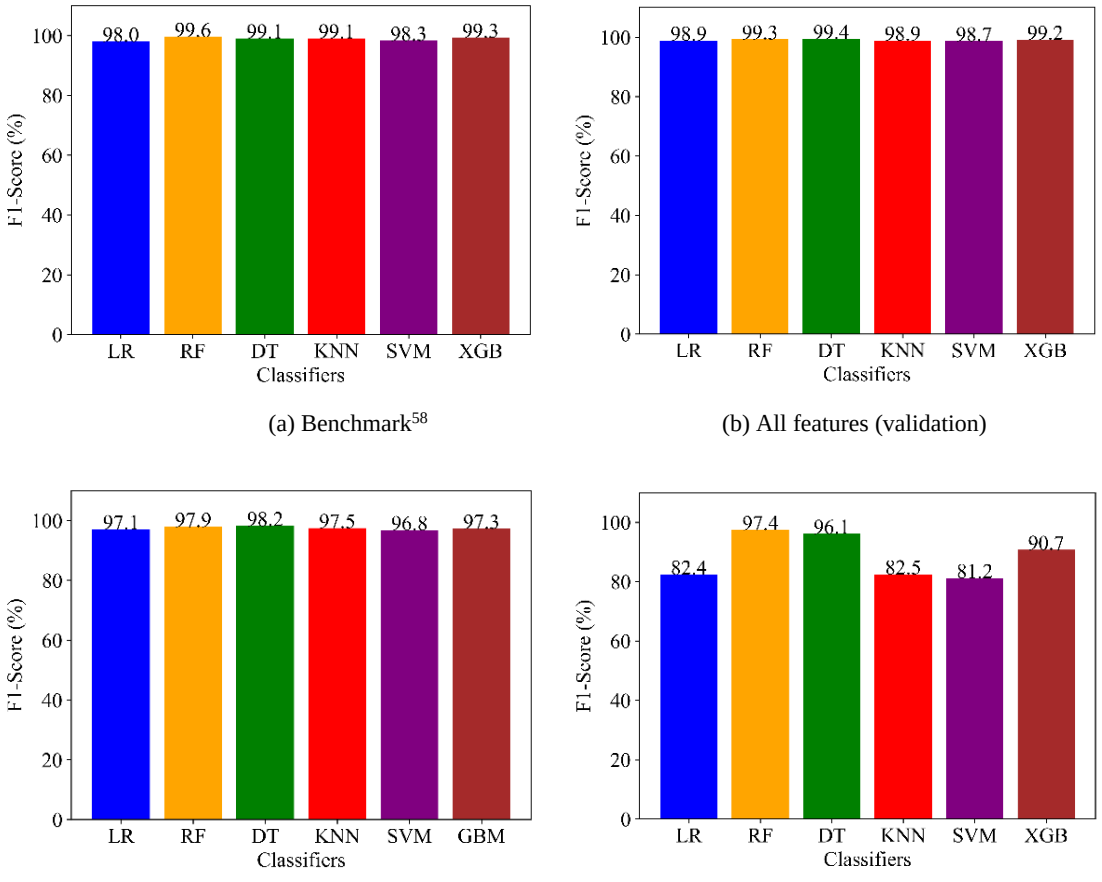
(a) Benchmark[58] (b) All features (validation)

(c) All features (test) (d) Partial features

Fig. 5. Recall scores of 6 classifiers: (a) result of benchmark algorithm which is susceptible to degradation by partial feature availability problem, (b) validation result of proposed algorithms which is susceptible to degradation by partial feature availability problem, (c) test result of proposed algorithm which is susceptible to degradation by partial feature availability problem (d) result of proposed algorithms with partial feature availability problem introduced.

Fig. 6(b) and (c) show the F1-Scores of the 6 ML models when the partial feature availability issue was ignored while Fig. 6(d) shows the F1-Scores of the ML models when the partial feature availability issue was considered. It can be seen in Fig. 6(b) and (c) that all models have high F1-scores when the partial feature availability issues were ignored. However, with the partial feature availability problem considered (in Fig. 6(d)), some of them decreased in value. For example, compared to the test experiment in Fig. 6(c), the F1-score of KNN decreased from 97.5% to 82.5%, while that of LR decreased from 97.1% to 82.4%. Again, similar to the previous results, the F1-score of the RF and DT models were least affected by the partial feature availability problem.

https://doi.org/10.24191/jsst.v5i1.101



(a) Benchmark[58] (b) All features (validation)

(c) All features (test) (d) Partial features

Fig. 6. F1-scores of 6 classifiers: (a) result of benchmark algorithm which is susceptible to degradation by partial feature availability problem, (b) validation result of proposed algorithms which is susceptible to degradation by partial feature availability problem, (c) test result of proposed algorithm which is susceptible to degradation by partial feature availability problem (d) result of proposed algorithms with partial feature availability problem introduced.

4.5 Discussion

In the previous subsection, simulation results showed that when the partial feature availability problem was ignored, the performances of most of the ML models were satisfactory but plummeted when it was considered. However, not all models were seriously adversely impacted by the problem. To determine the robustness of the machine learning models against the partial feature availability problem, Algorithm 3 was applied. Recall that Algorithm 3 has two parts. The first part generated binary values for vectors $[Mod_w^{\mathrm{Acc}}]_{w=1}^{W}$, $[Mod_w^{\mathrm{Pre}}]_{w=1}^{W}$, $[Mod_w^{\mathrm{Rec}}]_{w=1}^{W}$, and $[Mod_w^{\mathrm{F1}}]_{w=1}^{W}$ while the second part used a bitwise-AND operator to determine the partial-feature-robustness vector $\left[Mod_w^{\mathrm{T\_rob}}\right]_{w=1}^{W}$. When Algorithm 3 was applied, the values of the binary vectors generated by the first part of the algorithm were: $[Mod_w^{\mathrm{Acc}}]_{w=1}^{W}$= 111101, $[Mod_w^{\mathrm{Pre}}]_{w=1}^{W}$ = 011011, $[Mod_w^{\mathrm{Rec}}]_{w=1}^{W}$= 011000, and $[Mod_w^{\mathrm{F1}}]_{w=1}^{W}$ = 011001.

To explain these vectors, let us use $[Mod_w^{\mathrm{Acc}}]_{w=1}^{W}$= 111101 as an example. Observe that in the vector $[Mod_w^{\mathrm{Acc}}]_{w=1}^{W}$, all the binary digits are "1" with the exception of the 5$^{\mathrm{th}}$ digit which is a "0". If Fig. 3(d) is observed, it can be seen that the values of accuracies of the 6 ML models considered are LR = 91.44%, RF = 98.77%, DT = 98.2%, KNN = 91.06%, SVM = 89.92%, and GBM = 95.36%. In these accuracy

https://doi.org/10.24191/jsst.v5i1.101

results, it can be seen that the accuracies of all models with the exception of the 5th model (SVM) exceeded P_{th} (where P_{th} = 90%). Based on the binarisation procedure of Algorithm 3, it implies that at $w = 5$, $Mod_w^{Acc}$is set to a value of "0" but set to a value of "1" in all other instances of w. This explains why $[Mod_w^{Acc}]_{w=1}^{W}$ has a value of 111101. The values of the other vectors $[Mod_w^{Pre}]_{w=1}^{W}$, $[Mod_w^{Rec}]_{w=1}^{W}$, and $[Mod_w^{F1}]_{w=1}^{W}$ are computed in a similar way. The final output of Algorithm 3, $\left[Mod_w^{T\_rob}\right]_{w=1}^{W}$, is determined based on the bitwise-AND operation shown in Table 3:

Table 3. Illustration of the bitwise AND operation of Algorithm 3

	w = 1	w = 2	w = 3	w = 4	w = 5	w = 6
	LR	RF	DT	KNN	SVM	GBM
Mod_w^{Acc}	1	1	1	1	0	1
Mod_w^{Pre}	0	1	1	0	1	1
Mod_w^{Rec}	0	1	1	0	0	0
Mod_w^{F1}	0	1	1	0	0	1
Bitwise AND Operation $\left(Mod_w^{T\_rob}\right)$	0	1	1	0	0	0

From the result shown in Table 3, it can be seen that the value of the partial features robustness vector is $\left[Mod_w^{T\_rob}\right]_{w=1}^{W} = 011000$. This implies that the RF and DT are robust against partial feature availability problems for the scenario considered. This is because they occupy the 2nd and 3rd-bit positions of $\left[Mod_w^{T\_rob}\right]_{w=1}^{W}$ vector which are the only bit positions having values of "1" in the vector. This is confirmed by Fig. 3 through 6 where it was observed that the RF and DT machine learning models outperformed all the other ML models considered based on the chosen performance metrics. Therefore, for the combination of sensors considered in this work, deploying indoor occupancy detection systems that are based on RF and DT models stands a high chance of success in real environments.

5 CONCLUSION

In this paper, data from environmental sensors was applied to machine learning-based classification models to predict the occupancy status of an enclosed space. Three algorithms were developed, namely outlier removal, feature selection, and partial-feature-aware ML model selection algorithms. The machine learning models considered in this study included Logistic Regression (LR), Random Forest (RF), Decision Tree (DT), K-Nearest Neighbours (KNN), Support Vector Machines (SVM), and Gradient Boosting Machines (GBM) models. These models were applied to publicly available data from environmental sensors such as temperature, humidity, carbon dioxide (CO_2), and light sensors. It was observed that most of the ML models initially yielded high-performance results but were vulnerable to issues related to partial feature availability. However, further simulation experiments, which accounted for the partial feature availability issue considered, showed that only the RF and DT models were less affected by the partial feature availability problem and were therefore recommended for deployment in real environments using the combination of sensors considered in this study. Further experimental studies will be conducted as part of our future work.

ACKNOWLEDGEMENTS/ FUNDING

The authors declare that they have no known competing financial interests or funding to declare.

https://doi.org/10.24191/jsst.v5i1.101

CONFLICT OF INTEREST

The authors agree that this research was conducted in the absence of any self-benefits, commercial or financial conflicts and declare the absence of conflicting interests with the funders.

AUTHORS' CONTRIBUTIONS

Conceptualization: A. S. Afolabi
Data curation: A. S. Afolabi & O. A. Odetoye
Methodology: A. S. Afolabi & O. A. Akinola
Formal analysis: A. S. Afolabi & E. Adetiba
Visualisation: O. A. Akinola
Software: A. S. Afolabi
Writing (original draft): A. S. Afolabi, O. A. Akinola, O. A. Odetoye, & E. Adetiba
Writing (review and editing): A. S. Afolabi, O. A. Akinola, O. A. Odetoye, & E. Adetiba
Validation: E. Adetiba
Supervision: O. A. Akinola & E. Adetiba
Funding acquisition: Not applicable
Project administration: Not applicable

REFERENCES

1. Mosaico, G., Saviozzi, M., Silvestro, F., Bagnasco, A., & Vinci, A. (2019). Simplified state space building energy model and transfer learning based occupancy estimation for HVAC optimal control. In *Innovation to shape the future.* 5th International Forum on Research and Technology for Society and Industry (RTSI), (pp. 353–358). Florence, Italy. https://doi.org/10.1109/RTSI.2019.8895544

2. Acquaah, Y., Steele, J. B., Gokaraju, B., Tesiero, R., & Monty, G. H. (2020). Occupancy detection for smart HVAC efficiency in building energy: A deep learning neural network framework using thermal imagery. In Applied Imagery Pattern Recognition Workshop (AIPR) (pp. 1–6). Washington DC, USA. https://doi.org/10.1109/AIPR50011.2020.9425091

3. Acquaah, Y. T., Gokaraju, B., Tesiero, R. C., & Monty, G. H. (2021). Thermal imagery feature extraction techniques and the effects on machine learning models for smart HVAC efficiency in building energy. *Remote Sensing*, *13*(19), 3847. https://doi.org/10.3390/rs13193847

4. Do, H. M., Pham, M., Sheng, W., Yang, D., & Liu, M. (2018). RiSH: A robot-integrated smart home for elderly care. *Robotics and Autonomous Systems*, *101*, 74–92. https://doi.org/10.1016/j.robot.2017.12.008

5. Tan, B., Chen, Q., Chetty, K., Woodbridge, K., Li, W., & Piechocki, R. (2018). Exploiting WiFi channel state information for residential healthcare informatics. *IEEE Communications Magazine*, *56*(5), 130-137. https://doi.org/10.1109/MCOM.2018.1700064

6. Harrou, F., Zerrouki, N., Sun, Y., & Houacine, A. (2019). An integrated vision-based approach for efficient human fall detection in a home environment. *IEEE Access*, *7*, 114966–114974. https://doi.org/10.1109/ACCESS.2019.2936320

7. Bocus, M. J., & Piechocki, R. (2022). A comprehensive ultra-wideband dataset for non-cooperative contextual sensing. *Scientific Data*, *9*(1), 650. https://doi.org/10.1038/s41597-022-01776-7

8. Shah, S. A., Ahmad, J., Tahir, A., Ahmed, F., Russell, G., Shah, S. Y., Buchanan, W. J., & Abbasi, Q. H. (2020). Privacy-preserving non-wearable occupancy monitoring system exploiting Wi-Fi imaging for next-generation body centric communication. *Micromachines*, *11*(4), 379. https://doi.org/10.3390/mi11040379

https://doi.org/10.24191/jsst.v5i1.101

9. Singh, R., Pai, N., & Mahapatra, S. (2022). Enabling smart building applications on the edge using artificial intelligence. In *Smarter technologies for a sustainable and hyper-connected world*, Women in Technology Conference (WINTECHCON) (pp. 1–6). Bangalore, India. https://doi.org/10.1109/WINTECHCON55229.2022.9832044

10. Demrozi, F., Turetta, C., Chiarani, F., Kindt, P. H., & Pravadelli, G. (2021). Estimating indoor occupancy through low-cost BLE devices. *IEEE Sensors Journal*, *21*(15), 17053–17063. https://doi.org/10.1109/JSEN.2021.3080632

11. Leeraksakiat, P., & Pora, W. (2020). Occupancy forecasting using LSTM neural network and transfer learning. In *17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, (pp. 470–473). Phuket, Thailand. https://doi.org/10.1109/ECTI-CON49241.2020.9158103

12. Zou, H., Zhou, Y., Yang, J., & Spanos, C. J. (2018). Device-free occupancy detection and crowd counting in smart buildings with WiFi-enabled IoT. *Energy and Buildings*, *174*, 309–322. https://doi.org/10.1016/j.enbuild.2018.06.040

13. Yang, J., Pantazaras, A., Chaturvedi, K. A., Chandran, A. K., Santamouris, M., Lee, S. E., & Tham, K. W. (2018). Comparison of different occupancy counting methods for single system-single zone applications. *Energy and Buildings*, *172*, 221–234. https://doi.org/10.1016/j.enbuild.2018.04.051

14. Chen, Z., Jiang, C., & Xie, L. (2018). Building occupancy estimation and detection: A review. *Energy and Buildings*, *169*, 260–270. https://doi.org/10.1016/j.enbuild.2018.03.084

15. Zhang, T., & Ardakanian, O. (2019). A domain adaptation technique for fine-grained occupancy estimation in commercial buildings. In *Advancing Computing as a Science & Profession.* International Conference on Internet of Things Design and Implementation *(IoTDI)* (pp. 148–159). Montreal Quebac, Canada. https://doi.org/10.1145/3302505.3310077

16. Elkhoukhi, H., Bakhouya, M., Hanifi, M., & El Ouadghiri, D. (2019). On the use of deep learning approaches for occupancy prediction in energy efficient buildings. In *7th International Renewable and Sustainable Energy Conference (IRSEC)* (pp. 1–6). Agadir, Morocco. https://doi.org/10.1109/IRSEC48032.2019.9078164

17. Szagri, D., Dobszay, B., Nagy, B., & Szalay, Z. (2022). Wireless temperature, relative humidity and occupancy monitoring system for investigating overheating in buildings. *Sensors*, *22*(22), 8638. https://doi.org/10.3390/s22228638

18. O'Grady, T., Chong, H.-Y., & Morrison, G. M. (2021). A systematic review and meta-analysis of building automation systems. *Building and Environment*, *195*, 107770. https://doi.org/10.1016/j.buildenv.2021.107770

19. Wagner, D. N., Mathur, A., & Boor, B. E. (2021). Spatial seated occupancy detection in offices with a chair-based temperature sensor array. *Building and Environment*, *187*, 107360. https://doi.org/10.1016/j.buildenv.2020.107360

20. Yuan, L., Andrews, J., Mu, H., Vakil, A., Ewing, R., Blasch, E., & Li, J. (2022). Interpretable passive multi-modal sensor fusion for human identification and activity recognition. *Sensors*, *22*(15), 5787. https://doi.org/10.3390/s22155787

21. Hu, S., Wang, P., Hoare, C., & O'Donnell, J. (2023). Building occupancy detection and localization using CCTV camera and deep learning. *IEEE Internet of Things Journal*, *10*(1), 597–608. https://doi.org/10.1109/JIOT.2022.3201877

22. Ding, Y., Han, S., Tian, Z., Yao, J., Chen, W., & Zhang, Q. (2022). Review on occupancy detection and prediction in building simulation. *Building Simulation*, *15*, 333–356. https://doi.org/10.1007/s12273-021-0813-8

23. Zhang, R., Kong, M., Dong, B., O'Neill, Z., Cheng, H., Hu, F., & Zhang, J. (2022). Development of a testing and evaluation protocol for occupancy sensing technologies in building HVAC controls: A case study of representative people counting sensors. *Building and Environment*, *208*, 108610. https://doi.org/10.1016/j.buildenv.2021.108610

24. Matuska, S., Machaj, J., Hudec, R., & Kamencay, P. (2022). An improved IoT-based system for detecting the number of people and their distribution in a classroom. *Sensors*, *22*(20), 7912. https://doi.org/10.3390/s22207912

25. Song, C., Droitcour, A. D., Islam, S. M. M., Whitworth, A., Lubecke, V. M., & Boric-Lubecke, O. (2023). Unobtrusive occupancy and vital signs sensing for human building interactive systems. *Scientific Reports*, *13*, 954. https://doi.org/10.1038/s41598-023-27425-6

26. Liu, Y., Wang, T., Jiang, Y., & Chen, B. (2022). Harvesting ambient RF for presence detection through deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(4), 1571–1583. https://doi.org/10.1109/TNNLS.2020.3042908

27. Azam, M., Blayo, M., Venne, J.-S., & Allegue-Martinez, M. (2019). Occupancy estimation using wifi motion detection via supervised machine learning algorithms. In *Global Conference on Signal and Information Processing (GlobalSIP)* (pp. 1–5). Ottawa, Canada. https://doi.org/10.1109/GlobalSIP45357.2019.8969297

28. Ahmad, J., Larijani, H., Emmanuel, R., Mannion, M., & Javed, A. (2021). Occupancy detection in non-residential buildings – A survey and novel privacy preserved occupancy monitoring solution. *Applied Computing and Informatics*, *17*(2), 279–295. https://doi.org/10.1016/j.aci.2018.12.001

29. Amayri, M., Ploix, S., Bouguila, N., & Wurtz, F. (2019). Estimating occupancy using interactive learning with a sensor environment: Real-time experiments. *IEEE Access*, *7*, 53932–53944. https://doi.org/10.1109/ACCESS.2019.2911921

30. Emad-ud-Din, M., Chen, Z., Wu, L., Shen, Q., & Wang, Y. (2022). Indoor occupancy estimation using particle filter and SLEEPIR sensor system. *IEEE Sensors Journal*, *22*(17), 17173–17183. https://doi.org/10.1109/JSEN.2022.3192270

31. Pratama, A. R., Lazovik, A., & Aiello, M. (2019). Office multi-occupancy detection using BLE beacons and power meters. In *10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)* (pp. 0440–0448). New York, USA. https://doi.org/10.1109/UEMCON47517.2019.8993008

32. Oshima, H., Ishizone, T., Nakamura, K., & Higuchi, T. (2022). Occupancy detection for general households by bidirectional LSTM with attention. In *IECON 48th Annual Conference of the IEEE Industrial Electronics Society* (pp. 1–7). Brussels, Belgium. https://doi.org/10.1109/IECON49645.2022.9968594

33. Kwon, S.-Y., & Lee, S. (2022). In-vehicle seat occupancy detection using ultra-wideband radar sensors. In *23rd International Radar Symposium (IRS)* (pp. 275–278). Gdansk, Poland. https://doi.org/10.23919/IRS54158.2022.9905064

34. Odetoye, O. A., Afolabi, A. S., & Akinola, O. A. (2020). Development and scaled-up simulation of an automated electrical energy management system for passageway illumination. *International Journal of Emerging Electric Power Systems*, *21*(6), 20200124. https://doi.org/10.1515/ijeeps-2020-0124

https://doi.org/10.24191/jsst.v5i1.101

35. Andrews, J., Kowsika, M., Vakil, A., & Li, J. (2020). A Motion Induced Passive Infrared (PIR) sensor for stationary human occupancy detection. In *IEEE/ION Position, Location and Navigation Symposium (PLANS)* (pp. 1295–1304). Portland, USA. https://doi.org/10.1109/PLANS46316.2020.9109909

36. Ahmad Mahmud, N. F., & Azuana Ramli, N. (2020). Hybrid classification method to detect the presence of human in a smart building environment. In *International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI)* (pp. 1–5). Sakheer, Bahrain. https://doi.org/10.1109/ICDABI51230.2020.9325671

37. Aromolaran, O., Beder, T., Adedeji, E., Ajamma, Y., Oyelade, J., Adebiyi, E., & Koenig, R. (2021). Predicting host dependency factors of pathogens in Drosophila melanogaster using machine learning. *Computational and Structural Biotechnology Journal*, *19*, 4581–4592. https://doi.org/10.1016/j.csbj.2021.08.010

38. Adetiba, E., Ajayi, O. T., Kala, J. R., Badejo, J. A., Ajala, S., Abayomi, A., Badejo, J. A., Adetiba, E., & Adetiba, E. (2021). LeafsnapNet: An experimentally evolved deep learning model for recognition of plant species based on leafsnap image dataset. *Journal of Computer Science*, *17*(3), 349–363. https://doi.org/10.3844/jcssp.2021.349.363

39. Domingos, E., Ojeme, B., & Daramola, O. (2021). Experimental analysis of hyperparameters for deep learning-based churn prediction in the banking sector. *Computation*, *9*(3), 34. https://doi.org/10.3390/computation9030034

40. Feng, C., Mehmani, A., & Zhang, J. (2020). Deep learning-based real-time building occupancy detection using AMI data. *IEEE Transactions on Smart Grid*, *11*(5), 4490–4501. https://doi.org/10.1109/TSG.2020.2982351

41. Parise, A., Manso-Callejo, M. A., Cao, H., Mendonca, M., Kohli, H., & Wachowicz, M. (2019). Indoor occupancy prediction using an IoT platform. In *Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)* (pp. 26–31). Granada, Spain. https://doi.org/10.1109/IOTSMS48152.2019.8939234

42. Khalil, M., McGough, S., Pourmirza, Z., Pazhoohesh, M., & Walker, S. (2021). Transfer learning approach for occupancy prediction in smart buildings. In *12th International Renewable Engineering Conference (IREC)* (pp. 1–6). Amman, Jordan. https://doi.org/10.1109/IREC51415.2021.9427869

43. Fayed, N. S., Elmogy, M. M., Atwan, A., & El-Daydamony, E. (2022). Efficient occupancy detection system based on neutrosophic weighted sensors data fusion. *IEEE Access*, *10*, 13400–13427. https://doi.org/10.1109/ACCESS.2022.3146346

44. Mutis, I., Ambekar, A., & Joshi, V. (2020). Real-time space occupancy sensing and human motion analysis using deep learning for indoor air quality control. *Automation in Construction*, *116*, 103237. https://doi.org/10.1016/j.autcon.2020.103237

45. Kim, J., Min, K., Jung, M., & Chi, S. (2020). Occupant behavior monitoring and emergency event detection in single-person households using deep learning-based sound recognition. *Building and Environment*, *181*, 107092. https://doi.org/10.1016/j.buildenv.2020.107092

46. Ng, P. C., & She, J. (2019). Denoising-contractive autoencoder for robust device-free occupancy detection. *IEEE Internet of Things Journal*, *6*(6), 9572–9582. https://doi.org/10.1109/JIOT.2019.2929822

47. Wang, Z., Hong, T., Piette, M. A., & Pritoni, M. (2019). Inferring occupant counts from Wi-Fi data in buildings through machine learning. *Building and Environment*, *158*, 281–294. https://doi.org/10.1016/j.buildenv.2019.05.015

48. Kapoor, S., & Narayanan, A. (2023). Leakage and the reproducibility crisis in machine-learning-based science. *Patterns*, *4*(9), 100804. https://doi.org/10.1016/j.patter.2023.100804

49. Rosenblatt, M., Tejavibulya, L., Jiang, R., Noble, S., & Scheinost, D. (2024). Data leakage inflates prediction performance in connectome-based machine learning models. *Nature Communications*, *15*(1), 1829. https://doi.org/10.1038/s41467-024-46150-w

50. Suresh, L. P., & Kalidindi, N. R. (2022). Study of test for significance of Pearson's correlation coefficient. *International Journal of Science and Research (IJSR)*, *11*(10), 164–166. https://doi.org/10.21275/SR22915140002

51. Ye, W., Zheng, G., Cao, X., Ma, Y., & Zhang, A. (2024). Spurious correlations in machine learning: A survey (Version 2). *arXiv*. https://doi.org/10.48550/arXiv.2402.12715

52. Afolabi, A. S., & Akinola, O. A. (2024). Network intrusion detection using knapsack optimization, mutual information gain, and machine learning. *Journal of Electrical and Computer Engineering*, *20244(1)*, 7302909 https://doi.org/10.1155/2024/7302909

53. Candanedo, L. (2016). Occupancy Detection [Dataset]. *UCI Machine Learning Repository*. https://doi.org/10.24432/C5X01N

54. Candanedo, L. M., & Feldheim, V. (2016). Accurate occupancy detection of an office room from light, temperature, humidity and CO_2 measurements using statistical learning models. *Energy and Buildings*, *112*, 28–39. https://doi.org/10.1016/j.enbuild.2015.11.071

55. Shanthamallu, U. S., Spanias, A., Tepedelenlioglu, C., & Stanley, M. (2017). A brief survey of machine learning methods and their sensor and IoT applications. In *8th International Conference on Information, Intelligence, Systems & Applications (IISA)* (pp. 1–8). Larnaca, Cyprus. https://doi.org/10.1109/IISA.2017.8316459

56. Shahriar, S., Al-Ali, A. R., Osman, A. H., Dhou, S., & Nijim, M. (2020). Machine learning approaches for EV charging behavior: A review. *IEEE Access*, *8*, 168980–168993. https://doi.org/10.1109/ACCESS.2020.3023388

57. Mansi, M., Almobarak, M., Ekundayo, J., Lagat, C., & Xie, Q. (2024). Application of supervised machine learning to predict the enhanced gas recovery by CO_2 injection in shale gas reservoirs. *Petroleum*, *10*(1), 124–134. https://doi.org/10.1016/j.petlm.2023.02.003

58. Alam, S., Sari, R. M., Alfian, G., & Farooq, U. (2024). Room occupancy detection based on random forest with timestamp features and ANOVA feature selection method. *Journal of Computing Science and Engineering*, *18*(1), 10–18. https://doi.org/10.5626/JCSE.2024.18.1.10