


Sensor-based room inhabitation monitoring using robust ML models compatible with large datasets / real-time datastreams

Alexandru Pintea

(University of Sheffield, Sheffield, UK)

 <https://orcid.org/0009-0003-2158-9328>, alexandru.pintea@icee.org

Abstract: Smart homes, live streaming IoT devices, and smart sensors can all be optimised to enhance energy efficiency. In order to offer a cheap alternative to the traditional real-time monitoring systems, this study proposes a sensor-based occupancy system. The evaluation in real time of the number of occupants in buildings/ rooms /houses is reflected in the energy usage. Sensor data can provide insight into many characteristics of a considered environment. The sensor dataset considered was collected with the aim of determining how many people are present in a given space/room. The sensor data does not portray the people present in the room, but rather their impact on it (e.g. CO₂/ noise/ light level changes). The dataset was cleaned and preprocessed to optimise model performance. The results obtained by training several classifiers yielded accuracies that reach 98%-99%. The research provides an end-to-end solution for the considered problem, through data preprocessing/feature selection/outlier removal and model training/evaluation. Hyperparameters were tuned for more than twenty models. All chosen models and features were ranked based on performance and robustness. A novel solution for optimising sensor placement has also been proposed by this study, to further improve sensor-based monitoring systems.

Keywords: Machine Learning, classification, big data, live streams data, sensors

Categories: I.2.0, I.2.6, I.5.0, I.5.1

DOI: 10.3897/jucs.150393

1 Introduction

Nowadays, all automated smart spaces (e.g. houses, institutional buildings and vehicles) use high-end technology to improve user experience [Huda et al., 2024, Vasisht et al., 2018]). Yet, such improvements come with various impediments, one of which is increased energy costs. Regarding this, it is well known that energy consumption is directly proportional with the number of inhabitants present in the considered area [Cottafava et al., 2019, Han and Zhang, 2020]. One study that addresses the aforementioned topic is [Zhang et al., 2024], where the number of students in a classroom was estimated using environmental sensor data. This study found that a Deep Learning (DL) model is best suited for its needs.

[Chitnis et al., 2025] compares such studies, to further elaborate on which models should be used for each of the identified use cases. [Venzke et al., 2020] surveys multiple on-chip implementations of ANN models to rank their performance, while considering their robustness (memory and CPU/GPU needs). This study deemed that a 2 layer feed-forward NN is superior to all other considered NNs, given its performance and computational needs. Its 1493 parameters could provide real-time classification results in less than 36ms, which is comparable to less performant, non-NN alternatives.

[Begum et al., 2024] mainly focuses on identifying cyberattacks targeted at self-driving car sensors, in real time. It's an implementation of "AI anomaly detection", a field adjacent to data classification. Their AI model had to be adjusted from the simulation environment to the real-life environment, due to sensor ware. This reveals that AI models need to deal with sensor ware dynamically, to ensure prediction accuracy. Their SADC model surpassed others by 0.98% in outlier detection. Ultrasonic sensors were found superior to light sensors, as the anomalies introduced by attackers are less human interpretable and therefore easier to detect.

The work of [Khan et al., 2023] uses smartphone-embedded sensors that measure human activity, helping determine the activity performed by the user. Their proposed RF model surpassed the other models considered (K-Nearest Neighbor (KNN)/Decision Trees (DT)/ Support Vector Machine (SVM)/ Multinomial Logistic Regression). A variation of Long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] achieved RF-comparable results. The MEMS sensors considered were gyroscopes/magnetometers/pressure sensors/sound sensors. Classification over 9 labels was achieved with an RF accuracy of 95%.

[Mohammed et al., 2023] proved that convolutions enable accurate prediction on sensor-related data. [Khan et al., 2023] compared multiple models for their dataset, establishing RF superiority. DTs/SVMs/SADC/CNNs were shown to be equally accurate on the considered datasets, while Multinomial Logistic Regression was proven to be unfit for the considered problem (its accuracy was only 67%)(Table 1). This reveals that DTs/SVMs offer a robust alternative to RFs/CNNs, without compromising heavily on performance. They are optional for "big data" datasets.

The aim of this study is determining the number of people present in a space/room based solely on sensor data. Different sensors (entities that measure environmental properties) were ranked on their contribution. There is a limited number of dataset publicly available. The study use [Singh et al., 2018] dataset. Similar datasets were collected via sensors in a public building in Northern Europe during four days (March 2018-April 2019) [Schwee et al., 2019].

Introduced by [Pedregosa et al., 2011] Scikit-learn (SkLearn) was chosen for its models' robustness, which is detrimental when dealing with "big data" live streams. The dataset was cleaned, preprocessed, balanced, reduced though feature selection and then used for training multiple models. Five models were proposed after 20+ models were trained and evaluated (using supervised learning methods). Information collected from the literature review [Li et al., 2024] was used to choose classifiers and adjust their hyperparameters. Crossvalidation enabled all SkLearn models to generalise well, especially after numerous hyperparameter configurations were tested for all of them.

The paper continues with the methodology section including the process workflow, dataset preprocessing phase and the model training. Follows the section 3 with the classification models description and hyperparameter settings. Section 4 includes the evaluation and analysis of the employed models and of several aggregative models. Detailed discussions are presented in section 5. The paper concludes with the major results and future works.

2 Methodology

Generally speaking, the project lifecycle includes preprocessing the dataset, training/testing/evaluating models and providing theoretical insight on model deployment and maintenance. Parts of it shall now be discussed in further detail.

Model	Settings	Accuracy
[Khan et al., 2023] model		
Multinomial Logistic Regression	dual=False, tol= 10^{-4} , C=1, fit_intercept=True	67%
Gaussian Naive Bayes	var_smoothing= 10^{-9}	89%
Decision Tree	min_samples_split=2, min_samples_leaf=1	93% **
Random Forest	n_estimators=100, criterion="gini", random_state=43	95%*
K Neighbors	algorithm="auto", n_neighbors=10, weights="uniform"	91%
SVM	C=1, Kernel="rbf", Degree=3, gamma="scale"	93% **
LSTM Variant	own, unique structure	98.1% *
[Begum et al., 2024] model		
SADC	own, unique structure	94.7% **
[Mohammed et al., 2023] model		
CNN	Conv2D, BatchNorm2D, MaxPool, Conv2D, MaxPool, Conv2D, BatchNorm2D, MaxPool, Dropout, Sense, Dropout, Sense, Dropout, Sense	93.8% fault prediction *, 95.3 % classification *
* represents the best performing models		
** represents performative models with a low complexity (good, robust models)		

Table 1: Accuracy/Structure of literature review models.

2.1 Dataset

The chosen dataset Room Occupancy Estimation (collected by Adarsh Pal Singh and Sachin Chaudhari [Singh et al., 2018]) consists mostly of sensor information (Table 2). The initial csv dataset contains 10129 rows and 19 columns (18 features). Four temperature/ sound/ light sensors were used (12 in total) as well as one CO₂ sensor and two PIR (passive infrared) sensors. All sensors were hung from the ceiling of the considered room (6 × 4.6m), arranged to cover (sense) the whole area of the room (considering "FOV" sensing angles).

For the CO₂ sensor, zero-point calibration was performed by pulling a calibration pin for 7 seconds, after it stood in the experimental environment for 20 minutes. The CO₂ slope was calculated for each sliding window (S5_CO2_S1ope column). The potentiometer connected to the sound sensors was set to high quality settings. Both PIR sensor trimpots (for sensitivity/sensing time) were set to maximum quality. The test lasted 4 days. Target values range from 0-3 people (4 classes). Since data gets registered wirelessly every 30 seconds, the dataset is categorised as "big data", due to its heightened "velocity" (one of the 5 V-s of big data).

2.1.1 Data Cleaning

The dataset's date and time columns were dropped, as weekday/date/time cannot influence the number of people present in a controlled environment. Then, after CO₂ values were rounded to 2 decimal points, all duplicate rows were removed. No null values were found.

2.1.1.1 Data preprocessing

The experiment proceeded with normalising data, using minmax normalisation. This method ensures that column values are in [0,1] (unlike with mean-based normalisation).

Dataset initial structure				
Column	Datatype	Type	Unit	Description
Date	Date	Feature	YYYY/ MM/DD	The date of the measurement
Time	Date	Feature	HH:MM:SS	The time of the measurement
S1_Temp - S4_Temp	Real number	Feature	Celsius	The temperature measured by a sensor, in location 1 - location 4
S1_Light - S4_Light	Integer	Feature	Lux	The light measured by a sensor, in location 5 - location 8
S1_Sound - S4_Sound	Real number	Feature	Volts	The sound measured by a sensor, in location 9 - location 12
S5_CO2	Integer	Feature	PPM (parts per million)	The CO ₂ value measured in just one location, by a sensor that has a wide range of measurement
S5_CO2 Slope	Real number	Feature	None	This was determined from the numeric evolution of the previous CO ₂ value
S6_PIR, S7_PIR	Boolean	Feature	None	The motion detection value measured by a sensor, in location 13 and location 14
Room Occupancy Count	Integer	Target (ground truth)	None	The number of people present in a room. It was added manually by the scientists that collected the data.
Balanced Dataset structure			Imbalanced Dataset structure	
S1_Temp - S4_Temp S2_Light, S4_Light S1_Sound - S4_Sound S5_CO2_Slope S6_PIR, S7_PIR Room_Occupancy Count			S1_Temp - S3_Temp S1_Light - S4_Light S5_CO2_Slope S6_PIR, S7_PIR Room_Occupancy Count	

Table 2: Dataset initial structure as provided by [Singh et al., 2018], Balanced and Imbalanced dataset structure.

The ground truth column was left unnormalised. Then, the data was balanced from (8793, 17) to (1836, 17). Both the balanced/imbalanced datasets were used to train/test models.

Now, the way in which the data was balanced shall be explained. First of all, the least represented class was determined. Then, data was reduced from all other classes to match the number of entries from the class that had the least amount of entries (dataset rows). As such, all dataset classes ended up having the exact same number of entries.

An alternative would have been augmenting the data of all classes besides the most represented one, to match its row count. Yet, any such data augmentation would have been redundant, as it would have done little besides duplicate the information that was already present in the dataset. As such, the data subtraction method was chosen to enable fast model training, without discarding important data. For this purpose, a lot of data was removed from the vastly overrepresented "0" class, and almost none from the other classes. Since the "0" class had extremely similar datarow values, their loss was not significant.

2.1.1.2 Feature Selection

Since the pairwise analysis and correlation matrix of the initial experiment revealed little information about feature correlation, a threshold-based function was made to determine correlated dataset columns. After using other such feature ranking methods, the final sensor setup that yields a great global/per-class prediction accuracy can consist of only 11 sensors, instead of the initial 15. This reduces initial/operational sensor costs, as well as storage needs.

2.2 Model training method

A `compute_model` function was made to perform input/output and training/testing data separation, while also training/evaluating/visualising all SkLearn models. It outputs training/testing accuracy, weighted F1/F2 scores, MSE/RMSE, a classification report and a confusion matrix. Such metrics were collected for the non-SkLearn NN model too.

The balanced dataset was used to train numerous models with unique hyperparameter combinations, using GridSearchCV. Five-fold crossvalidation was used to rank all GridSearchCV models based on accuracy. Once the five most accurate models were identified, they were also tested on imbalanced data, which might be common in live data feeds.

3 Classification models

The section includes the presentation of the classification algorithms from SkLearn that are used in the analysis [Bishop and Nasrabadi, 2006], including the main hyperparameters with specified values [Pedregosa et al., 2011].

C1. Gradient Boosting Classifier uses gradients to update weights, reducing bias and improving generalization. It builds additive models with regression trees for performance and interpretability [Friedman, 2001]. `RandomForestClassifier`'s evaluation metrics similar to those of `GradientBoostingClassifier`, when `random_state=43`, was used. Yet still, `GradientBoostingClassifier`'s results are more consistent regardless of `random_state`, making it more reliable.

C2. HistGradientBoostingClassifier is a faster, histogram-based version of Gradient Boosting, optimized for large datasets (`n_samples ≥ 10.000`) and inconsistent data (NaN values). Designed for real-time, big data streams [De-La-Hoz-Franco et al., 2018], it is preferred over Gradient Boosting Classifier despite slight accuracy differences.

C3. RandomForestClassifier averages multiple decision trees to improve classification [Breiman, 2001]. For it, trees were trained on subsets of CV data to prevent identical results, which would reduce the model to a single tree [Pintea, 2025]. [Khan et al., 2023] registered their best RF performance using `random_state=43`, but in this study `random_state=None` improved predictions the most.

C4. BaggingClassifier distributes data subsets to multiple classifiers and aggregates predictions via averaging or majority vote [Breiman, 1996]. It offers four data sampling methods and requires only 20 estimators for high accuracy. Slightly reducing `max_sample` to 0.9 improved results as well.

C5. ExtraTreesClassifier [Geurts et al., 2006] is similar to a RF tree, but it gets trained on all the training dataset data. Its nodes specialize in feature subsets. It's faster than RF, optimized with `max_features=None` and `random_state=43` via GridSearchCV for improved accuracy.

Model (ordered by accuracy)	Hyperparameters
GradientBoosting-Classifier	learning_rate=0.9, random_state=43, warm_start=True, init='zero'
HistGradient Boosting	interaction_cst="pairwise", random_state=43, class_weight="balanced"
RandomForest Classifier	criterion="entropy", max_features="sqrt", bootstrap=True, random_state=None, warm_start=True, class_weight="balanced"
BaggingClassifier	n_estimators=20, max_samples=0.9, max_features=0.8, bootstrap=False, bootstrap_features=True, oob_score=False, warm_start=True, random_state=43
ExtraTrees Classifier	max_features=None, random_state=43, warm_start=True
Decision Tree Classifier	splitter="random"
NN Conv1D	Conv1D(16, 3, activation='relu', input_shape=(X_train.shape[1], 1)); MaxPooling1D(2); Flatten(); Dense(32, activation='relu') Dense(4, activation='softmax') model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
MLPClassifier	activation="tanh", solver="lbfgs", learning_rate="adaptive", nesterovs_momentum=False, early_stopping=True, warm_start=True
SVC	kernel="poly", degree=5, gamma="scale", shrinking=True, cache_size=10, decision_function_shape="ovo", random_state=43
Kneighbors Classifier	n_neighbors=4, weights="distance", algorithm="brute", leaf_size=1, p=1, metric="braycurtis"
ExtraTreeClassifier	class_weight='balanced', criterion="entropy", max_features="log2"
PassiveAggressive-Classifier	fit_intercept=True, early_stopping=True, validation_fraction=0.2, loss="squared_hinge", random_state=43, warm_start=True, average=True
RidgeClassifierCV	alphas=1, fit_intercept=True, class_weight=None
LinearSVC	penalty="l1", class_weight="balanced"
SGDClassifier	loss='hinge', penalty=None, fit_intercept=True, shuffle=True, random_state=43, learning_rate='optimal', early_stopping=True, warm_start=True, average=True
RidgeClassifier	class_weight='balanced'
Perceptron	penalty="l1", eta0=0.00001, random_state=43, early_stopping=True, validation_fraction=0.14, warm_start=True
RadiusNeigh-borsClassifier	radius=1, weights="distance", algorithm="auto", leaf_size=30, outlier_label=None
NearestCentroid	metric="euclidean", shrink_threshold=6
AdaBoostClassifier	n_estimators=40, learning_rate=0.9, random_state=43

Table 3: Hyperparameters deployment for ML models.

Model (ordered by accuracy)	Acc	Precision	Recall	MSE	RMSE	F1	F2
GradientBoostingClassifier	0.9884	0.9886	0.9884	0.0115	0.1075	0.9884	0.9884
HistGradientBoosting	0.9855	0.9860	0.9855	0.0144	0.1202	0.9856	0.9855
RandomForestClassifier	0.9855	0.9855	0.9855	0.0375	0.1938	0.9855	0.9855
BaggingClassifier	0.9826	0.9834	0.9826	0.0173	0.1316	0.9828	0.9826
ExtraTreesClassifier	0.9826	0.9827	0.9826	0.0404	0.2011	0.9826	0.9826
DecisionTreeClassifier	0.9768	0.9769	0.9768	0.0462	0.2150	0.9768	0.9768
NN Conv1D	0.9760	0.9726	0.9711	0.0288	0.1699	0.9702	0.9704
MLPClassifier	0.9682	0.9691	0.9682	0.0549	0.2343	0.9684	0.9682
SVC	0.9624	0.9642	0.9624	0.1069	0.3270	0.9625	0.9622
KneighborsClassifier	0.9595	0.9599	0.9595	0.0404	0.2011	0.9596	0.9595
ExtraTreeClassifier	0.9450	0.9457	0.9450	0.1069	0.3270	0.9450	0.9449
PassiveAggressiveClassifier	0.9277	0.9278	0.9277	0.0982	0.3134	0.9274	0.9275
RidgeClassifierCV	0.9248	0.9234	0.9248	0.1763	0.4198	0.9233	0.9240
LinearSVC	0.9219	0.9216	0.9219	0.1473	0.3839	0.9217	0.9219
SGDClassifier	0.9190	0.9212	0.9190	0.1242	0.3525	0.9191	0.9188
RidgeClassifier	0.8988	0.8987	0.8988	0.2052	0.4529	0.8964	0.8972
Perceptron	0.8930	0.8987	0.8930	0.1878	0.4334	0.8877	0.8891
RadiusNeighborsClassifier	0.8843	0.8885	0.8843	0.3323	0.5765	0.8803	0.8815
NearestCentroid	0.8294	0.8372	0.8294	0.3352	0.5790	0.8200	0.8232
AdaBoostClassifier	0.7312	0.8477	0.7312	0.3843	0.6199	0.7172	0.7115

Table 4: Evaluating ML models for balanced datasets.

Model	Acc	Precision	Recall	MSE	RMSE	F1	F2
GradientBoostingClassifier	0.9946	0.9946	0.9946	0.0053	0.0732	0.9946	0.9946
HistGradientBoosting	0.9946	0.9946	0.9946	0.0053	0.0732	0.9946	0.9946
RandomForestClassifier	0.9946	0.9946	0.9946	0.0053	0.0732	0.9946	0.9946
BaggingClassifier	0.9935	0.9935	0.9935	0.0064	0.0802	0.9935	0.9935
ExtraTreesClassifier	0.9930	0.9930	0.9930	0.0112	0.1061	0.9930	0.9930

Table 5: Model Evaluation for Classification Algorithms (C1-C5) on imbalanced datasets.

C6. DecisionTreeClassifier. A decision tree [Quinlan, 1986] uses decision nodes to branch out to other decision nodes, that ultimately formulate a classification label. It is one of the most human interpretable models.

C7. Deep Neural Network (Conv1D) was used to convolute unidimensional data, compressing its information. It was trained for 60 epochs with early stopping, which resulted in its performance being comparable to RF. Yet, it is not as robust as RF.

C8. MLPClassifier is a multi-layer perceptron with fully-connected dense layers [Almeida, 2020]. SkLearn's MLP supports log-loss optimization via LBFGS, SGD, or Adam. GridSearchCV proved that LBFGS is optimal for the considered dataset.

C9. SupportVectorClassifier. [Cortes and Vapnik, 1995] estimates non-linear class boundaries with support vectors by defining hyperplanes. Its SkLearn extension supports non-linear boundaries.

C10. KNeighborsClassifier K-nearest neighbours considers the euclidean distance between datapoints to determine how likely they are to belong to the same class [Kramer, 2023]

Model	Acc	Precision	Recall	MSE	RMSE	F1	F2
VotingClassifier	0.9942	0.9942	0.9942	0.0057	0.076	0.9942	0.9942
OneVsOneClassifier	0.9913	0.9913	0.9913	0.0317	0.1783	0.9913	0.9913
OutputCodeClassifier	0.9884	0.9886	0.9884	0.0115	0.1075	0.9884	0.9884
ClassifierChain	0.9884	0.9886	0.9884	0.0115	0.1075	0.9884	0.9884
MultiOutputClassifier	0.9884	0.9886	0.9884	0.0115	0.1075	0.9884	0.9884
OneVsRestClassifier	0.9826	0.9829	0.9826	0.0173	0.1316	0.9827	0.9826

Table 6: Evaluating Aggregative models for balanced datasets.

C11. ExtraTreeClassifier This model consists of one extra-tree, which was previously defined when discussing the ExtraTreesClassifier model [Geurts et al., 2006]. Such models should only be used with ensemble methods.

C12. PassiveAggressiveClassifier (SkLearn) is designed for large-scale learning. It updates weights after incorrect predictions only, following an "aggressive" course of action (upon making correct predictions, it remains unadjusted, "passive") ([Crammer et al., 2009]).

C13. RidgeClassifierCV. Even if the ridge method usually converts its targets (ground truths) to either -1/1, the SkLearn implementation allows for multiclass classification. Its structure is equivalent to the RidgeClassifier model, but its results were improved through crossvalidation.

C14. LinearSVC. A variation on the previously discussed SVC, LinearSVC uses a linear decision boundary to separate classes. It uses a one-vs-rest scheme to take care of multiclass classification. LinearSVC supports both dense and sparse inputs.

C15. SGDClassifier [Bottou, 2010] uses linear stochastic gradient descent with a regularizer to prevent gradient explosion. Within this study, its performance was observed to not indicate gradient vanishing either.

C16. RidgeClassifier. Ridge Classifier [Horel, 1962] treats classification problems as a regression task. It was previously discussed when addressing RidgeClassifierCV. This implementation makes use of no crossvalidation and has different hyperparameters.

C17. Perceptron. Modeled after the biological definition of a perceptron, this model was initially only used for binary classification [Rosenblatt, 1958]. Yet, its SkLearn implementation enabled it to perform multiclass classification. The SkLearn Perceptron model was implemented as a wrapper for SGDClassifier.

C18. RadiusNeighborsClassifier considers all nodes in a certain radius of the current datapoint. The averaged "opinion" (usually classification label) of the closest points is considered to calculate the current point's class.

C19. NearestCentroid. Instead of measuring the distance to the nearest neighbour (like KNN [Steinbach and Tan, 2009]), the Nearest Centroid measures the distance to the nearest class centroid.

C20. AdaBoostClassifier. AdaBoost [Freund and Schapire, 1997] is a method that trains a classifier on the whole dataset, then increases weights of misclassified labels for training the next version of the initial classifier. Its number of estimators adjusts how many such classifiers are made.

4 Evaluation and Analysis

A depth analysis involves the use of many models. The five best ones (C1-C5) were detailed in the previous section followed by the supplementary tested models (C6-C20). Hyperparameters deployment for the ML models (C1-C20) are specified in Table 3. Table 4 shows the evaluation of the C1-C20 models for balanced datasets, while Table 6 presents the evaluation results for the aggregative models on the same balanced datasets. The evaluation of the best C1-C5 classifiers for the imbalanced dataset is detailed in Table 5. An extended analysis further includes several aggregative models (C21-C26).

4.1 Aggregative models

C21. VotingClassifier [Breiman, 1996] achieved 99.4% accuracy in this study. For this, combined five classifiers (C1-C5) using soft voting/majority rule to select the most voted label.

C22. OneVsOneClassifier required only one of the best classifiers identified by this study. Multiple instances of that classifier were trained for each identified class pair (e.g. 0 vs 1, 2 vs 3, ...), to best discern between them. `OneVsOneClassifier` is suitable for small datasets only. It achieved 99.1% accuracy in this study.

C23. OutputCodeClassifier. `OutputCodeClassifier` introduced by [Dietterich and Bakiri, 1994], represents each class label as an array of 1/0 values (similarly to onehotencoder). One binary classifier is fittest per bit stored. Classification is done by generating a datapoint using the input data, then choosing one of its closes neighbours' label.

C24. ClassifierChain proposed by [Read et al., 2011], performs multilabel classification using sequential binary classifiers. For it, each classifier is trained on all features and on previous predictions. `OutputCodeClassifier`, `ClassifierChain`, and `MultiOutputClassifier` all matched the accuracy of the original model within them (C1).

C25. MultiOutputClassifier. [Borchani et. al., 2015] trains separate classifiers per target, and is mainly used for binary-to-multitarget conversion. Using it with C1, which already supports multilabel, is redundant. Yet, it had to be tested to confirm that no performance improvements can be obtained.

C26. OneVsRestClassifier. fits one classifier per label. It is interpretable and robust as it uses lightweight binary classifiers. It is suited for large data, but underperformed on the considered sensor dataset.

5 Analysis, Discussions and Observations

5.1 Preliminary Observations

In order to go beyond the scope of evaluating AI models on static data, additional testing/training datasets were generated. This was done in order to attempt emulating varying sensor placements. In other words, the real-world data was adjusted to contain values that correspond to different sensor locations. This was done by applying a scalar change in each feature that was considered for training/testing AI models.

Each individual sensor was multiplied by a random value between [0.5, 1.5], which was kept consistent across rows, but not across columns. This corresponds with replacing each of the given sensors randomly within the given room, to enable studying the changes that may occur when changing the initial sensor placement. To achieve this, the initial feature values from the real-world dataset were multiplied by a random value between [0.5, 1.5] (as described), which is equivalent to repositioning the given sensors. Then, a data augmentation method was applied to the newly obtained real-world dataset, in order to obtain new data points that deviate slightly from the initial data.

This was done in order to both increase the size of the training/testing datasets, as well as to introduce noise into the considered data. In order to enable comparisons between the tests conducted on the newly generated data and the initial tests conducted on the unaltered dataset data, the training/testing data split ratio was kept constant. As such, the study can provide insight on model behaviour on both small and medium-sized datasets, as well as on augmented data.

It should be also noted however that this study does not aim to provide pre-trained models that can deal with varying sensor configurations. In order to satisfy such demands, the study provides a framework that can be inputted when any custom user data, to output models that can best analyse it, in order to offer predictions. Such an output is superior to the results that may be provided by a model trained on multiple sensor configurations.

While it may be considered that training models on just one sensor layout leads to heavily biased predictions, such biases do not lower the quality of the provided predictions. On the contrary, having models that are custom-trained on user sensor configurations can offer predictions that are better suited for the environment considered. Training models on site can help models learn customised baselines for air density, pollution, temperature, human inhabitation and other factors that may influence their predictions.

Regarding obstructions and sensor blind spots, little to no digital solutions can be provided. As such, they should be avoided by placing sensors on empty walls that cannot be obstructed. Signs informing individuals about the sensor's presence should be placed near sensor locations, instructing people to not obstruct them. To deal with sensor blind spots, customised sensor configurations should be provided to maximise room coverage. Sensors with large perception FOVs can also help alleviate this problem, even if they may prove to be more expensive than their alternatives.

Besides the aforementioned problems, outliers may also be introduced by transient occlusions. There is no way to avoid them, but they can simply be excluded when the data is preprocessed. Alternatively, a strategy that resembles the presented "broken sensor" data replacement technique can be employed to deal with missing/vastly inaccurate sensor readings.

Regarding improper sensor placement, not much can be done though AI or any digital means of correction. Only proper sensor placements that cover the entire area of the given room can determine good model predictions. In order to get sensor placements that function best with the proposed models, it would be advisable to follow the instructions provided by the authors of the dataset. Adarsh Pal Singh and Sachin Chaudhari [Singh et al., 2018]) describe the sensor layout they used to collect data in great detail, to ensure it can be replicated. They used a 6m x 4.6m room for their measurements and 7 sensor nodes placed in a heptagram formation. The interval for collecting data was set to 30s, to account for the hardware capabilities of all the sensors. This interval was also most likely chosen to ensure a significant shift in readings, to eliminate redundant data. All "Heating, Ventilation, and Air Conditioning" (HVAC) systems were excluded from the data collection process, to maintain the purity of the readings provided. Yet, in a real-life scenario such systems may not be dispensable and should be included to provide data that best represents a regular usage scenario. As such, it is advisable to consider the location of all HVAC elements and position sensors in such a way that reduces their effect on their readings to a minimum, without requiring for them to be disabled (as was the case for the currently used dataset). As such, each individual sensor setup should be done in a customised manner, to best account for the environment that contains the measuring and data collection hardware. Besides, the FOVs of all sensors should be considered, as well as their reach.

A problem that greatly affected the data collection process was the availability of the test room considered. The schedules of the various subjects that inhabited the room also influenced the data collection process. This is why data collection only lasted 4 days, which is quite a lengthy amount of time that most landowners cannot afford to spend on data collection. As such, this study proposes an accelerated method for data collection that can afford to collect more data in a short amount of time.

If all sensors can be set to capture readings every 10-15s, then the same amount of data collected by this study can be achieved in 1-2 days. For this to be accurate however, test subjects must be instructed to inhabit the given room in a way that describes the expected inhabitation patterns, performing a change in inhabitation levels at a rate that is double than the one expected.

In other words, the number of people present in the given room should be changed twice as fast as the shifts in inhabitation that are expected from the given location. This enables realistic data to be captured at a rate that enables for maximum data collection. In such a scenario, it may also be possible to involve more than 3 people, which is the maximum number of people that inhabited the room that was used for data collection. This assumes that it should be easier to sign up a lot of people for a short period of time than to ask a few people to partake in the study for a long period of time. Such an assumption is only sensible and needs no further justification.

5.2 Other Observations

From the models that can function optimally on limited computational resources, the DecisionTreeClassifier model has proven to be superior in terms of testing accuracy. Its internal information also revealed insightful information regarding the importance of the features considered. By visualising the internal DT structure, the order in which the features are considered can be analysed in order to rank them. Other simplistic models such as the KNeighborsClassifier, SVC and LinearSVC returned good testing accuracy values, but none can rival the DecisionTreeClassifier which proved its superior capabilities on most of the other evaluation metrics considered.

Given the limited capabilities of certain embedded systems, such a model may be the only one that can provide accurate results in real-time to represent room occupancy estimations. Yet, the centralised data collection and analysis unit does not have to be an embedded system. It can be a small computer, the likes of a Raspberry PI. If on-chip data analysis is preferred instead, then a DT model may be the only choice available, depending on the amount of RAM that can be considered. The RAM can vary significantly with price, usually going from 1MB to 64MB. Yet still, the first choice for a data collection and processing unit should be a small computer, not an embedded chip.

Real-time monitoring systems are required to have little to no latency. As such, both the hardware and the software needed to provide real-time responses has to be optimised. As the software consists of an AI model (which may contain more estimators), it can only be improved by altering its internal structure, or its hyperparameters. Since changing internal structures of models exceeds the capabilities of this study, hyperparameter search was conducted with rigour to ensure that all models are tuned to the given data.

Beside that, numerous models that can provide accurate predictions were provided, to enable for the selection of the most appropriate one for each use case. As such, models can be chosen by considering the computational capabilities of the hardware chosen by somebody that wishes to implement the system described in this paper.

It would be advised that they maximise their computational capabilities to a degree that grants them the predictive accuracy they desire, on the budget they have. If latency becomes an issue, then a different AI model should be chosen to enable either large data streams or limited processing capabilities.

Given that several of the models proposed have a simplistic internal structure, their low latency is sure to enable for real-time processing. If some of the highly accurate models fail to do so (on certain low-end hardware), then less capable but more robust models should be used instead (as suggested by the model ranking provided by this paper). This way, a system-wide latency of 200-400 ms should be achievable without any further modifications to the system.

5.3 Final Observations

The insight that was provided by the study until this point centered mostly around data science and other non-practical aspects of the considered problem. As such, this section aims to elaborate further on the problem, in order to offer solutions to the practical problems that a real-life sensor system may face. The first, most obvious issue that may plague such a real-life system is hardware failure.

Both the considered sensors and the data collection hardware utilised are subject to experiencing permanent or temporary failure. While some types of hardware failures may be dealt with without human assistance, some cannot.

If a sensor breaks, then a backup sensor can be used to replace it until the broken sensor is removed and fixed. This requires that for each required sensor, another one be installed to serve as backup. Such an approach would double sensor costs, without guaranteeing that the backup sensors are going to function properly when turned on. Such a problem can be attributed to were induced by sensor underusage, which would represent the default state of such sensors. As such, it may prove better to simply neglect broken sensors without using any backups. In order to fill in data gaps determined by broken sensors, replacement data can be sampled from past data. Such data will be chosen in accordance with the similarity between the other sensor readings and the formerly collected data, to provide reliable (accurate) data substituents. In the meantime, maintenance personnel should be sent to fix or replace the broken sensors.

If the data collection hardware breaks, then the system is simply going to shut down. In this case, the only way of keeping the system online would be to use a physical backup. As such, a hardware backup should be installed for this purpose when the system is first put into place. This should resolve any issues relating to the main data collection hardware, as long as precautions are taken to ensure that the backup suffers the least non-usage damage. It should be shielded from the environment and kept sealed in a container that minimises humidity and maintains a cold climate. In order to activate the backup, a hardware link should be made between the main component and the backup, to identify failures. Alternatively, the backup could be kept in a constant state of hibernation to be activated through WiFi by human maintenance personal. Both ways of activating the backup should be implemented.

Sensor ware is another issue that is known to cause constant issues to hardware systems. It is the primary cause of non-provoked sensor recalibration. As time goes by, sensors fall out of tune and provide readings that are not in line with their initial perception of the environment. This can only be corrected by replacing/recalibrating sensors at set intervals (e.g. every 2 years). Such replacements/recalibrations should not be costly, and can be determined through data analysis. For this purpose, classification problems such as "anomaly detection" may be employed.

Misreadings can also be determined by debris (dust/other). Such issues can only be solved through human intervention. As such, regular sensor cleaning is suggested to ensure that the collected data is as accurate as possible. Besides the sensors and the data collection hardware, the system's energy supply may also cause maintenance problems. This too, can only be solved by providing backup hardware. The backup can be activated in a manner that resembles the activation of the data collection unit. Yet, for this purpose, hardware links that trigger the activation of the backup electricity generator are commercially available and need no customisation.

In order to deal with varying ambient conditions, large datasets are needed. Such datasets can only be collected after keeping the sensor system online for months on end (which is something the current study cannot afford to do).

Continuous learning (self-learning) models can also help improve prediction accuracy on irregular/seasonal environment changes. Alternatively, different models can be trained for each season, to maintain predictive accuracy in the face of environmental drift. Such continuous learning models can be derived from any known model architecture, which includes the architectures proposed by this study.

5.4 Analysis and Discussions

This study has revealed that imbalanced data determines classification improvements only for the overrepresented class/classes. It also revealed that non-NN `SkLearn` models are quite powerful on numeric data. Data visualisation proved detrimental to identifying outliers. Consulting `SkLearn` hyperparameter documentation was detrimental in selecting the appropriate hyperparameters, which resulted in performant classification, [Pedregosa et al., 2011]. Such hyperparameters/models were searched in scientific papers as well. Such research defined the standards for expected model performance (accuracy). Such standards were reached (and exceeded) through the use of automatic hyperparameter selection (`GridSearchCV`) and data normalisation/balancing. It was noticed that across most confusion matrices generated by the imbalanced dataset, only the "0" class was predicted with great accuracy. On the balanced datasets however, the confusion matrices show great accuracy values for all classes, especially for the "0" and "3" classes. As such, the balanced dataset trained the models to generalise better, making it superior to the imbalanced dataset, despite it having considerably less rows.

Sound sensors proved to be the most useful at determining the number of people present. Temperature and light sensors were proved to be the least relevant for the same purpose. Even if the CO_2 column was dropped from the dataset, its slope (`S5_CO2_Slope`) proved very useful in determining the number of people present. As such, the CO_2 sensor is still in the final, proposed sensor configuration. As such, the types of sensors needed, as well as their position in the room can be determined.

It was also proven that both training/testing splits of 75%/25% and 80%/20% lead to performant model generalisation. This is the case when considering training/validation/testing splits as well. Within the `computer_model` function, the proportion of training/testing data was never altered, to ensure that models can be compared objectively. If the training/testing data ratio had been altered from one more to another, any performance improvement could have been attributed to the ratio, and not to the model structure. As such, no ratio modifications were made once the 75%/25% and 80%/20% ratios were deemed optimal.

The Conv1D NN underwent an ablation study, to ensure its robustness. The test was manual, not automated. Adding/removing layers and adjusting hyperparameter values did not modify the model's performance, which means that the model is robust. The acc/val_acc plots (Figure 1) reveal that only a slight overfitting can be observed between the testing/training datasets, as the distance between the plot lines is quite small.

HistGradientBoosting Classifier was chosen for a big data stream, as it is the most easily scalable. The model has proven itself to be reliable, as it was always tested on randomised balanced datasets. It's fault tolerance (as with others) is no more than 10 misclassified datapoints per class. This you can observe in all the balanced dataset correlation matrices. Since more than 300 rows were found in the testing dataset that was used to generate the correlation matrices, it can be stated that fault tolerance is low. As such, due to minimal fault tolerance, the models proposed in this paper (Algorithms C1-C5) are all performant. Leaving only 2 decimal points in the S5_CO2_Slope helped remove data rows with similar values, which made training more robust. It basically made training faster without discarding any vital information. Also regarding CO₂ levels, it was noted that no abnormal/dangerous levels of CO₂ were present in a dataset, as such no CO₂ outliers had to be removed.

Randomly selecting rows for data balancing ensures that each time the notebook code is executed, a new balanced dataset is created. As such, the performance of the models can always be studied on a new balanced dataset, to reduce dataset bias. In other words, randomising balanced dataset rows improves model generalisation capabilities. Ditching the Date/Time columns was done to make models learn independently from such metrics. There is no reason to think that the ground truth from 11/11/2024 12:00am and 11/11/2026 12:00am have any reason to be similar, if a generalising model is considered. Of course, if the dataset is collected in a place that has strong periodic data repetitiveness, such columns should be kept and used for training (e.g. during recess tourist attractions are prone to be more visited). Yet, this was not the case for the considered dataset. For the get_correlated_variables function, multiple threshold values were tried, which led to certain columns being removed from the balanced dataset. A model was trained with the remaining features, until its accuracy was deemed appropriate.

The lifecycle could be improved by performing data augmentation to balance datasets and test the final, and propose models on live big data streams.

Hundreds (and in one case thousands) of figures were generated (see [Github]). Their insights (Figures 1-2) were aggregated by this paper's Tables 3-10.

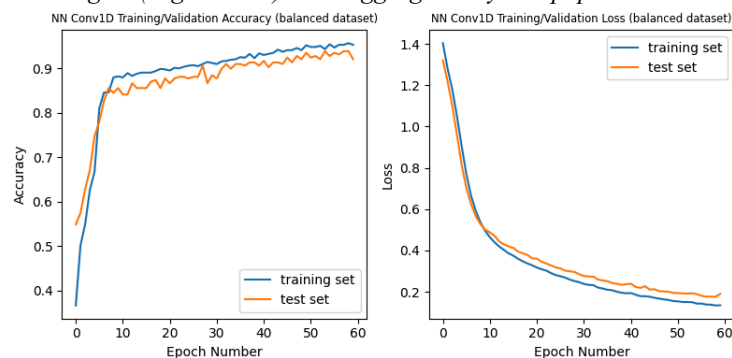
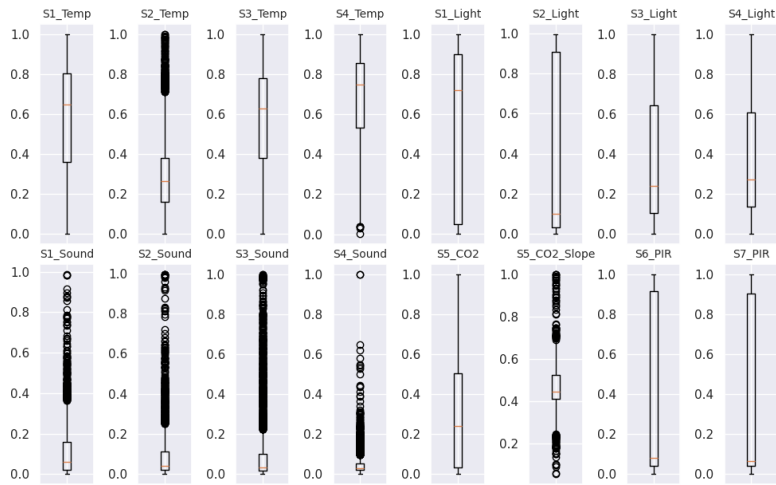
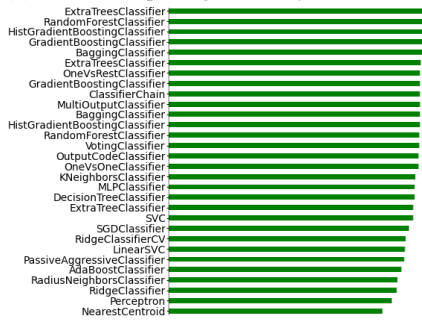


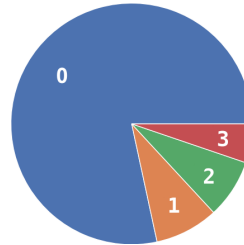
Figure 1: NN Conv1D Training/Validation Accuracy (left), Loss (right) NN train_acc/val_acc plots for balanced data.



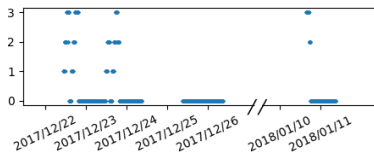
(a) Outlier boxplots generated for a realistic noisy HVAC 15 second datastream.



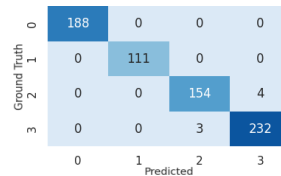
(b) Training accuracy ranking of all models on a realistic noisy HVAC 15 second datastream.



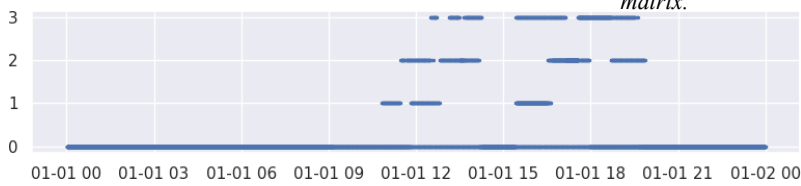
(c) Distribution of classification labels within the initial, unbalanced dataset.



(d) Data collection dates.



(e) HistGradientBoostingClassifier classification matrix.



(f) Data collection times.

Figure 2: Relevant examples of generated figures.

5.4.1 Feature and Model Ranking

In order to rank the given dataset's features, multiple feature ranking methods were used for objectivity (BFS/FFS, LASSO, Correlation Matrices, MAD, DT, MI, RF, Exhaustive Selection, RFE, Ridge, SelectKBest, Fisher, Variance Threshold). For this, only the most trustworthy unaugmented data was used.

This was attempted however and has provided results that are virtually identical to the ones provided by the real-life data. As such, only the initial rankings based on real-life data were kept for inclusion in this study.

When it comes to model ranking however, the initial data of the dataset was considered insufficient. In order to provide model rankings that best reflect diverse real-life scenarios, the data was augmented to account for different sensor placements, different HVAC conditions and noisy environments. Besides, a "15 second" data collection interval was simulated using the newly augmented data. This was done by linearly interpolating every 2 rows present in the dataset, which had a 30 second data collection interval between them. After averaging the model rankings of all the considered data augmentation methods, an average ranking of all considered models was included in this paper. The model rankings that were calculated for each data augmentation method individually were kept on the [Github] repository of the project. This was only done because they varied quite a lot more than their "feature ranking" equivalents which provided near identical rankings. Yet still, they did not vary significantly either. This proves that the models proposed by this study are generally applicable to different sensor placements, different HVAC conditions and varying levels of environment noise.

Hyperparameters for feature ranking techniques were adjusted in relation to the computational capabilities of the hardware to execute code for this study. A/B testing was also used to determine the best hyperparameters for the aforementioned methods. See ranking tables from: Feature Ranking (Table 7) and Augmentation-Agnostic Model Ranking (Table 8).

5.5 Sensitivity Analysis (Ablation Study)

An ablation study was performed for all features present in the balanced dataset, to ensure that they are all useful in optimizing model performances. In other words, exactly one feature was extracted from the dataset used to train models, to ensure that the removed variable determines a drop in model performance. This was done for all considered features and for all considered classifiers, to ensure objectivity. The results of the ablation study have been ranked by accuracy and per model, to best visualise their relevance. By inspecting the CSVs containing the aforementioned results, it can be clearly determined that there is no consistency between the considered models in terms of feature importance. For example, "BaggingClassifier" performed better than it did initially when one of the "S1_Sound", "S3_Sound", "S4_Sound", "S2_Sound" or "S5_CO2_Slope" columns was removed, but "AdaBoostClassifier" performed better than beforehand when one of the "S7_PIR", "S2_Temp", "S1_Temp", "S2_Sound", "S3_Temp" or "S5_CO2_Slope" features was removed.

Sensor	Fisher	Ridge	RFE	RF	LASSO	MAD	MI	DT
S1_Light	15	12	1	1	12	2	2	4
S3_Light	1	5	4	5	5	8	4	3
S3_Temp	11	3	2	6	3	1	6	1
S5_CO2_Slope	6	1	1	3	1	13	5	12
S1_Temp	12	8	7	11	8	3	7	7
S2_Light	14	14	1	2	13	10	1	5
S1_Sound	2	11	1	7	10	14	12	9
S4_Temp	16	4	10	13	4	4	10	6
S2_Temp	10	7	3	8	7	11	3	10
S4_Sound	7	2	12	15	2	16	13	11
S5_CO2	8	15	5	10	13	5	8	8
S2_Sound	3	6	9	4	6	15	9	13
S4_Light	4	16	8	12	14	9	8	2
S7_PIR	9	9	11	14	9	6	14	14
S3_Sound	13	10	6	9	11	12	11	15
S6_PIR	5	13	13	16	12	7	15	13

Sensor	Exh	SKB	VT	FFS	BFS	Avg	Final-Rank
S1_Light	1	1	1	1	1	4.15	1
S3_Light	1	1	1	10	10	4.46	2
S3_Temp	1	10	1	10	10	5.00	3
S5_CO2_Slope	1	10	10	1	1	5.00	3
S1_Temp	1	1	1	1	1	5.23	4
S2_Light	1	1	1	10	1	5.69	5
S1_Sound	1	10	10	1	1	6.85	6
S4_Temp	1	10	10	1	1	6.92	7
S2_Temp	1	10	10	10	1	7.00	8
S4_Sound	1	10	10	1	1	7.77	9
S5_CO2	1	10	1	10	10	8.00	10
S2_Sound	1	10	10	10	10	8.15	11
S4_Light	1	10	10	10	10	8.77	12
S7_PIR	1	10	1	10	10	9.08	13
S3_Sound	1	10	10	10	1	9.15	14
S6_PIR	10	10	1	10	1	9.69	15

Table 7: Feature Ranking. Notations: rank 1 for True/ 10 rank False; RF (RandomForest), MAD (Mean Absolute Difference), MI (mutual Information), Exh (Exhaustive), SKB (Select KBest), VT (Variance Threshold), RFE (Recursive Feature Elimination).

While removing "S5_CO2_Slope" or "S2_Sound" improve performance for both the aforementioned model, this is inconsistent with the results outputted by the rest of the models. For example, no columns could be removed from the dataset to improve the performance of "DecisionTreeClassifier" (the initial dataset structure was superior to all others). As such, no generally valid feature ranking can be determine from the ablation study. Yet, once a system designer has decided on which of the proposed models they wish to utilise, the ablation study information can prove very useful. Such information can mainly be used to reduce the number of sensors in the system, which can result in rather substantial expense reductions (especially if large surveillance areas are considered).

Table 9 reveals the model ranking of all models used in the ablation study. The order of the mentioned features represents model performance, as all models trained on the considered datasets were sorted (in a descending manner) by training accuracy.

Model	Avg 15 sec	Avg HVAC	Avg HVAC 15 sec	Avg	Rank
ExtraTreesClassifier	1.25	2.00	4.75	2.67	1
RandomForestClassifier	5.00	3.00	2.50	3.50	2
HistGradientBoostingClassifier	3.75	6.13	3.25	4.38	3
OneVsRestClassifier	7.13	6.00	4.50	5.88	4
VotingClassifier	2.75	9.25	6.25	6.08	5
BaggingClassifier	8.88	5.13	5.00	6.33	6
OutputCodeClassifier	4.25	11.38	5.00	6.88	7
GradientBoostingClassifier	10.13	4.00	9.75	7.96	8
ClassifierChain	11.38	5.63	9.25	8.75	9
MultiOutputClassifier	10.13	6.13	10.25	8.83	10
OneVsOneClassifier	10.63	11.63	13.25	11.83	11
KNeighborsClassifier	14.00	13.75	12.38	13.38	12
MLPClassifier	11.38	14.38	14.75	13.50	13
DecisionTreeClassifier	14.00	14.38	13.38	13.92	14
SVC	16.00	16.88	16.00	16.29	15
ExtraTreeClassifier	16.63	16.00	17.00	16.54	16
AdaBoostClassifier	18.38	20.63	17.88	18.96	17
SGDClassifier	19.88	18.00	21.75	19.88	18
PassiveAggressiveClassifier	19.38	21.00	20.00	20.13	19
LinearSVC	20.88	20.50	19.38	20.25	20
RidgeClassifierCV	22.63	20.25	21.38	21.42	21
RadiusNeighborsClassifier	24.25	22.50	23.38	23.38	22
RidgeClassifier	24.13	23.75	23.25	23.71	23
Perceptron	22.63	25.00	24.88	24.17	24
NearestCentroid	25.75	26.00	25.63	25.79	25

Table 8: Augmentation-Agnostic Model Ranking. Average of ranking notations: Avg 15 sec, Avg HVAC repositioned HVAC noise, Avg HVAC 15 sec (Avg rank repositioned HVAC noise 15 sec).

The column names mentioned in the table represent the feature of the initial dataset that was removed in order to achieve the ranking shown in the table (by row number). The removal of features that rank closer to the top of the table proved to improve model performance more than the removal of the ones below them. Cells containing the value "NONE" represent models that were trained on the initial dataset structure (without removing any columns). As such, the table makes the features that should be removed from each model's dataset clearly visible, even if no generally-applicable (model agnostic) conclusions can be drawn from it. With this being said, the balanced dataset structure was kept unaltered for the rest of the study.

Table 10 reveal the amount of RAM memory occupied by each model, as well as their average prediction times. The models were sorted in a descending order by both aforementioned metrics, to enable for an immediate visualisation of their time-based and size-based rankings. By considering the information from Tables 9-10 and from model accuracies, models can be chosen based on hardware capabilities. Table 10 containing the time-based ranking of models also reveals that all models are fit for real-time usage, as they are all vastly under 100 milliseconds. Yet, most of latency introduced in the system is going to originate from the hardware, which may or may not allow for any of the aforementioned models to be used in real-time scenarios. As far as RAM memory usage goes, the largest model shown in the above table has a RAM size of 3.5 MB.

AdaBoost Classifier	S7_PIR S2_Temp S1_Temp S2_Sound S3_Temp S5_CO2_Slope NONE S4_Sound S4_Light S6_PIR S1_Sound S3_Sound S2_Light
Bagging Classifier	S1_Sound S3_Sound S4_Sound S2_Sound S5_CO2_Slope NONE S6_PIR S2_Temp S1_Temp S7_PIR S4_Light S2_Light S3_Temp
Kneighbors Classifier	S6_PIR S7_PIR S2_Sound S4_Light S3_Sound S1_Sound NONE S5_CO2_Slope S4_Sound S3_Temp S2_Temp S1_Temp S2_Light
LinearSVC	S2_Temp S5_CO2_Slope S7_PIR S2_Sound NONE S4_Sound S1_Sound S6_PIR S3_Sound S4_Light S2_Light S3_Temp S1_Temp
MLPClassifier	S6_PIR S2_Sound S1_Sound S4_Light NONE S7_PIR S4_Sound S5_CO2_Slope S2_Temp S3_Temp S3_Sound S1_Temp S2_Light
MultiOutput Classifier	S2_Light NONE S1_Sound S4_Light S1_Temp S6_PIR S5_CO2_Slope S2_Sound S4_Sound S7_PIR S3_Temp S3_Sound S2_Temp
RidgeClassifier	S6_PIR S7_PIR S1_Sound S2_Sound S2_Temp S3_Sound NONE S4_Sound S4_Light S5_CO2_Slope S1_Temp S2_Light S3_Temp
SGDClassifier	S7_PIR S4_Sound S5_CO2_Slope S6_PIR S1_Sound S2_Sound S4_Light NONE S2_Temp S3_Sound S2_Light S3_Temp S1_Temp
VotingClassifier	NONE S2_Light S5_CO2_Slope S1_Sound S4_Sound S7_PIR S6_PIR S3_Temp S3_Sound S2_Sound S1_Temp S4_Light S2_Temp
SVC	S7_PIR S2_Sound S6_PIR S3_Sound NONE S4_Sound S2_Temp S1_Sound S4_Light S3_Temp S5_CO2_Slope S2_Light S1_Temp
RidgeClassifier CV	S7_PIR S2_Sound S3_Sound S1_Sound S6_PIR NONE S4_Sound S2_Temp S5_CO2_Slope S3_Temp S4_Light S2_Light S1_Temp
RandomForest	NONE S7_PIR S6_PIR S1_Temp S2_Light S3_Sound S2_Sound S4_Light S1_Sound S5_CO2_Slope S2_Temp S4_Sound S3_Temp
NearestCentroid	NONE S4_Sound S3_Sound S2_Sound S1_Sound S5_CO2_Slope S4_Light S2_Temp S6_PIR S3_Temp S7_PIR S1_Temp S2_Light
RadiusNeighbors	S7_PIR S1_Sound S3_Sound S2_Sound S6_PIR S4_Sound S5_CO2_Slope NONE S2_Temp S4_Light S3_Temp S1_Temp S2_Light
Perceptron	S2_Temp S6_PIR NONE S3_Sound S4_Light S4_Sound S2_Sound S1_Temp S2_Light S7_PIR S1_Sound S3_Temp S5_CO2_Slope
PassiveAggressive	S7_PIR S4_Sound NONE S6_PIR S1_Sound S2_Temp S2_Sound S3_Sound S5_CO2_Slope S4_Light S3_Temp S2_Light S1_Temp
OneVsOne Classifier	NONE S7_PIR S6_PIR S1_Sound S1_Temp S5_CO2_Slope S2_Sound S3_Temp S4_Light S4_Sound S2_Temp S2_Light S3_Sound
OneVsRest Classifier	S5_CO2_Slope S2_Sound S2_Light S1_Temp S4_Sound S3_Temp S1_Sound S2_Temp NONE S3_Sound S4_Light S6_PIR S7_PIR
OutputCode Classifier	NONE S2_Light S5_CO2_Slope S4_Sound S1_Temp S6_PIR S2_Temp S3_Temp S4_Light S3_Sound S1_Sound S2_Sound S7_PIR
HistGradient Boosting	S1_Sound S4_Sound S7_PIR S3_Sound S2_Light S2_Sound S4_Light S5_CO2_Slope NONE S6_PIR S1_Temp S3_Temp S2_Temp
ClassifierChain	S2_Light NONE S1_Sound S4_Light S1_Temp S6_PIR S5_CO2_Slope S2_Sound S4_Sound S7_PIR S3_Temp S3_Sound S2_Temp
DecisionTree	NONE S7_PIR S4_Sound S5_CO2_Slope S4_Light S2_Sound S1_Sound S6_PIR S3_Sound S1_Temp S2_Temp S3_Temp S2_Light
ExtraTree Classifier	S5_CO2_Slope S3_Sound S7_PIR S1_Sound S2_Temp S6_PIR S3_Temp S1_Temp NONE S4_Sound S2_Sound S2_Light S4_Light
ExtraTrees Classifier	S1_Temp S7_PIR S1_Sound S2_Sound NONE S3_Sound S3_Temp S2_Temp S4_Light S5_CO2_Slope S4_Sound S6_PIR S2_Light
Gradient Boosting	S2_Light NONE S1_Sound S4_Light S1_Temp S6_PIR S5_CO2_Slope S2_Sound S4_Sound S7_PIR S3_Temp S3_Sound S2_Temp

Table 9: Ranking of all features that can be removed from each stated model to improve performance (ablation study).

Model	RAM Size (Bytes)	Model	Avg. Pred. Time (MS)
LinearSVC	1287	ExtraTreeClassifier	0.004377
RidgeClassifier	1422	SGDClassifier	0.004409
RidgeClassifierCV	1488	Perceptron	0.004491
Perceptron	1640	RidgeClassifierCV	0.004671
NearestCentroid	1739	PassiveAggressiveClassifier	0.005378
SGDClassifier	2203	RidgeClassifier	0.005406
PassiveAggressiveClassifier	2224	LinearSVC	0.006766
DecisionTreeClassifier	10912	GradientBoostingClassifier	0.008911
SVC	25518	NearestCentroid	0.009091
ExtraTreeClassifier	25712	MultiOutputClassifier	0.009777
AdaBoostClassifier	26218	BaggingClassifier	0.017784
MLPClassifier	34531	SVC	0.020799
KNeighborsClassifier	108881	RandomForestClassifier	0.023185
RadiusNeighborsClassifier	131834	OneVsRestClassifier	0.023871
BaggingClassifier	241594	ClassifierChain	0.026191
GradientBoostingClassifier	313477	MLPClassifier	0.027635
MultiOutputClassifier	313729	OneVsOneClassifier	0.028554
ClassifierChain	313834	AdaBoostClassifier	0.030529
OneVsOneClassifier	354440	KNeighborsClassifier	0.034882
OneVsRestClassifier	399931	DecisionTreeClassifier	0.035161
OutputCodeClassifier	521117	OutputCodeClassifier	0.055842
RandomForestClassifier	887183	ExtraTreesClassifier	0.060277
ExtraTreesClassifier	888597	RadiusNeighborsClassifier	0.074584
HistGradientBoostingClassifier	1169862	HistGradientBoostingClassifier	0.090852
RandomForestClassifier	1411385	RandomForestClassifier	0.109600
VotingClassifier	3499648	VotingClassifier	0.184881

Table 10: Ranking of all considered models, based on RAM usage and average prediction time.

If on-chip processing is expected, then at least 8 MB of RAM should be made available, to ensure that embedded OS variants (such as Linux = 4 MB) can run seamlessly. Yet, a centralised data processing unit is detrimental to the system's design, as it removes the need for code conversion to assembly/Embedded C. Using such a unit (with gigabytes of RAM) enables for the Python code and models provided by this study to be utilised without any modifications. Yet, while gigabytes of RAM may be ideal, costs can be lowered vastly by using embedding system for AI processing instead (if code conversion costs are not considered).

The results of this study can be improved in various ways, using methods such as edge computing or IoT devices. Tools such as FlexSlice [Mohajer et al.,2025] can be used to optimise the allocation of resources in large sensor-based surveillance networks. An example of such a resource optimisation is selective sensor activation. If sensors located next to entry points detect an unexpected number of inhabitants, they can activate such sensors in the area surrounding them. Until then, all sensors that are not in close proximity to entry points can be deactivated, to save energy and reduce operational costs. CCTV systems can be activated locally, in a similar manner.

Useful patterns of dynamic offloading can also be determined using graphs, to enhance the ways in networks are sliced. Network slicing in the context of the current study can be used to activate sensors selectively, throughout areas that are close in size to the room that was used to collect the "Room Occupancy Estimation" dataset. This is especially important when large areas need to be surveyed in an energy efficient way, considering varying traffic patterns. For such applications, tools such as FlexSlice should be used. The purpose for using such tools should be maximising the amount of dormant sensor areas without compromising on the detection capabilities of the system. While virtual simulations should be conducted first (using NS-3.26, or other virtual network slicers), physical tests should also be conducted on the surveyed area, once the entirety of the system is put into place. For such tests, sensors should be calibrated to the different HVAC conditions that each network slice has to deal with. Digital adjustment tools should also be put into place to enable system maintainers to change the parameters of the systems, if any environmental parameters change (e.g. from one season to another).

5.6 Sensor Location Optimisation

Each column containing sensor readings was augmented to emulate different sensor locations. This was done by simply multiplying the normalized values of those column by a scalar, then normalizing them. MinMax normalization was used both times. While it was assumed that sensor readings increase linearly with distance, it cannot be denied that all sensors used by this experiment perceive larger values when closer to inhabitants. Light, temperature, sound as well as CO₂ all increase in value when sensors are positioned closer to the subjects they are meant to track. Since no superior method to the one mentioned could be found for numeric sensor data (in the considered literature or through A/B testing), it was used repetitively for all sensor columns.

In order to determine by how much sensor values should be increased/decreased (to describe sensor replacements), multiple data augmentation intervals were considered. To ensure that the hardware used in this study is able to sustain the computational load of the interval search, intervals of 0.1 were considered, between 0 and 2. In other words, all intervals such as [0, 0.1], [0.1, 0.2], ... [1.8, 1.9], [1.9, 2.0] were considered for each column individually. The sole purpose of using intervals rather than fixed values was to enable for random value selection, within the considered interval. For absolutely each of the columns and each of the intervals mentioned, a well-performing robust model was trained. For this, the study has used the DecisionTreeClassifier model, which achieved a testing accuracy of about 0.968 on unaugmented balanced data. No other robust model has surpassed it on such data. As such, this model was used for all of the following tasks, with the best hyperparameters that were previously determined on unaugmented data. It is also worth mentioning that while multiplying columns by 1.1 or 0.9 may not seem significant, such changes are more pronounced when augmented normalised data that lies between 0 and 1 (which is where MinMax normalisation puts it). After all of the aforementioned models were trained for each column, the ones with the superior testing accuracy were labelled as being the best. Yet, so far, only one sensor has been considered for augmentation.

At this point, knowing the best intervals for data augmentation should be used to determine all possible configurations in which all sensors can be augmented at the same time, to optimise sensor positioning.

While this study does not explicitly provide a correlation between the augmented data values and the distance that lies between sensors and subjects, such information can be determined empirically, on site (using A/B testing while monitoring live sensor values). In other words, sensors can be repositioned until their readings match the augmented output values of this study. With this being said, the next stage of optimisation shall now be explained further.

To determine which sensors should be moved and by how much, the intervals that provided the best predictions are used for each of the considered sensors. Multiple arrays matching the number of sensor in length are generated, to represent all True/False combinations available. If the number of sensor is 3, then all the combinations would be 000, 001, 010, 011 and 111. This helps to determine which of the best augmentation intervals of certain sensors can go along optimally with others. Since this study has kept 12 sensors in its balanced dataset, $2^{12} = 4096$ sensors had to be trained. This almost exceeded the RAM available on the device, requiring intensive garbage collection at every step. This is the main reason why the imbalanced dataset was not used and why one of the most robust models was used here as well (the `DecisionTreeClassifier`). After all models finished their training and evaluation, a CSV containing all their evaluation metrics was created. This CSV contained the evaluation metrics of an identical model trained on the initial, unaugmented, balanced dataset. About 2700 models surpassed its testing accuracy. In other words, the accuracy of the `DecisionTreeClassifier` was improved from 0.968 to 0.991, by optimising sensor placement (theoretically, though data augmentation). Since there are about 2700 sensor placements described by the data augmentation intervals used, it can be assumed with confidence that at least one of them can be replicated in practice.

Beside optimizing sensor positioning, the intervals chosen by this study can also help determine sensor ranges and other specifications. If a system designer knows that certain sensors provide better readings from afar rather than from closeby for the problem considered by this study, they may choose sensors that minimise costs around such characteristics. Once such details are sorted out, it is advisable to segment the area that needs surveillance into portions equivalent in size with the room that was considered for this study. This ensures that both coverage and blind spots are sorted out optimally, while also maximising the accuracy of inhabitation monitoring.

It is also worth mentioning that all the data preprocessing steps that were used within the unaugmented part of the study were used in the augmented part of the study as well. Essentially, the exact same columns were dropped and truncated in both parts of the study. This ensured that models from both aforementioned parts of the study can be compared amongst themselves seamlessly. As such, no improvements that occurred from one part to another can be attributed to anything else other the interval-based data augmentation technique proposed by this study. As far as the author is aware, this method or any implementations of it are not provided anywhere else, apart from this study.

Its implementation is not an adaptation of any pre-existing code base and was conducted solely by the author of this study. As such, this study has made its source code and its outputs public, available at no cost on Github.¹

While the proposed can detect the number of people present in a room, it cannot offer any further information. Future work may include assessing more details regarding the people that inhabit the room, besides their number. At this point however, the system can be used instead of CCTV for periods of time in which no people are expected to be in certain rooms. This would entail shutting down CCTV cameras for such periods, to reduce electricity costs. The proposed system may trigger CCTV systems back into place once it detects unexpected levels of inhabitation at any given point in time. Overall, the system could be part of a larger AI surveillance system that may include face recognition and other such capabilities.

6 Conclusions and Future Work

Sensor data was successfully used to predict the number of people present in a room. This was accomplished after the data was cleaned/preprocessed/balanced and visualised to remove outliers. Scientific literature references helped adjust model hyperparameters, when GridSearchCV performance was stagnant. Twenty models were considered initially, out of which the five best ones were proposed for usage on live data. `HistGradientBoostingClassifier` was deemed the most fit for big data live streams. Six more (bulky, aggregative) models were trained using the estimators of the five best models. Two of the latter (`VotingClassifier`, `OneVsOneClassifier`) exceeded the accuracy of `GradientBoostingClassifier`. They were not proposed for big data streams due to lack of robustness.

The novel sensor location optimisation technique proposed has helped improve model performance drastically, as the data made public by this study reveals. For the model that was used to test it half of the 4096 computed sensor location configurations were proven superior to the initial one.

After considering numerous AI models, it was proven that accurate prediction results can be obtained with fewer sensors than the initial dataset provided. This means that the number of people in a room can be determined by using very few sensors, which reduces initial/operational costs, reducing for example smart homes costs and encouraging innovation in building automation systems.

Even if typically used on image data exclusively, convolutional layers were shown to function optimally for numeric sensor data too. The literature review papers reached accuracies of 98.1%, which were exceeded by the models in this study (on a different sensor dataset). A boosting classifier reached 98.8% accuracy, while a vote-based model reached 99.4% accuracy. Yet, the histogram-based boosting classifier was chosen ultimately for big data live streams, due to superior robustness.

Future work may include adjusting models to consider physical sensor ware, correcting inaccurate readings to retain predictive accuracy. The models developed by this study may have practical application that reach beyond the dataset's initial purpose. The sensors present in the dataset may be used to study how many people are present in a room at night, in low light conditions. A sensor-based method that does not need any light can be developed to replace CCTV in low light conditions, in some instances.

¹[Github] <https://github.com/AlexandruPintea2000/Sensor-based-room-inhabitation-monitoring>.

References

- [Almeida, 2020] Almeida, L. B.: “Multilayer perceptrons”; *Handbook of Neural Computation* (2020) C1.2, CRC Press.
- [Begum et al., 2024] Begum, M., Raja, G., Guizani, M.: “Ai-based sensor attack detection and classification for autonomous vehicles in 6g-v2x environment”; *IEEE Trans. Veh. Technol.*, 73, 4 (2024) 5054–5063.
- [Bishop and Nasrabadi, 2006] Bishop, C. M., Nasrabadi, N. M.: “Pattern recognition and machine learning”; (2006) New York, Springer.
- [Borchani et al., 2015] Borchani, H., Varando, G., Bielza, C., Larranaga, P.; “A survey on multi-output regression”; *Wiley Inter. Rev. Data Min. Knowl. Discov.*, 5,5 (2015) 216–233.
- [Bottou, 2010] Bottou, L.: “Large-scale machine learning with stochastic gradient descent”; *Proc. 2010 COMPSTAT Int. Conf. Comput. Stat.* (Aug) 177–186, Physica-Verlag HD.
- [Breiman, 2001] Breiman, L.: “Random forests”; *Mach. Learn.*, 45 (2001) 5–32.
- [Breiman, 1996] Breiman, L.: “Bagging predictors”; *Mach. Learn.*, 24 (1996) 123–140.
- [Chen and Guestrin, 2016] Chen, T., Guestrin, C.: “Xgboost: A scalable tree boosting system”; *Proc. 2016 ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.* (2016) 785–794.
- [Chitnis et al., 2025] Chitnis, S., Somu, N., Kowli, A.: “Occupancy estimation with environmental sensors: the possibilities and limitations”; *Energy Build. Environ.*, 6, 1 (2025) 96–108.
- [Cortes and Vapnik, 1995] Cortes, C., Vapnik, V.: “Support vector machine”; *Mach. learn.*, 20, 3 (1995) 273–297.
- [Mohajer et al., 2025] A. Mohajer, J. Hajipour and V. C. M. Leung, “Dynamic Offloading in Mobile Edge Computing With Traffic-Aware Network Slicing and Adaptive TD3 Strategy,”; *IEEE Communications Letters*, vol. 29, 1, 95–99 (2025)
- [Cottafava et al., 2019] Cottafava, D., Magariello, S., Ariano, R., et al.: “Crowdsensing for a sustainable comfort and for energy saving”; *Energy Build.*, 186 (2019) 208–220.
- [Crammer et al., 2009] Crammer, K., Kulesza, A., Dredze, M.: “Adaptive regularization of weight vectors”; *Proc. 2009 Adv. Neural Inf Process. Syst.*, 22.
- [De-La-Hoz-Franco et al., 2018] De-La-Hoz-Franco, E., Ariza-Colpas, P., Quero, J. M., Espinilla, M.: “Sensor-based datasets for human activity recognition—a systematic review of literature”; *IEEE Access*, 6 (2018) 59192–59210.
- [Dietterich and Bakiri, 1994] Dietterich, T. G., Bakiri, G.: “Solving multiclass learning problems via error-correcting output codes”; *J. Artif. Intell. Res.*, 2 (1994) 263–286.
- [Freund and Schapire, 1997] Freund, Y., Schapire, R.E.: “A decision-theoretic generalization of on-line learning and an application to boosting”; *J Comput. Syst. Sci.* 55.1 (1997) 119–139.
- [Friedman, 2001] Friedman, J. H.: “Greedy function approximation: a gradient boosting machine”; *Ann. Statist.*, (2001): 1189–1232.
- [Geurts et al., 2006] Geurts, P., Ernst, D., Wehenkel, L.: “Extremely randomized trees”; *Mach. learn.*, 63 (2006) 3–42.
- [Han and Zhang, 2020] Han, K. Zhang, J.: “Energy-saving building system integration with a smart and low-cost sensing/control network for sustainable and healthy living environments: Demonstration case study”; *Energy Build.*, 214 (2020) 109861.
- [Hastie and Tibshirani, 1998] Hastie, T., Tibshirani, R.: *Ann. Stat.*, 26, 2 (1998) 451–471,.
- [Huda et al., 2024] Huda, N. U., Ahmed, I., Adnan, M. et al.: “Experts and intelligent systems for smart homes’ transformation to sustainable smart cities: A comprehensive review”; *Expert Syst. Appl.*, 238 (2024) 122380.

- [Horel, 1962] Horel, A. E.: “Application of ridge analysis to regression problems”; *Chem. Eng. Progress*, 58 (1962) 54–59.
- [Hochreiter and Schmidhuber, 1997] Hochreiter S., Schmidhuber, J.: “Long Short-Term Memory”; *Neural Computation*, 9, 8 (1997) 1735–1780.
- [Khan et al., 2023] Khan, Y. A., Imaduddin, S., Singh, Y. P. et al.: “Artificial intelligence based approach for classification of human activities using mems sensors data”; *Sensors*, 23, 3 (2023) 1275.
- [Kramer, 2023] Kramer, O.: “K-nearest neighbors”; *Dimensionality reduction with unsupervised nearest neighbors*, *Intell.Syst. Ref. Libr.*, 51 (2013) 13–23, Springer, Berlin, Heidelberg.
- [Li et al., 2024] Li, T., Liu, X., Li, G., Wang, X., Ma, J., Xu, C., Mao, Q.: “A systematic review and comprehensive analysis of building occupancy prediction”; *Renew. Sustain. Energy Rev.*, 193 (2024) 114284.
- [Mäntyjärvi et al., 2004] Mäntyjärvi, J., Himberg, J., Kangas, P., et al.: “Sensor signal data set for exploring context recognition of mobile devices”; In 2004 *Int. Conf. on Pervasive Comput. PERVASIVE*, (Apr) 18–23.
- [Mohammed et al., 2023] Mohammed, O., Khound, P., Su, P. et al.: “Multilevel artificial intelligence classification of faulty image data for enhancing sensor reliability”; *Proc. 2023 Eur. Saf. Reliab. Conf., ESREL*, (2023) 1778–1783.
- [Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., et al.: “Scikit-learn: Machine learning in python”; *J. Mach. Learn. Res.*, 12 (2011) 2825–2830.
- [Pintea, 2025] Pintea, A.: “Studying Air Pollution with ML Classifiers”; *J. Environ. Acc. Manag.*, 13, 1 (2025) 41–51.
- [Quinlan, 1986] Quinlan, J. R.: “Induction of decision trees”; *Mach. Learn.*, 1 (1986) 81–106.
- [Read et al., 2011] Read, J., Pfahringer, B., Holmes, G., Frank, E.: “Classifier chains for multi-label classification”; *Mach. learn.*, 85 (2011) 333–359.
- [Rosenblatt, 1958] Rosenblatt, F.: “The Perceptron: A Probabilistic Model for Information Storage and Organization”; *Psychol. Rev.*, 65, 6 (1958) 386.
- [Schwee et al., 2019] Schwee, J. H., Johansen, A., Jørgensen, B. N., Kjærgaard, M. B., Matterna, C. G., Sangogboye, F. C., and Veje, C.: “Room-level occupant counts and environmental quality from heterogeneous sensing modalities in a smart building”; *Sci. Data*, 6, 1 (2019) 287.
- [Singh et al., 2018] Singh, A. P., Jain, V., Chaudhari, S., et al. (2018). “Machine learning-based occupancy estimation using multivariate sensor nodes”; In 2018 *IEEE Globecom Wkshps*, (Dec) 1–6, <https://www.archive.ics.uci.edu/dataset/864/room+occupancy+estimation>.
- [Steinbach and Tan, 2009] Steinbach, M., Tan, P-N.: “kNN: k-nearest neighbors”; *The top ten algorithms in data mining (2009)* 165–176, Chapman and Hall/CRC.
- [Teh et al., 2020] Teh, H. Y., Kempa-Liehr, A. W., Wang, K. I-K.: “Sensor data quality: A systematic review”; *J. Big Data*, 7, 1 (2020) 11.
- [Vasisht et al., 2018] Vasisht, D., Jain, A., Hsu, C.-Y., Kabelac, Z., Katabi, D.: “Duet: Estimating user position and identity in smart homes using intermittent and incomplete rf-data”; *Proc. 2018 ACM Interact. Mobile Wearable Ubiquitous Technol.*, 2, 2, (Oct) 1–21.
- [Venzke et al., 2020] Venzke, M., Klisch, D., Kubik, P. et al.: “Artificial neural networks for sensor data classification on small embedded systems”; *arXiv (2020) preprint 2012.08403*.
- [Zhang et al., 2024] Zhang, Z., He, J., Kumar, A., Rahman, S.: “AI-based space occupancy estimation using environmental sensor data”; *Proc. 2024 IEEE Power Energy Soc. Innov. Smart Grid Technol. Conf., ISGT* (Feb) 1–5.