Predicting Room Occupancy based on sensor metrics for enabling automation of temperature controls

Ahsan N.
College of Computing and Informatics
Drexel University
Philadelphia, PA
an948@drexel.edu

Dheeraj K.
College of Computing and Informatics
Drexel University
Philadelphia, PA
dkd989@drexel.edu

Seth F.
College of Computing and Informatics
Drexel University
Philadelphia, PA
sgf46@drexel.edu

Uyen N.
College of Computing and Informatics
Drexel University
Philadelphia, PA
un32@drexel.edu

August 2025

1 Abstract

The project aims at utilizing the data collected by a set of 7 room sensor nodes to record several room characteristics like temperature, light, sound, CO_2 emissions, rate of change in the CO_2 concentration and passive infrared (detecting motion) emissions. The sensor data was collected in a $6m$ x $4.6m$ room over several days and hours of **7** days, across **2** different weeks, every 30 minutes. So it covers both weekdays and weekend data. It includes the target value which determines number of people in the room based on the sensor data. The room occupancy varied between 0 to 3 during those days.

The project utilizes **4** different machine learning models and an ensemble model, to help predict the right room occupancy. Based on the metrics for **precision, recall, f-measure** and **accuracy**, we will arrive at a conclusion as to what model fits the best to help determine the correct room occupancy. The details are mentioned in the below sections, including an evidence based on principle component analysis (PCA) as to why the specific model would fit the best for the prediction.

2 Background and related work

Estimating the room occupancy based on the sensor features and buildings helps in providing knowledge to energy controls inside the room to save energy by efficiently determining the right operational controls required for the occupancy of a room. As we read through other experiments and projects, there have been developed some deep learning architectures that have proven effective in estimating the occupancy from the raw sensor data. The data has some meticulously collected features across the days which might convey useful information for occupancy estimation.

The data was published by **UCI room occupancy data** set benchmark, recorded across several days with 30 minutes interval. The details of the data set have been detailed out in below sections.

A modeling experiment based on a similar data set was conducted by Mao et al. 2014. They used Multinomoial Logistic Regression, Linear Discriminant Analysis (LDA), Multiclass Support Vector Machine (MSVM), Multi-layer Perception (MLP) Classifier, LightGBM, XGBoost, Random Forest models and recorded the metrics across all models. Their evaluation rendered Random Forest as the best model given it's weighted F1 score of 0.9985, AUC of 0.99996, and a Balanced Accuracy of 0995.

The other experiments around modeling was done by Candanedo and Feldheim in 2016. They predicted a binary room occupancy using the similar sensors where the models used were Logistics regression, SVM and Random Forest and the accuracy was attained to the scale of 95%.The best model that was determined was the Random Forest.

The other work that we looked at was the Real-Time Occupancy Estimation by Chen etal. in 2018, wherein they used Decision Tree, k-NN and Neural Networks, achieving the accuracy ranging between 83 to 90%. They did not provide the details of what and how they determine the best model in their approach.

In our approach, we have used four different models: Logistic Regression, k-NN, linear discriminant analysis and proportional odds. We also experimented with an ensemble model which determines using the four models we chose, to compare with individual model observations in terms of metrics to determine the best fit for the given data set to determine the predicted occupancy.

3 Data and Feature set details

The data includes a set of 19 columns, which includes 18 features and 1 target/label, which is the total room occupancy count for the corresponding features. It is a Multivariate Time-series data set published on **UC Irvine Machine Learning Repository**. Given the fact that the data was collected over several days which

includes weekends (probably), the data was class imbalanced, in favor of 0 occupancy room data. Some key features from the dataset are:

- Date
- Time
- S1_Temp - S4_Temp - Temperature readings from 4 sensors
- S1_Light – S4_Light - Light intensity readings from 4 sensors
- S1_Sound – S4_Sound - Sound level readings from 4 sensors
- S5_CO_2 - CO_2 concentration level
- S5_CO_2 Slope - Rate of change in CO_2 concentration
- S6_PIR, S7_PIR Passive Infrared (motion) sensor values
- Target Values: Room_Occupancy_Count
  Target variable: occupancy count {0, 1, 2, 3}

Values of each count:

Class/Label	Count	Percentage
0	8228	81.23%
1	459	4.53%
2	748	7.38%
3	694	6.85%

4 Methods

Multiple algorithms and methods were used to arrived at the data classification to predict the room occupancy for a given set of features. The models and methods used to do prediction and analyze the data to determine the evaluation were:
(1) Closed form Linear Regression, (2) Linear Discriminant Analysis (LDA), (3) K-Nearest Neighbors (k-NN), (4) Proportional Odds, (5) Principal Component Analysis and (6) Bagging Ensemble.

Closed Form Linear Regression: The term "linear" in linear regression refers to the fact that the mathematical equation used to describe this type of regression model is in the form of: This model calculates continuous values (i.e. values that increase/decrease over time) as opposed to categorical. Closed Form Linear Regression (CFLR) is a two dimensional representation of data with the features known as x and a single label known as y. This is known as closed-form because it calculates the optimal parameters in a single step, unlike in gradient ascent/descent the parameters are constantly adjusted to reach an optimal solution. The main associated equation/model for CFLR is:

$$\theta = (X^T X)^{-1} X^T y \tag{1}$$

The cost function for CFLR is the mean squared error, which measures the distance between the predicted values and the actual values from the labels; it is the following:

$$\text{Mean squared error} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2}$$

Other metrics used to measure error in regression include mean absolute error which sums the distances between the predicted and actual values:

$$\text{Mean absolute error} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3}$$

The last statistic to cover is symmetrical mean absolute percentage error (SMAPE) which is calculated as:

$$SMAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{\left(\frac{y_i+\hat{y}_i}{2}\right)} \tag{4}$$

Linear Discriminant Analysis:

Linear Discriminant Analysis (LDA) is a supervised machine learning technique that performs both **dimensionality reduction** and **classification**. It finds a linear combination of the features that best separates two or more classes. The data is projected into a lower-dimensional space in a way that

- maximizes the separability of different classes (between-class variance), and
- minimizes the variance within each class (within-class variance).

LDA is conceptually similar to Principal Component Analysis (PCA), since both project the data into a new feature space. However, unlike PCA which ignores class labels and only maximizes overall variance, LDA explicitly uses the class information to find projections that improve class separability.

A **linear discriminant** function can be written as $g(X) = Xw$, where $X \in R^{n\times d}$ is the data matrix with d features and w is the projection matrix. This function determines how the data is mapped into the lower-dimensional discriminant space.

Two-Class LDA: For the binary classification case, the steps are as follows:

- Compute the mean vector of each class:

$$\mu^0(0), \mu^{(1)} \in R^d \tag{5}$$

- Compute the covariance matrices for each class:

$$\Sigma^{(0)}, \Sigma^{(1)} \tag{6}$$

- Define the objective function, which maximizes the separation between class means while penalizing large within-class variance:

$$J(w) = (\mu^{(0)}w - \mu^{(1)}w)^2 - \lambda(w^T\Sigma^{(0)}w + w^T\Sigma^{(1)}w) \tag{7}$$

- Define the **within-class scatter** and **between-class scatter** matrices:

$$S_W = \Sigma^{(0)} + \Sigma^{(1)} \tag{8}$$

$$S_B = (\mu^{(0)} - \mu^{(1)})^T(\mu^{(0)} - \mu^{(1)}) \tag{9}$$

- Solve the generalized eigenvalue problem:

$$S_W^{-1} S_B w = \lambda w \tag{10}$$

- The optimal projection vector w is the eigenvector corresponding to the largest eigenvalue.
- Project data onto the new axis:

$$Z = Xw \tag{11}$$

- Classification can then be performed by thresholding:

$$\hat{y} = \begin{cases} 0, & z < c \\ 1, & \text{otherwise} \end{cases} \tag{12}$$

where c is a chosen threshold.

Multi-Class LDA: Since our dataset is a multiclass data set, it is important to explain how it is handled using LDA. For k classes, the procedure generalizes as follows:

1. Compute the overall mean of the dataset:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{13}$$

2. Define the within-class scatter matrix:

$$S_W = \sum_{i=1}^{k} \Sigma^{(i)} \tag{14}$$

3. Define the between-class scatter matrix:

$$S_B = \sum_{i=1}^{k} |C_i|(\mu^{(i)} - \mu)^T(\mu^{(i)} - \mu) \tag{15}$$

where $|C_i|$ is the number of observations in class i.

4. Solve the generalized eigenvalue problem:

$$S_W^{-1} S_B w = \lambda w \tag{16}$$

5. Select the top $(k-1)$ eigenvectors corresponding to the largest eigenvalues. This yields the projection matrix:

$$W \in R^{d \times (k-1)} \tag{17}$$

6. Project data into the new space:

$$Z = XW \tag{18}$$

7. Compute the mean of each class in the projected space:

$$\mu_i' = \frac{1}{|C_i|} \sum_{x \in C_i} (xW) \tag{19}$$

8. Classify each new observation by assigning it to the nearest class mean:

$$\hat{y} = \text{argmin}_i(z - \mu_i'), \quad z \in Z \tag{20}$$

Summary: Thus, LDA reduces the dimensionality of the data to at most $(k-1)$ for k classes, while preserving the information needed to discriminate between the classes. In the reduced space, classification can be performed using a nearest-centroid rule or linear discriminant boundaries.

K-Nearest Neighbors (k-NN)

The k-Nearest Neighbors algorithm is a non-parametric classification method that predicts a room occupancy label (0 = empty, 1–3 = occupied counts) by considering the majority class among the nearest training samples. Unlike regression-based models, k-NN does not explicitly learn parameters during training, but instead bases decisions directly on the stored training data.

All environmental features (temperature, humidity, light, CO_2, sound, PIR sensors) are rescaled so that they are comparable in scale for the distance calculation. Each feature is standardized by subtracting its mean and dividing by its standard deviation:

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}, \qquad j = 1, \ldots, d, \tag{21}$$

where x_{ij} is the raw measurement of feature j for observation i, and μ_j and σ_j are the mean and standard deviation of that feature in the training set. After this step, all features contribute on the same scale within the similarity measure.

For a query point z (a new set of sensor readings), the similarity to each training sample z_i is measured using the Euclidean distance

$$D(z, z_i) = \left(\sum_{j=1}^{d} (z_j - z_{ij})^2 \right)^{1/2}, \tag{22}$$

which computes the overall difference between two sensor readings. Smaller distances indicate that the query and training examples are very similar (e.g., nearly identical CO_2 concentration and PIR activity), while larger distances indicate greater dissimilarity.

The set of the k nearest neighbors is then the index set

$$\mathcal{N}_k(z) = \{i_1, \ldots, i_k\} \subset \{1, \ldots, N\}, \tag{23}$$

such that $D(z, z_{i_1}) \leq D(z, z_{i_2}) \leq \cdots \leq D(z, z_{i_k})$ are the k smallest distances.

The prediction is made by majority voting across these neighbors:

$$\hat{y}(z) = \text{argmax}_{c \in \{0,1,2,3\}} \sum_{i \in \mathcal{N}_k(z)} w_i \, \delta(y_i, c), \tag{24}$$

where $\delta(y_i, c)$ is the Kronecker delta, equal to 1 if $y_i = c$ and 0 otherwise. The weights w_i determine how strongly each neighbor contributes to the vote.

Two weighting schemes are considered. In uniform voting, each neighbor has equal weight:

$$w_i = 1. \tag{25}$$

In distance-weighted voting, closer neighbors contribute more:

$$w_i = \frac{1}{D(z, z_i) + \varepsilon}, \qquad \varepsilon > 0. \tag{26}$$

A probabilistic interpretation can also be given by normalizing the weighted votes into class probabilities:

$$\widehat{p}(c \mid z) = \frac{\sum_{i \in \mathcal{N}_k(z)} w_i \, \delta(y_i, c)}{\sum_{i \in \mathcal{N}_k(z)} w_i}, \tag{27}$$

and the predicted label is $\hat{y}(z) = \operatorname{argmax}_c \, \widehat{p}(c \mid z)$.

Finally, the neighborhood size k and the weighting scheme are treated as hyper-parameters, chosen by validation to maximize predictive performance on the held-out validation set:

$$(k^*,\ \mathrm{s}^*) = \operatorname{argmax}_{k \in \mathcal{K},\ \mathrm{s} \in \{\mathrm{u},\mathrm{d}\}} \mathrm{Score}_{\mathcal{V}}(k, \mathrm{s}), \tag{28}$$

where $s = scheme, u = uniform, d = distance$, $\mathrm{Score}_{\mathcal{V}}$ is a performance metric such as accuracy or macro-F1, computed using the true occupancy labels from the dataset.

k-NN for this dataset operates by standardizing environmental features (27), computing distances between new sensor readings and historical samples (28), selecting the k most similar historical patterns (29), and applying the weighted voting rule (30–31). The resulting model defines piecewise-constant decision boundaries in the feature space. Within this structure, the "empty" class forms a distinct cluster, while the occupied counts (1–3) appear in overlapping regions.

Proportional Odds Upon our initial observations of the dataset, the classes appeared to be ordinal in nature. For a list of values to be ordinal, there must be a set order, such as low, medium, high. Since our classes are integers 0-3, and a guess that is further away from the true class is more incorrect, we decided to incorporate the ordinal classifier model of proportional odds (also known as ordinal logistic regression).

The key detail that the model is based on is called the proportional odds assumption. This assumption is that the odds ratios between all classes is the same. This means that the logits all parallel, with a trained threshold.

The formula for predicting classes in proportional odds can be seen in eq. 29. In order to get $P(Y = j|X)$, $P(Y \leq j - 1|X)$ is subtracted from eq. 29. There are two variables being trained for this model, α and β. α is the threshold for each sigmoid curve. There is a unique value for each class in the dataset. β is a vector of coefficients. The vector is equal in length to the number of features, and it the same for each class. This is how the model follows the proportional odds assumption.

$$P(Y \leq j|X) = \frac{1}{1 + e^{-(\alpha - x^T\beta)}} \tag{29}$$

We trained the model using gradient ascent. Equations 32 and 33 show the gradient formulas for α and β. Note that s is sigmoid($alpha + X^T\beta$). s' is the derivative of s. $p_{i,j}$ is the probability of class j given X, which is equal to P($Y \leq j|X$) - P($Y \leq j-1|X$). N is the number of samples and j is the threshold class.

$$g_i = \begin{cases} -s' & \text{if } trueY = 0 \\ s'_{trueY} - s'_{trueY-1} & \text{if } 1 \leq trueY \\ s'_{trueY-1} & \text{if } trueY = K \geq 1 \end{cases} \tag{30}$$

$$eq_j = \begin{cases} 1 & \text{if } trueY = j \\ -1 - s'_{trueY-1} & \text{if } 1 \leq trueY \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

$$\frac{\partial \ell}{\partial \alpha_j} = \sum_{i=1}^{N} c_{i,j} \frac{s'_j}{p_{i,Y_i}}, \quad j = 0, \ldots, K \tag{32}$$

$$\frac{\partial \ell}{\partial \beta_f} = \sum_{i=1}^{N} \frac{g_i}{p_{i,Y_i}} x_{if}, \quad f = 1, \ldots, p \tag{33}$$

Before the gradients are multiplied by the learning rate and summed with the previous alpha and beta, they are multiplied by weights. The is one weight per class, which is set before learning begins and does not change. The formula for the weights is show in eq. 34 for a single weight. The purpose of the class weights is to lessen the impact of overrepresented classes, such as our class 0, and increase the impact of classes with little representation.

$$w_j = \frac{N}{K \cdot count(j)} \tag{34}$$

Training continues for either a maximum number of epochs or a different limit is reached. We chose to set an early escape with the difference in loss being $1e^{-10}$, as the optimal learning rate was very low ($1e^{-5}$), and we observed improvements through large numbers of epochs. A maximum number of epochs of 20000 was selected.

Bagging Ensemble

Ensemble learning attempts to utilize multiple models together to make its predictions. Bagging is one of the core ensemble algorithms. The bagging approach creates M data subsets of equal size to the training data. In order to have unique subsets, data is selected randomly with replacement. The subset is generated by randomly selecting a row in the training dataset N times, where N is the number of samples. The replacement part is due to the fact that not all of the training samples will be present in the subset, and instead will be replaced with duplicates of other rows.

After generating the subsets, a separate model is trained for each subset. When all M models are trained, it is then time to predict. Each prediction is performed by all M models, resulting in an NxM matrix. In order to convert this to a vector of the predictions, a vote is conducted. The counts for each class predicted for a given sample are summed up, and the class with the majority vote wins the prediction.

Our bagging algorithm was created to use a single base model and train M clones of that model. We ran the model for all four of our individual models. With the exception of proportional odds, all models were trained using the same setups as they used individually. The maximum epochs used for proportional odds was $2,000$ instead of the $20,000$ it had individually, to allow for training in a reasonable timeframe. Bagging was run with various subset sizes to find an ideal number of for each of the models as the base. The number of subsets trained per model can be seen in the Results section.

5 Evaluation

Data and pre-processing

The data preprocessing and conversion, as mentioned below, were accomplished or attempted. The date and time fields were proven not having any or very little impact on the overall results. It was mostly the features pertaining to the sensors that were effective in accurate prediction of room occupancy. The details of data are mentioned in the Data section above. We tested the metrics for both testing and training/validation set but we only published the training/validation set results for matrix and the confusion matrix below under Results Section.

Prior to beginning passing the training set through each of the models and testing the validation set against the model, following pre-processing was either done/attempted:

- All the String fields were converted to floats since we did not end up including any non-float/integer fields as our final feature set.
- The date and time features where ignored/dropped because these did not seem to have any impact on the outcome of the classification.
  - Attempted to convert date to **one-hot-encoded** day of the week
  - Attempted to convert time to hour of the day (24-hour scale)
- No rows were omitted/dropped from the original data set.
- We also attempted to apply SMOTE (Synthetic Minority Over-sampling Technique) for mitigating class imbalance. It should ideally have helped to generate new minority class examples, making the dataset more balanced and thereby enable models to make more balanced decisions. But this did not have a substantial impact on the accuracy. The technique was not implemented from scratch, but instead using a python `'imblearn'` library.
- A sample data set was synthetically generated with AI tool, with same set of features and similar size, so as to cross-validate the final evaluation. The observations from the data set were also consistent with what we observed with the original data set.
- The data set was shuffled and $\frac{2}{3}rd$ of data was used as training and the remaining $\frac{1}{3}rd$ as validation/testing data set.

Machine Learning

We conducted experiments on four individual models: k-Nearest Neighbor (k-NN),Linear Discriminant Analysis (LDA), Closed Form Linear Regression (CFLR) and Proportional Odds Model. k-NN achieved the best performance, with 99% accuracy and consistently high precision and recall across all classes. LDA performed nearly as well, with 98% accuracy, showing stable results across different occupancy classes. CFLR achieved a moderate accuracy of 95%, but it struggled particularly with categorical predictions. Proportional Odds Model was the weakest, with 87% accuracy, and proved to be a poor fit for sensor data. Bagging Results showed improved stability for k-NN and LDA, confirming their robustness, while the impact on CFLR and Proportional Odds was minimal. To address the class imbalance, as depicted in the data section, we observed some overlap in predictions for classes 1-3.

After testing the addition of SMOTE during preprocessing, we found the impact to the overall accuracy was marginal at best, and significantly lower at worst. Only a slight gain was observed for k-NN. LDA and Proportional Odds saw a small decrease in accuracy (2-3%), and CFLR saw a significant drop in accuracy by 13%. Due to the negative impact, we omitted SMOTE from our final pre-processing approach.

Principal Component Analysis (PCA)

To validate model performance, we conducted PCA to reduce the 19 sensor features into two principal components for visualization. Class 0 (empty room, shown in blue) formed a tight and distinct cluster, making it easy to separate from occupied states. Classes 1–3 (shown

in red, green, and yellow) exhibited significant overlap, making fine-grained separation more challenging. This visualization supports our evaluation that distinguishing empty vs. occupied is highly reliable, while differentiating between occupancy counts (1–3) is harder.

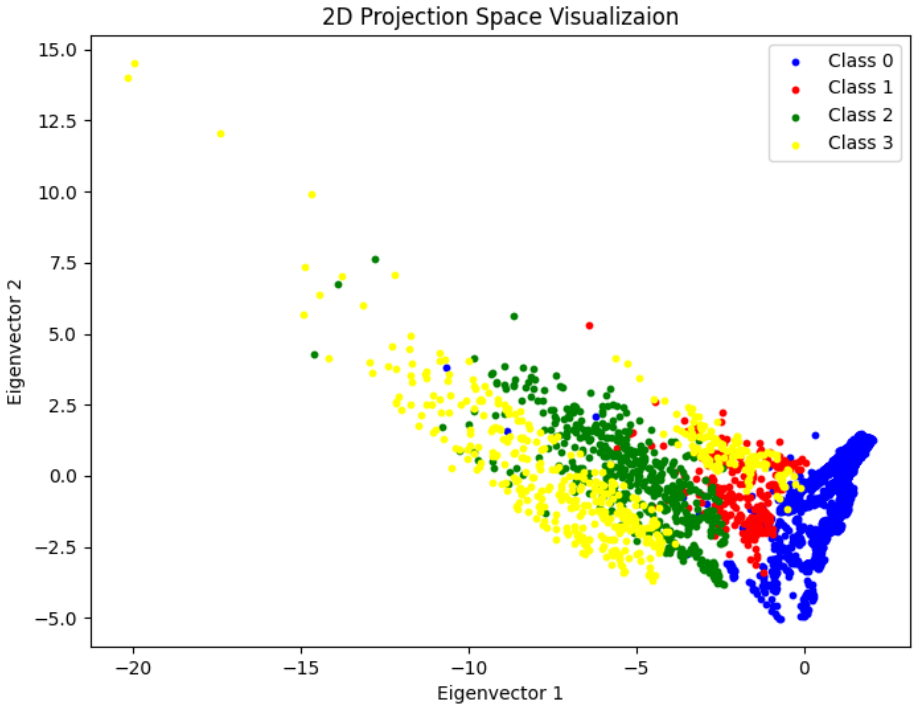

Figure 1: Principal Component Analysis

6 Results for Validation set

- Individual Models
  - Metrics

Model	Precision	Recall	f-Measure	Accuracy
Linear Discriminant Analysis	0.9562	0.9715	0.9638	98.64%
Closed Form Linear Regression	0.8071	0.8520	0.8289	94.73%
Proportional Odds	0.7163	0.8409	0.7736	87.06%
k-NN	0.9759	0.9797	0.9778	99.32%

  - Confusion Matrices
    - Linear Discriminant Analysis

Class/Label	0	1	2	3	Actual Target Count
0	2724	0	0	20	2744
1	0	163	1	0	164
2	0	0	228	23	251
3	0	0	1	216	218

    - Closed Form Linear Regression

Class/Label	0	1	2	3	Actual Target Count
0	2697	46	1	0	2744
1	3	158	3	0	164
2	0	15	193	43	251
3	0	18	49	151	218

* Proportional Odds

Class/Label	0	1	2	3	Actual Target Count
0	2425	304	15	0	2744
1	8	154	2	0	164
2	0	17	191	43	251
3	0	0	48	170	218

* k-NN

Class/Label	0	1	2	3	Actual Target Count
0	2739	1	1	3	2744
1	0	162	2	0	164
2	0	0	241	10	251
3	1	0	5	212	218

- Bagging

Model	Subset Count
Linear Discriminant Analysis	100
Closed Form Linear Regression	45
Proportional Odds	10
k-NN	55

Model	Precision	Recall	f-Measure	Accuracy
Linear Discriminant Analysis	0.9562	0.9715	0.9638	98.64%
Closed Form Linear Regression	0.8045	0.8503	0.8267	94.64%
Proportional Odds	0.7145	0.8402	0.7723	87.50%
k-NN	0.9701	0.9735	0.9718	99.14%

7 Conclusion

k-NN consistently delivered the best performance, achieving 99% accuracy with very high precision and recall. This makes it the most robust choice for occupancy estimation. LDA also performed strongly (98% accuracy), providing a reliable alternative model when lower computational cost is preferred. Proportional Odds and Linear Regression were less effective, with accuracy of 87% and 95% respectively, and showed limitations in handling categorical predictions.

The sensor features (temperature, light, sound, CO_2, PIR motion) proved to be highly effective inputs, capturing meaningful environmental patterns linked to occupancy. A PCA analysis confirmed these findings by showing a clear separation between empty rooms and occupied rooms. However, differentiating between 1, 2, and 3 occupants remained more challenging due to overlapping clusters, highlighting a natural limitation of the sensor data.

Overall, our results demonstrate that the proposed system is feasible for real-time occupancy estimation. Such a system has direct applications in Smart buildings for adaptive HVAC, lighting, and safety systems, Energy optimization, by reducing energy waste when spaces are unoccupied, Workplace analytics, providing data-driven insights into room utilization, Smart home automation, enabling more responsive and efficient living environments.

8 Future Work

Future directions could include :

- Exploring deep learning or advanced/other approaches for improved multi-class separation incorporating additional sensors (e.g., WiFi/Bluetooth signals)
- Testing the system in real-world deployment scenarios to validate scalability and reliability
- Expanding the data set for real practical use with larger groups of people and space
- Model specific pre-processing
- Exploring more pre-processing techniques to overcome over or under representation of one/few class(es)

9 References

1. Singh, A. & Chaudhari, S. (2018). Room Occupancy Estimation [Dataset]. UCI Machine Learning Repository.https://doi.org/10.24432/C5P605
2. 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 https://doi.org/10.1016/j.enbuild.2015.11.071

3. Chen, Z., Jiang, C., Xie, L. (2018). Building occupancy estimation using environmental sensors and WiFi. Energy and Buildings https://doi.org/10.1016/j.enbuild.2018.06.026

4. Mao, S., Yuan, Y., Li, Y., Wang, Z., Yao, Y., & Kang, Y. (2024). Room occupancy prediction: Exploring the power of machine learning and temporal insights https://doi.org/10.12691/ajams-12-1-1

5. ML Learning Blogs: Linear Regression Explained Linear Regression Regression Error Metrics