Replicating and Learning: An Exploration of Machine Learning-Based Occupancy Estimation Using Multivariate Sensor Nodes

Maryam CHAHKANDI
Student Number: **501276281**
Supervisor: Dr **Tamer Abdou**
Date of Submission: **Dec 7, 2023**

Ryerson University

Table of Contents

Literature Review[1]

Business Understanding

Occupancy sensors provide real-time data that can be actioned to improve processes across building spaces. The benefits of occupancy sensors include creating more practical spaces that meet people's needs, based on demand data, ensuring maintenance is efficient, identifying vacant spaces then reducing costs and waste, by controlling lighting and temperature.

A lot of research has been carried out in the literature for occupancy detection, i.e., if the room is occupied or not. Although detection alone can help in improving energy savings, estimating the precise number of occupants can make the system even more energy-efficient.

Data Understanding

The data is a publicly available dataset on the "Room Occupancy Estimation Data Set" from the UCI Machine Learning Repository[2], for estimating the precise number of occupants in a room using multiple low_cost and non-intrusive sensors like temperature, light, sound, CO2 and PIR((Passive Infrared). The sensor nodes were deployed in a room in star configuration and measurements were recorded for a period of four days.

The focus of this paper is on occupancy estimation. In this approach instead of using a single heterogeneous sensor node, multiple nodes in a (6 m x 4.6 m) room transmitting data to the sink periodically. The occupancy in the room varies between 0 and 3 people. The ground truth of the occupancy count in the room was noted manually.

Methodology

Introduction

There are quite a few papers on ML based occupancy estimation [5]–[9]. In prior studies, diverse machine learning approaches were explored for occupancy estimation. In [5], Hidden Markov Model (HMM), Artificial Neural Network (ANN), and Support Vector Machine (SVM) were applied, with HMM exhibiting the best performance at 75% accuracy. The dynamic nature of occupancy levels highlighted the need for metrics beyond accuracy, leading to the recommendation of F1 score and confusion matrix.

In [6], an ambient sensor system utilized a Radial Basis Function (RBF) neural network, but lacked cross-validation, potentially limiting effectiveness in large spaces. [9] employed sensors measuring CO2, air volume, auxiliary, and room temperature, using binned occupancy levels rather than point estimates. A similar binning approach in [8], employing a convolutional deep bidirectional long short-term memory model, achieved high accuracy.

In [7], a network with three sensor nodes utilized Extreme Learning Machines (ELM) for feature selection, accurately determining occupancy levels. However, precision suffered when exact numbers were required, although estimation was successful for up to twenty-two occupants. Those primarily explored single-sensor approaches, lacking adaptability for large spaces and precise estimates.

In contrast, this paper involves deploying multiple heterogeneous sensor nodes, transmitting data periodically to a sink, and occupancy estimation is a step beyond occupancy detection, aiming to precisely determine the number of occupants in a room (0 to 3), employs various

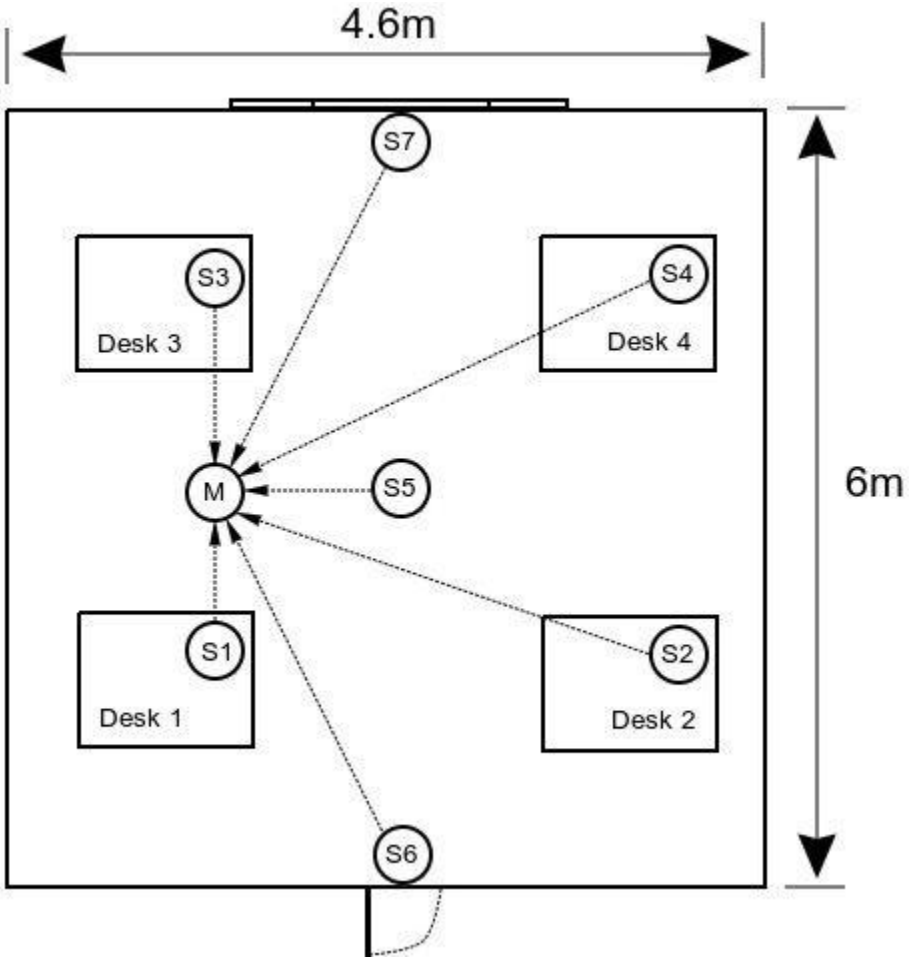
machine learning techniques, including linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), SVM (Linear and RBF kernels), and random forest (RF).

Key contributions include:

- Deployment of multiple heterogeneous sensor nodes in a room for enhanced data collection.
- Performance comparison of estimation accuracy and F1 score across various feature combinations, both homogeneous and heterogeneous.
- Discussion of data preprocessing and feature engineering techniques, handling a vast dataset, and introducing new features like the slope of CO2 to enhance performance.
- Integration of principal component analysis (PCA) alongside supervised learning techniques, exploring the performance of transformed and reduced feature sets compared to the original extensive feature set."

Experimental Setup

Fig I



In the experimental setup, a wireless sensor network (WSN) was deployed, shown in Fig I. The room contains four office desks, a window with blinds, and a glass door. The WSN, based on Zigbee, forms a star network with seven nodes—six slave nodes (S1-S6) and one master node. Various non-intrusive sensors (temperature, illumination, sound, CO2, and PIR) were employed.

Sensor nodes S1-S4 were placed at desks and equipped with low-cost sensors, while S5 focused on CO2 readings for comprehensive room analysis. Nodes S6 and S7, located on the ceiling, specifically housed PIR sensors for motion detection:

Table I

Specifications of the Sensors Used in the Experiment

Sensor	Parameter	Resolution	Accuracy
BH1750	Light	1 LUX	1.2 times
MAX4466	Sound	0.01V*	-
MH-Z14A	CO2	5ppm	±50ppm
Digital PIR	Motion	-	-

*Sound level considered in terms of voltage and not dB.

Each sensor node, utilizing a microcontroller board, sampled and transmitted data to a master node, via Zigbee radio, every 30 seconds. Temperature, light, and CO2 sensors were sampled once in this timeframe, considering their slower changes. In contrast, PIR and sound sensors required constant polling to prevent data loss. The microcontroller polled the PIR every 2.5 seconds for motion events, sending a '1' to the master node if any occurred within 30 seconds. For sound sensors, the algorithm determined the maximum peak-to-peak voltage achieved in the same timeframe. The master node, equipped with Zigbee radio, received and merged sensor data, appending it to a file along with the current timestamp.

Feature Engineering

Following tasks were done to convert raw sensor data into a usable dataset:

1. Merging timestamps for uniformity due to variations in data arrival times.
2. Deleting feature vectors with missing PIR and sound data, resulting in a more credible dataset.
3. Creating a final dataset with over 10,000 points and 16 features. A regression based method is proposed for calculating a new CO2 slope, derived from real-time CO2 values, for improved correlation with room occupancy.
4. Manual establishment of ground truth through registering the exact time and desk number for individuals entering or leaving the room.

Machine Learning Algorithms

In machine learning, both supervised and unsupervised algorithms were employed. Initially, supervised learning methods, including LDA, QDA, RF, and SVM, were utilized for multiclass classification. Gaussian distribution assumptions were made by LDA and QDA, while SVM employed a hyperplane for classification without making data assumptions. SVM used a one-vs-one scheme for multiclass problems and included a tunable penalty hyperparameter.

RF, a classifier comprising decision trees, was employed, leveraging an ensemble approach, and each tree contributed to the final outcome. The classification and regression tree (CART) algorithm was used for tree formation, considering the gini index for split quality. RF's tunable hyperparameters included the number of trees, pruning parameters, and prevention of overfitting.

Moving to unsupervised learning, PCA was applied for dimensionality reduction. PCA decomposed the dataset into orthogonal components capturing maximum variance. The reduced dataset was then evaluated using the aforementioned supervised models.

Experiments and Results

The ML algorithms discussed earlier were implemented using Scikit-learn[3]. Valuation metrics, such as accuracy, F1 score, and confusion matrix, underwent assessment through 10-fold cross-validation. Due to the time-series nature of the data, no shuffling occurred before cross-validation to prevent similar data points from the test set entering the training data. For SVM, normalization of training set features was performed to achieve zero mean and unit variance since the algorithm is scale-variant. The normalization constants from the training set were applied to scale the testing set during each iteration of the cross-validation loop.

Beyond linear SVM, the assessment also considered results with the RBF kernel to allow for non-linear classification boundaries. The penalty hyperparameter ranged from 10^(-4) to 10^4 for each feature set, and the reported metric values reflect the best results. Regarding RF, a forest with 30 trees was maintained, and metrics were calculated by averaging over 100 iterations. Given the dataset's skewness, particularly with more points corresponding to an empty room (zero occupancy), the macro F1 score, which calculates metrics for labels separately, was deemed more reliable than the micro F1 score[4].

Table II presents the accuracy and macro F1 scores for different feature combinations within our dataset, as assessed by the various ML algorithms employed.

Table II
CROSS VALIDATION ACCURACY AND F1 SCORE

Feature	Metric	LDA	QDA	SVM (Linear)	SVM (RBF)	RF
Temp{1,2,3,4}	A F1	0.840 0.479	0.862 0.590	0.866 0.554	0.895 0.730	0.869 0.657
Light{1,2,3,4}	A F1	0.973 **0.928**	0.919 **0.854**	0.973 **0.929**	0.973 **0.927**	0.972 **0.925**
Sound{1,2,3,4}	A F1	0.851 0.449	0.879 0.544	0.875 0.542	0.885 0.591	0.887 0.601
PIR{6,7}	A F1	0.869 0.474	0.869 0.474	0.870 0.466	0.870 0.460	0.870 0.460
CO2	A F1	0.809 0.383	0.808 0.409	0.812 0.286	0.812 0.314	0.763 0.329
Slope	A F1	0.852 0.387	0.831 0.394	0.870 0.462	0.870 0.510	0.876 0.564
CO2, Slope	A F1	0.891 0.556	0.867 0.590	0.890 0.592	0.888 0.635	0.873 0.559
Temp{1,2,3,4}, CO2, Slope	A F1	0.903 0.653	0.881 0.680	0.904 0.667	0.912 0.750	0.894 0.684
Temp{1,2,3,4}, CO2,Slope, Sound{1,2,3,4}	A F1	0.920 0.735	0.908 0.749	0.933 0.793	0.924 0.782	0.918 0.731
Temp{1,2,3,4}, CO2, Slope, Sound{1,2,3,4} , PIR{6,7}	A F1	0.922 0.737	0.910 0.748	**0.934** **0.793**	0.924 0.780	0.919 0.734
Temp{1,2,3,4}, CO2, Slope, Sound{1,2,3,4},PIR{6,7} ,Light{1,2,3,4}	A F1	0.980 0.946	0.957 0.911	0.982 0.948	**0.984** **0.953**	0.978 0.933

*The numbers in curly bracket denotes the Sensor ID.
**A denotes Accuracy and F1 denotes macro F1 score.

In the initial supervised learning phase, data fusion was exclusively homogeneous, as detailed in the first section of Table II. The results indicate that the proposed CO2 slope feature demonstrates promising performance, outperforming CO2 for most algorithms. Its efficacy notably improves when combined with other features. However, apart from light, other features exhibit high accuracy but poor F1 scores, prompting the exploration of additional feature combinations. The subsequent phase involved heterogeneous fusion, documented in the latter half of Table II. Feature combinations were generated by iteratively adding one sensor type at a time in a greedy manner until the complete dataset was obtained. Notably, light was considered only in the end. The best performance in the homogeneous case was observed with light sensors, achieving 97.3% accuracy and an F1 score of 0.929. This bias towards light can be explained by the common practice of individuals turning on lights upon arrival and off upon departure. However, relying solely on light sensors for occupancy estimation could lead to vulnerabilities, such as false positives when individuals leave the room with lights still on. In systems, requiring occupancy-based on light control, light cannot be considered a feature. Hence, light was initially excluded from heterogeneous fusion, despite its superior performance.

The complete dataset, inclusive of all sensors, achieves the most accurate occupancy estimation, with SVM using an RBF kernel yielding 98.4% accuracy and an F1 score of 0.953. Other algorithms exhibit similar performance, except QDA, which settles at an F1 score of 0.911. Even without light features, the complete dataset achieves a good accuracy of 93.4% and a moderately high F1 score of 0.793 with linear SVM. Confusion matrices for the best cases in both feature sets are provided in Tables III and IV.

Table III

CONFUSION MATRIX FOR LINEAR SVM CASE FOR THE COMPLETE DATASET DEVOID OF LIGHT FEATURES

	Predicted 0	Predicted 1	Predicted 2	Predicted 3
Actual 0	8117	43	41	27
Actual 1	104	336	19	0
Actual 2	65	48	502	133
Actual 3	21	7	154	512

Table IV

CONFUSION MATRIX FOR SVM WITH RBF KERNEL CASE FOR THE COMPLETE DATASET

	Predicted 0	Predicted 1	Predicted 2	Predicted 3
Actual 0	8196	1	3	28
Actual 1	0	453	6	0
Actual 2	0	0	712	36
Actual 3	10	1	67	616

SVM with an RBF kernel generally outperforms linear SVM, especially with fewer features, while their F1 scores are comparable with a higher number of features. In such cases, linear SVM may be preferable due to faster computations. The Gaussian assumption holds true for most cases, as there is comparable performance of LDA and QDA to SVM.

The highest accuracy occurs when all 16 features are considered. Given the exceptional accuracy and F1 score of the four light features, PCA was performed on the feature set without lights in an attempt to reduce dimensionality from 12 to a significantly smaller value.

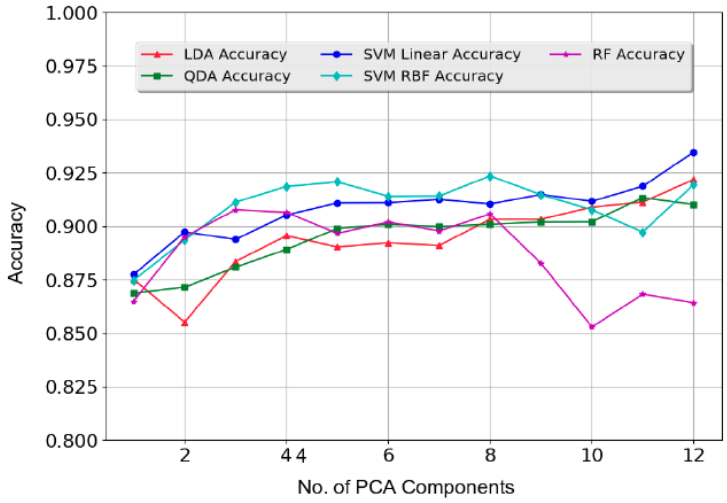


Fig II. Accuracy of different ML models with respect to the number of PCA components

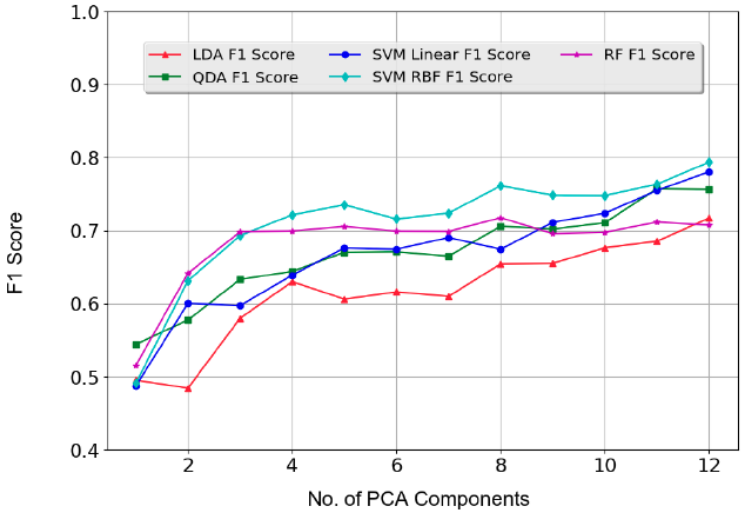


Fig III. F1 score of different ML models with respect to the number of PCA components

Fig II and Fig III show that even with as low as four components, SVM with RBF kernel gives an accuracy of around 92% and a moderate F1 score of 0.72. Linear SVM also shows a similar performance.

Conclusion and Future Work

This paper presents a deployment scheme for occupancy count estimation using multiple multivariate sensor nodes. Various methods for processing extensive data from the Wireless Sensor Network (WSN) are described. The proposed CO2 slope feature, derived through linear regression, improves accuracy and F1 metric performance. Results show a promising 98.4% accuracy and a high F1 score of 0.953, particularly with SVM using an RBF kernel. The study documents results for various feature combinations and explores dimension reduction using PCA. Even with four principal components, an accuracy of 92% and a moderate F1 score of 0.72 are achievable without considering light features. Experiments were conducted in a small room, with plans to extend the model to larger workspaces. Future work includes considering additional features and conducting real-time experiments.

Project Approach

Abstract

Occupancy sensors provide real-time data that can be actioned to improve processes across building spaces. The benefits of occupancy sensors include creating more practical spaces that meet people's needs, based on demand data, ensuring maintenance is efficient, identifying vacant spaces then reducing costs and waste, by controlling lighting and temperature.

Dataset and Objective

To replicate the paper in the Literature Review part, I will use the "Room Occupancy Estimation Data Set" dataset from the UCI Machine Learning Repository, for accurate estimating of the number of occupants in a room using Machine Learning models . The dataset includes information gathered from multiple sensor-nodes of temperature, light, sound, CO2, and PIR (Passive Infrared), in a (6 m x 4.6 m) room, transmitting data to a master node sink periodically,

every 30 seconds, that were recorded for a period of four days . The goal is to estimate the precise number of occupants in a room.

Methodology

Using machine learning classification models and utilizing Python packages like pandas, Scikit-learn[14], Seaborn[15], Matplotlib[16], Scipy[17], statistics[18], statsmodels[19], for data manipulation, machine learning, cross-validation evaluation, and visualization, I will investigate how ensemble techniques can outperform individual ML models, conduct cross-validation for performance assessment. I will employ statistical analysis to address class imbalance, identify relevant features, test hypotheses about feature importance, assess model significance, and interpret results.

Research Questions

I will try to replicate the study and provide responses to the important questions were addressed in the research on Room Occupancy Estimation:

1. **Model Selection and Evaluation:** Which machine learning model(s) were selected, and how was their performance evaluated in terms of metrics?
2. **Feature Selection:** What combinations of feature sets were explored, and how did these combinations impact the accuracy of occupancy estimation?
3. **Dimensionality Reduction:** Why was principal component analysis (PCA) applied, and what is the significance of evaluating the performance using a reduced-dimensional dataset?
4. **Dataset Characteristics:** Is the dataset time-series, and how does this impact the experimental design, in the context of not shuffling data prior to cross-validation?
5. **Normalization:** Why were training set features normalized, and how were the normalization constants utilized during cross-validation?
6. **Handling Skewed Data:** How was the issue of skewed data, especially with more points corresponding to an empty room(Imbalanced Class) addressed, and were specific considerations or techniques employed to relieve biases?
7. **Impact of Light Sensors:** How did the performance of light sensors compare to other features, and were there notable challenges or advantages associated with using light sensors for occupancy estimation?

8. **Exploratory Data Analysis (EDA) and Feature Understanding:** How can EDA conduct to uncover underlying patterns, understand relationships between features, identify potential outliers, address class imbalance, examine feature shapes, and measure central tendency, dispersion, distribution, and normality for individual characteristics and variations of the features?

Significance of the Questions:

Answering these questions is crucial for optimizing energy-efficient room occupancy systems. It provides insights into feature importance, and the impact of sensor configurations on accuracy, ensuring robust models for dynamic occupancy. Building managers, and researchers in the field will benefit from enhanced accuracy and adaptability in room occupancy estimation.

Exploratory Data Analysis

I summarized the dataset's characteristics by using a combination of statistical calculation and data visualization.

Data Story:

Categorical:

- **Ordinal**

  - **Date** YYYY/MM/DD: the date of the experiment
  - **Time**: HH:MM:SS the time of the experiment
  - **Room_Occupancy_Count**: Ground Truth(0 to 3) ** Manual establishment of ground truth through registering the exact time and desk number for individuals entering or leaving the room.

Numeric:

- **Continous**

  - **S1_Temp**: Sensor node 1 temperature(celesius)
  - **S2_Temp**: Sensor node 2 temperature(celesius)
  - **S3_Temp**: Sensor node 3 temperature(celesius)
  - **S4_Temp**: Sensor node 4 temperature(celesius)
  - **S1_Light**: Sensor node 1 light(LUX)
  - **S2_Light**: Sensor node 2 light(LUX)
  - **S3_Light**: Sensor node 3 light(LUX)
  - **S4_Light**: Sensor node 4 light(LUX)
  - **S1_Sound**: Sensor node 1 sound(voltage)
  - **S2_Sound**: Sensor node 2 sound(voltage)
  - **S3_Sound**: Sensor node 3 sound(voltage)
  - **S4_Sound**: Sensor node 4 sound(voltage)
  - **S5_CO2**: Sensor node 5 CO2
  - **S5_CO2_Slope**: An Improved Correlation with Room_Occupancy(A regression based method is proposed for calculating a new CO2 slope)
  - **S6_PIR**: digital passive infrared (PIR) sensor node 6 in the room(Binay)
  - **S7_PIR**: digital passive infrared (PIR) sensor node 7 in the room(Binay)

Univariate Statistics

I did Univariate Statistics to deal with each variable and every one type of data, to understand the relationship among the data, including shape and the spread of the data and the validity to use it for gaining insights.

There are three types of univariate statistics:

1. **General information:** (data type, count of total values, number of unique values)
2. **Range and middle:**(min, max, mean, median, mode, quartiles)
3. **Normality and spread:**(standard deviation, skewness)

Table V

index	Type	Count	N Unique	Mean	StdDev	Min	Max
Date	object	10129	7	•	•	•	•
Time	object	10129	10129	•	•	•	•
S1_Light	int64	10129	68	25.445	51.011	0.0	165.0
S2_Light	int64	10129	82	26.016	67.304	0.0	258.0
S3_Light	int64	10129	177	34.248	58.401	0.0	280.0
S4_Light	int64	10129	75	13.22	19.602	0.0	74.0
S5_CO2	int64	10129	186	460.86	199.965	345.0	1270.0
S6_PIR	int64	10129	2	0.09	0.286	0.0	1.0
S7_PIR	int64	10129	2	0.08	0.271	0.0	1.0
Room_Occ...	int64	10129	4	0.399	0.894	0.0	3.0
S1_Temp	float64	10129	24	25.454	0.351	24.94	26.38
S2_Temp	float64	10129	69	25.546	0.586	24.75	29.0
S3_Temp	float64	10129	29	25.057	0.427	24.44	26.19
S4_Temp	float64	10129	27	25.754	0.356	24.94	26.56
S1_Sound	float64	10129	231	0.168	0.317	0.06	3.88
S2_Sound	float64	10129	185	0.12	0.267	0.04	3.44
S3_Sound	float64	10129	258	0.158	0.414	0.04	3.67
S4_Sound	float64	10129	106	0.104	0.121	0.05	3.4
S5_CO2_Slope	float64	10129	1579	-0.005	1.165	-6.296	8.981

Quick insights:

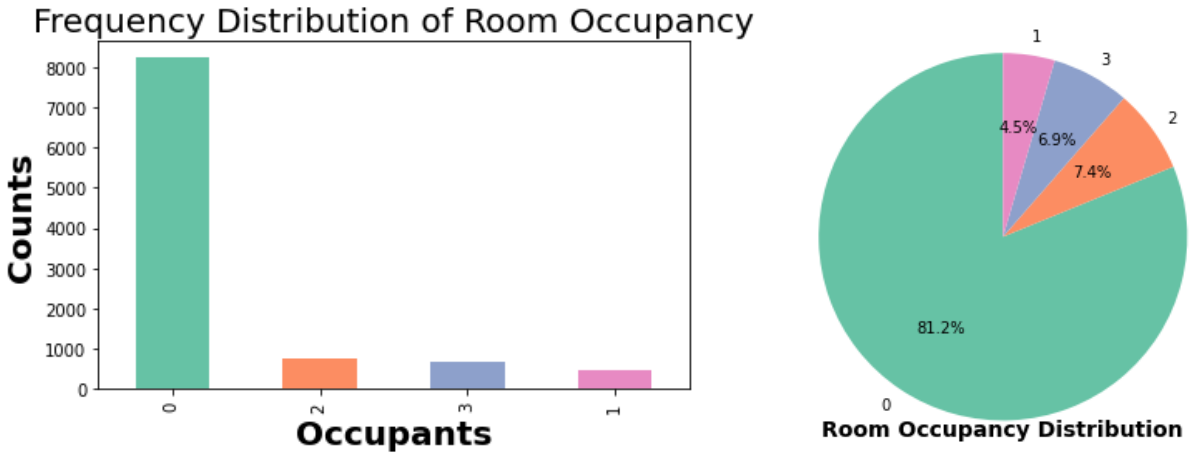
- There are **10129** instances in the dataset.
- **Dependent Variable**: Room_Occupancy_Count is to be predicted and explained with other features in the dataset, it represents a valuable of interest
- **Independent Variable**: All of other features, the one that can predict and explain Room_Occupancy_Count
- The dataset has **no missing values**; every feature has 10129 observations.
- The class **Room_Occupancy_Count** is imbalanced, with many zero values (0, 1, 2, 3).
- PIR sensors (S6_PIR, S7_PIR) are binary and mainly have **zero values**.
- Light sensors (S1_Light, S2_Light, S3_Light, S4_Light) predominantly show **zero values**.
- Temperature sensors exhibit relatively **small variations**.
- Sound sensors capture **small variations**.
- Sound levels (S1_Sound, S2_Sound, S3_Sound, S4_Sound) display **high skewness**.

Considering Univariate Statistics in (Table V) and the insights obtained, the following actions are identified for the next steps::

- ☑ Dealing with Skewed data and sensors with Zero Values (Light Sensors): When splitting the dataset for training and testing, I will use stratified sampling to maintain the distribution of different occupancy classes in both sets. I will focus on the F1-score metric with Class weights setting, to assess model performance.
- ☑ Small variations of Temperature and Sound Sensors: I might consider normalization or feature scaling
- ☑ Data Preprocessing: I might involve encoding categorical variables
- ☑ Explore Correlations: I will investigate the correlation between different sensors and occupancy
- ☑ Dealing with outliers: I will visualize distributions for variables

Visualizing Distributions Of data

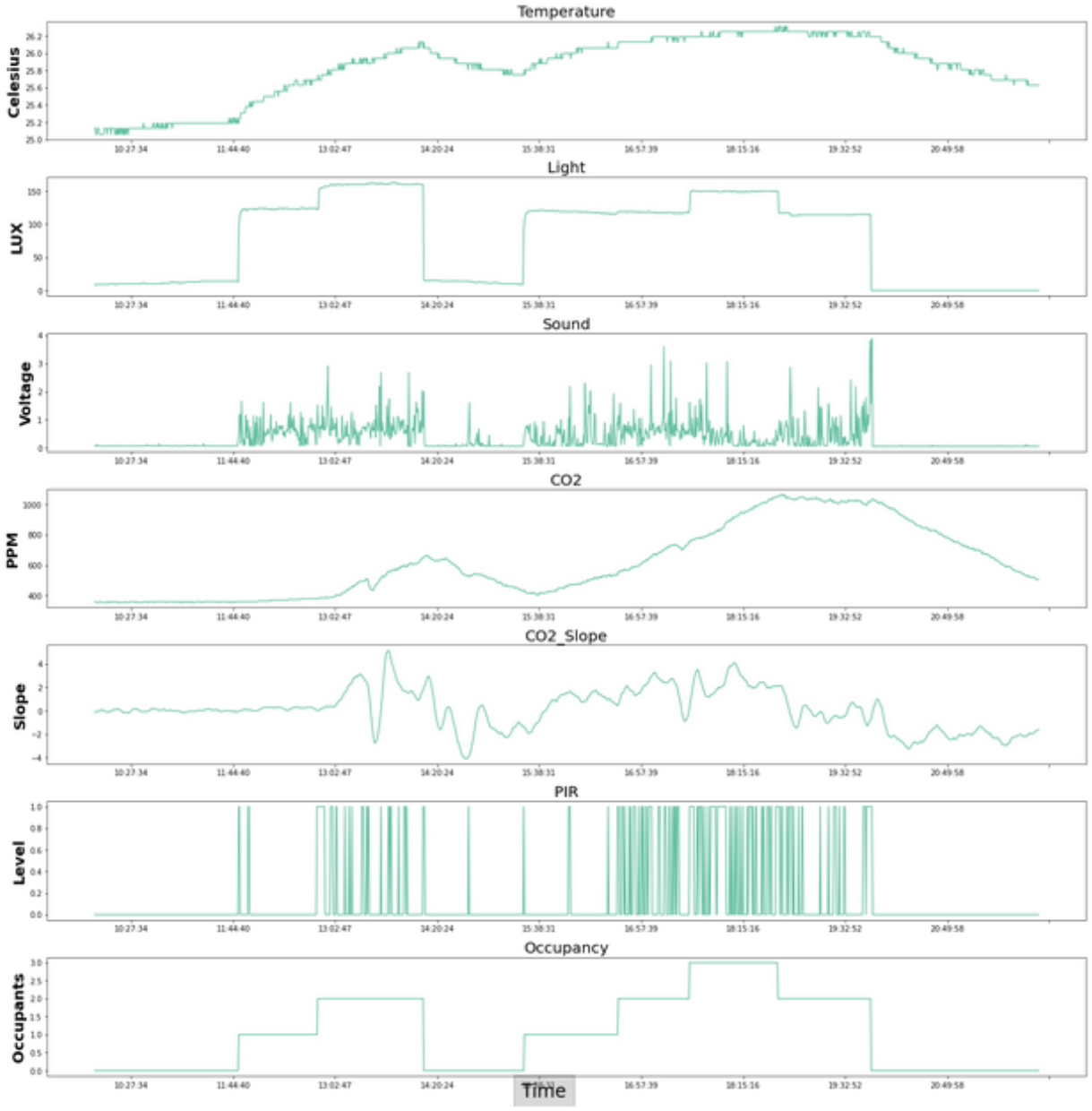
Fig 4(Frequency Distribution of Room_Occupancy_Count)



The frequency distribution of Room_Occupancy_Count (Fig4) infers that majority of Instances with Zero Occupancy. This suggests that the dataset is imbalanced, with a significant portion of instances having no occupants in the room.

It's important to be cautious when working with imbalanced datasets, as models may become biased towards the majority class.

Fig 5(Visualizing data from selective sensors and the CO2 slope for a period of 12 hours on 23/12/2017)



After looking at the time-series plots of the sensors given in Fig 5, all data seemed to give an excellent deviation to the number of occupants in the room.

Visualizing Outliers

Based on Univariate Statistics in (Table V):

- Features "S1_Light", "S2_Light", "S3_Light", "S4_Light", "S5_CO2", "S6_PIR", and "S7_PIR" exhibit characteristics of outliers, a high standard deviation relative to the mean and a notable difference between the mean and the maximum.
- The variables "S6_PIR" and "S7_PIR" are binary; so the focus will be on patterns and correlations rather than outliers, since the concept of outliers doesn't apply.

Visualizing these features would help in further assessing and deciding how to treat potential outliers. Here, I utilized both violinplot and boxplots for visualizing outliers.

Lights Outliers:

Fig 6

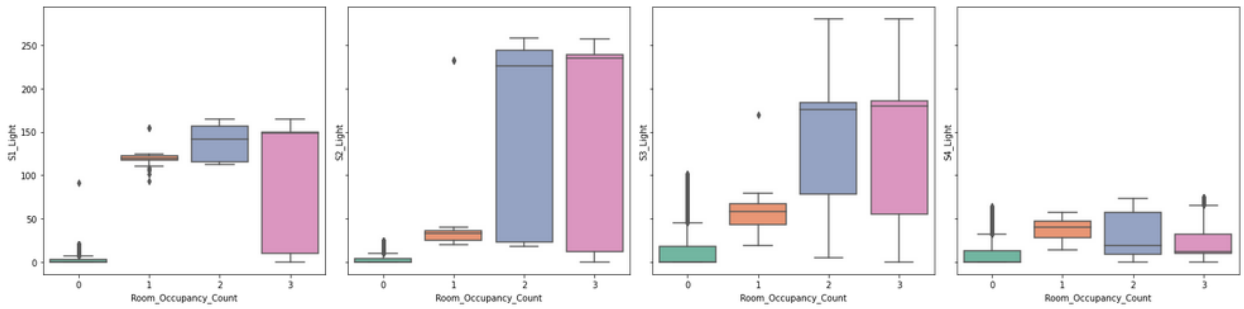


Fig 6 shows that for unoccupied rooms, light levels are consistently below 100(LUX), with noticeable outlier points, indicating lower variability, with noticeable outlier points. Occupied

rooms, however, exhibit a wider and more varied range of light levels with no significant outliers.

Fig 7

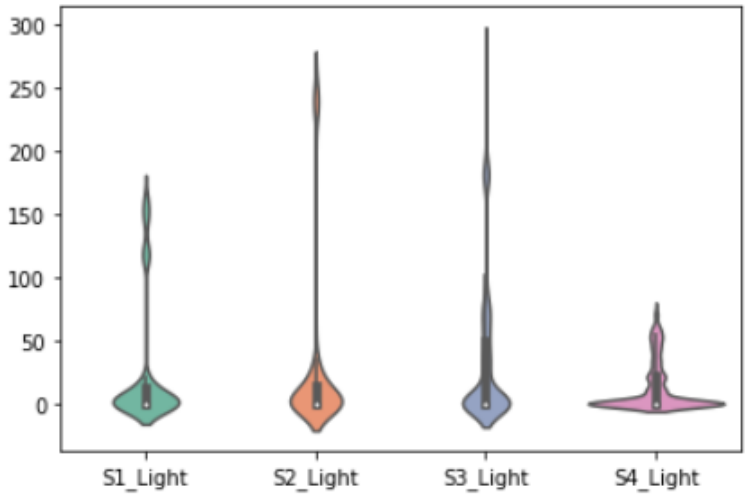


Fig 7 shows there are lots of outliers for all light sensors, there is more density around (0 or no light). Indoor light levels range between 100 and 300(LUX), those are above the 75th percentile and important. Excluding them will lead to the loss of valuable information.

CO2 Outliers:

Fig 8

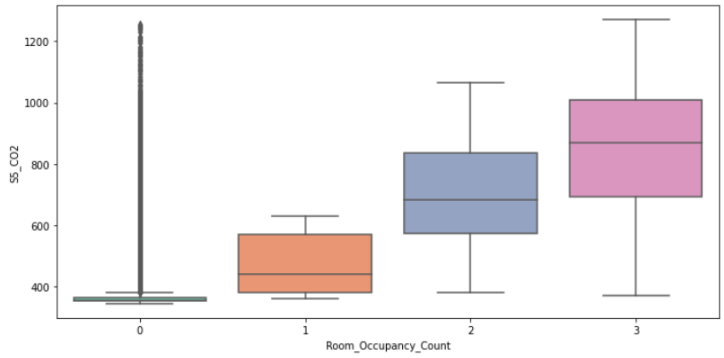


Fig 8 shows that for unoccupied rooms, CO2 levels are consistently below 400(ppm), indicating lower variability, with noticeable outlier points. Occupied rooms, however, exhibit a wider and more varied normal range of CO2 levels with no significant outliers.

Fig 9

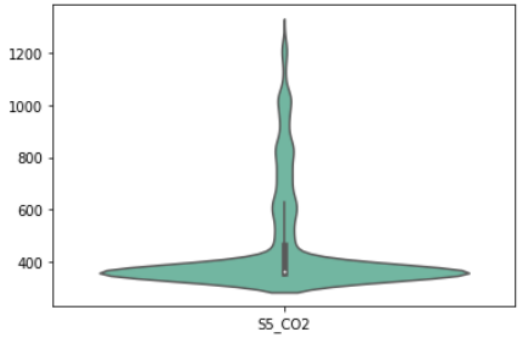
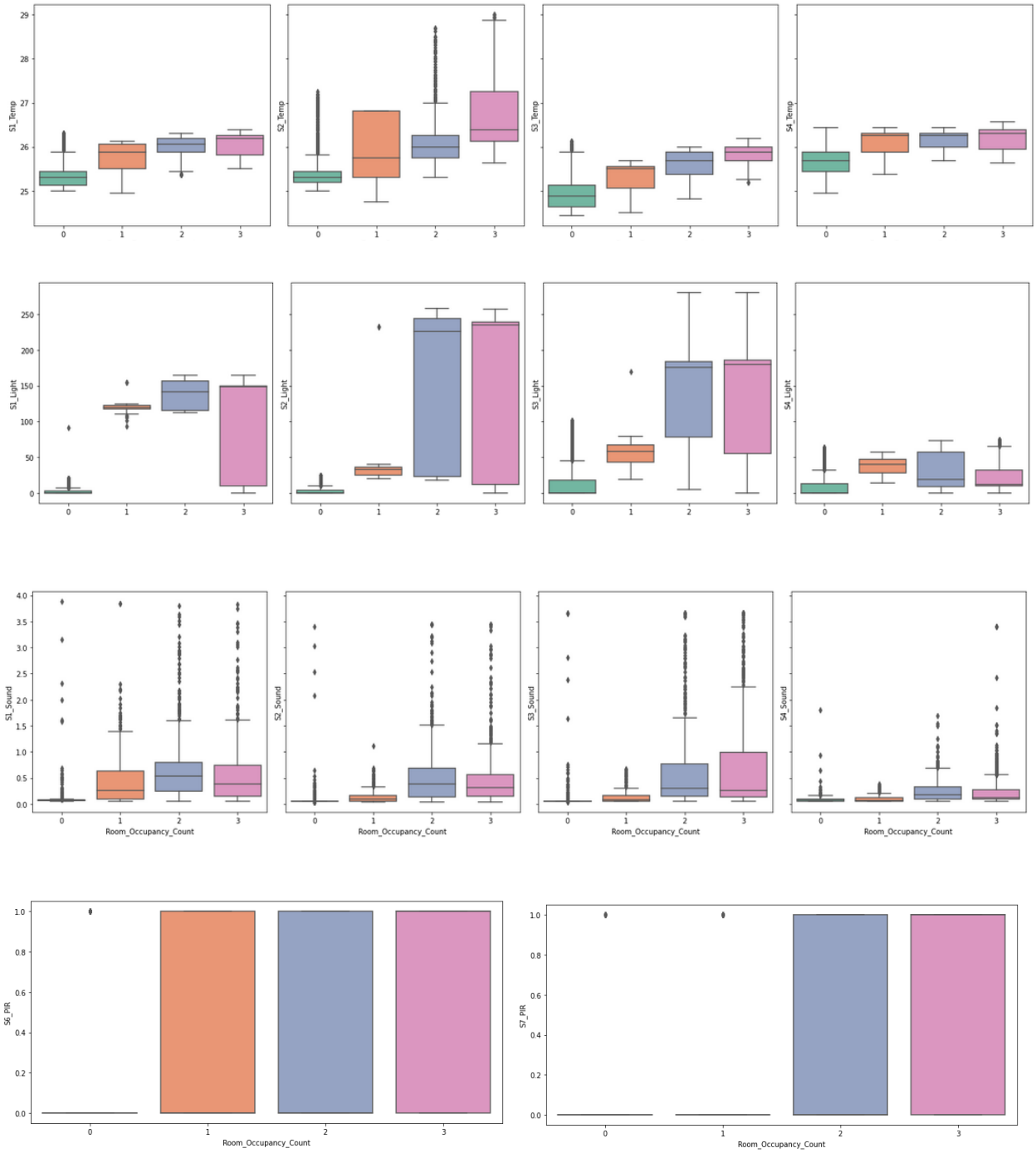


Fig 9 indicates that there are many outliers for CO2 sensor, it can be seen that the density is around (400ppm). Typical indoor CO2 level with good ventilation, range between 400 and 1100(ppm), value level greater than 1,200ppm shows a poor air quality and requires ventilation to the room. As the extreme values of the CO2 level are valuable, excluding them will lead to the loss of valuable information.

Fig 10 (Visualization of Room Occupancy vs Variables)



The Visualization of Room_Occupancy_Count vs Variables(Fig 10) infers that:

- Light, Sound and Motion sensors along with Room_Occupancy exhibit temporal patterns throughout the day, have zero values during the night time and higher values when the room is occupied.
- The S2_Temp has recorded higher values.
- The sound sensors variable show higher when the room has more than one occupant.

Conclusion

Visualizations the distribution of outliers in sensor readings "S1_Light," "S2_Light," "S3_Light," "S4_Light," "S5_CO2," "S6_PIR," and "S7_PIR," carry meaningful information, excluding such values could lead to the loss of valuable information.

Variables Correlation And Feature Selection:

Feature Selection[11] is the process that removes irrelevant and redundant features from the data set. The model, in turn, will be of reduced complexity, thus, easier to interpret. I will study this process in three parts, Filter, Wrapper, and Embedded methods.

Filter methods:

Applying feature selection methods will remove correlated features. As the target, Room_Occupancy_Count is categorical it doesn't make sense to use **Correlation Heatmap**, that is typically more relevant for data with numerical targets. On the other hand, the effectiveness of feature selection methods relies on the nature of relationships in the data. So, it is advisable to consider whether the relationships between predictors and the target class are **linear** or **non-linear** before choosing feature selection methods that align with the patterns in the data.

Linear Relation Assumption:

When assuming a **linear** relationship between two elements X, y, there are 4 **Filter methods** to choose from (Table VI). Filter methods select predictor features based on their relation to the target variable. A subset of features(X) is selected based on their relationship to the target variable(y) :

Table VI

X \ y	CONTINUOUS(NUMERICAL)	CATEGORICAL
CONTINUOUS (NUMERICAL)	**Pearson's Correlation**	**ANOVA**
CATEGORICAL	Anova	**Chi-Squared**

They measure the relevance of the features with the target via **statistical tests**. In the context of Room Occupancy estimation, X is a set of Numerical/Categorical features and y; Room_Occupancy_Count is Categorical. Therefore, **ANOVA** for NUM/CAT and **Chi_squared** for CAT/CAT have been examined.

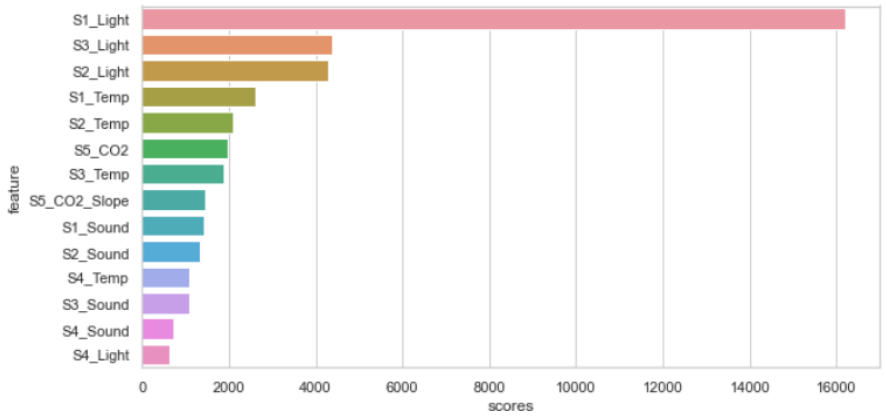
Best Numeric Features (ANOVA Test):

ANOVA Test is the score function for finding dependency of numeric features, to select the best numeric features for classification of Room_Occupancy_Count, The more the dependency the more important will be the feature.

I have used the F_classif[26] as a score function of SelectKBest[27] , for selecting the best k features based on Anova F-value returns (F-statistic scores, p values).

For classification, SelectKBest has chi2, f_classif, mutual_info_classif methods, which based on F-test, estimate the degree of linear dependency between two random variables.

Fig 11-1 (Anova–Numeric Predictors vs Room_Occupancy_Count)



	feature	p-value	scores
0	S1_Light	0.0	16188.975985
1	S3_Light	0.0	4384.458790
2	S2_Light	0.0	4293.604225
3	S1_Temp	0.0	2611.836757
4	S2_Temp	0.0	2100.372099
5	S5_CO2	0.0	1965.146696
6	S3_Temp	0.0	1876.377292
7	S5_CO2_Slope	0.0	1468.167543
8	S1_Sound	0.0	1422.084604
9	S2_Sound	0.0	1336.209925
10	S4_Temp	0.0	1104.041973
11	S3_Sound	0.0	1077.812393
12	S4_Sound	0.0	718.125256
13	S4_Light	0.0	618.804920

Fig 11-1 indicates:

● strong dependencies between the numeric features and the Room_Occupancy_Count.

● All features have a p-value of 0.0, suggesting that there are significant differences in means between groups.

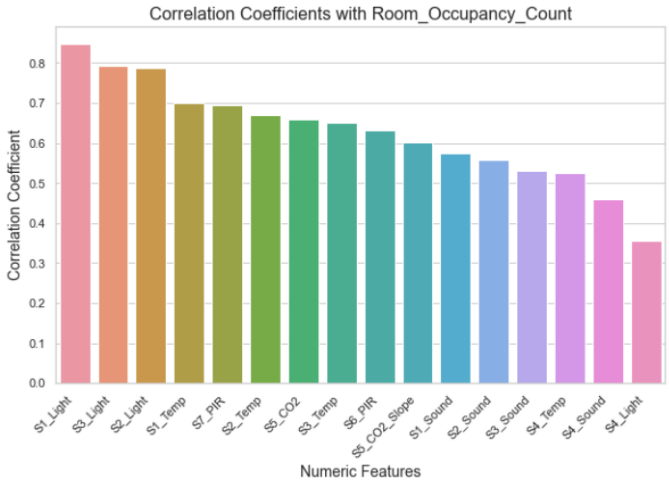
● The scores associated with each feature reflect their importance in predicting Room_Occupancy_Count,.

● S1_Light, S3_Light, and S2_Light exhibit particularly high scores, emphasizing their substantial impact on predicting room occupancy.

Pearson Correlation:

The Pearson correlation coefficient is another statistical measure that quantifies the strength and direction of a linear relationship between each numeric feature and the target variable, Room_Occupancy_Count.

Fig 11-2 (Correlation Coefficient–Numeric Predictors vs Room_Occupancy_Count)



S1_Light	0.849058
S3_Light	0.793081
S2_Light	0.788764
S1_Temp	0.700868
S7_PIR	0.695138
S2_Temp	0.671263
S5_CO2	0.660144
S3_Temp	0.652047
S6_PIR	0.633133
S5_CO2_Slope	0.601105
S1_Sound	0.573748
S2_Sound	0.557853
S3_Sound	0.531685
S4_Temp	0.526509
S4_Sound	0.460287
S4_Light	0.355715

Name: Room_Occupancy_Count, dtype: float64

Fig 11-2 shows:

- Features S1_Light, S3_Light, S2_Light, S1_Temp, S7_PIR have a strong positive correlation with Room_Occupancy_Count. As these features increase, the room occupancy count tends to increase.
- Features S2_Temp, S5_CO2, S3_Temp, S6_PIR, S5_CO2_Slope have a moderate positive correlation. An increase in these features is associated with a moderate increase in room occupancy count.

- Features S1_Sound, S2_Sound, S3_Sound, S4_Temp show a moderate to weak positive correlation.
- Features S4_Sound, S4_Light have a weak positive correlation.
- The feature S4_Light seems to have a weak positive correlation.

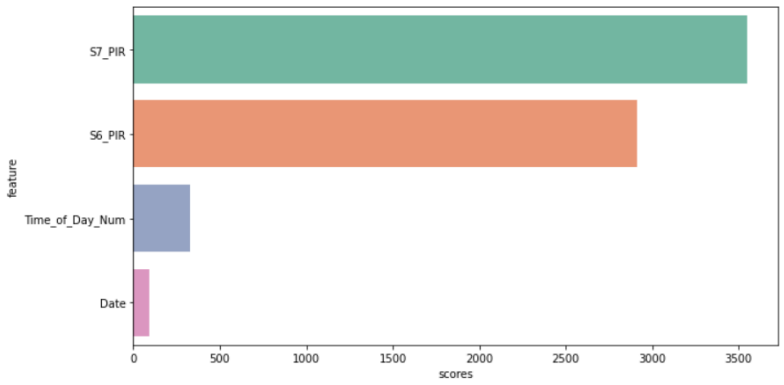
Chi-Square

Chi-Square[10] is a statistical test that is used to determine whether there is a significant difference between the observed frequencies of a categorical variable and the expected frequencies. I have used the chi2[28] as a score function of SelectKBest[27] , for selecting the best k features based on chi-squared stats returns (Chi2 statistic scores, p values). used to select the best categorical features for Room_Occupancy_Count.[1]

Fig 11-3

[1] **Time feature Encoding:**

It is necessary to transform the Time categorical values into the relevant interval ones, for further processing. (Feature Time has been encoded into new feature 'Time_of_Day_Num')



	feature	p-value	scores
0	S7_PIR	0.000000e+00	3548.368211
1	S6_PIR	0.000000e+00	2914.570104
2	Time_of_Day_Num	3.758998e-71	329.671331
3	Date	2.677279e-19	89.594301

The Fig 11-3 table indicates the significance of association between categorical features and the target variable (Room_Occupancy_Count).

Both S7_PIR and S6_PIR features have p-values of 0.0, indicating a highly significant association with Room_Occupancy_Count. The scores are also relatively high, suggesting a strong dependency.

The Time_of_Day_Num feature has a p-value of 3.76e-71, indicating an extremely significant association. The high score (329.67) suggests a strong dependency.

The Date feature has a p-value of approximately 2.68e-19, also indicating a highly significant association with Room_Occupancy_Count. The score (89.59) suggests a dependency.

In summary, based on the chi-square test results:

- S7_PIR, S6_PIR, Time_of_Day_Num, and Date are highly associated with Room_Occupancy_Count.

- The p-values close to zero indicate strong evidence against the null hypothesis of independence.
- The scores provide a measure of the strength of the dependency, with higher scores indicating stronger dependencies.

Non-Linear Relation:

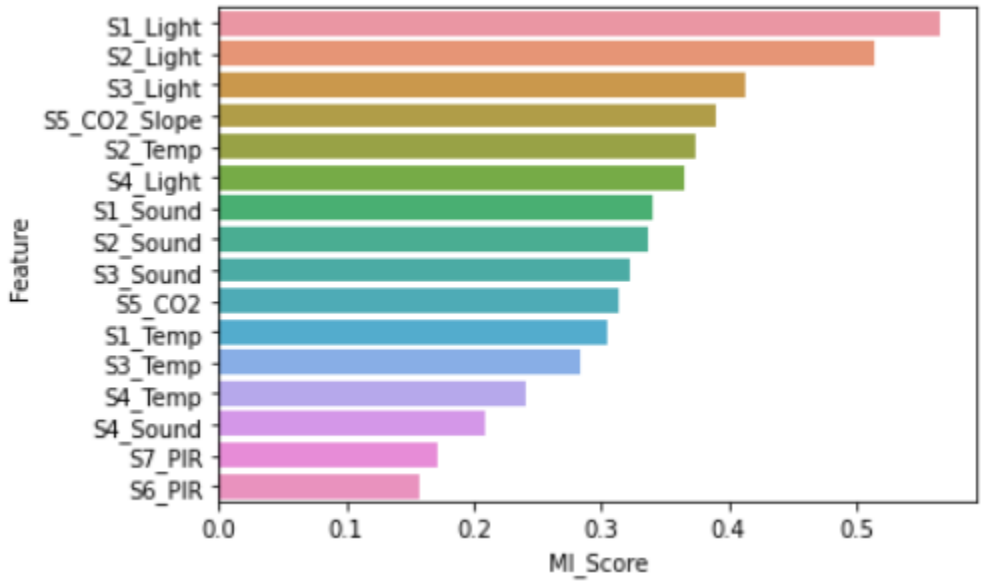
If the relationship between features and Room_Occupancy_Count is assumed to be **non-linear**, so far, the methods discussed fail to capture any relationship.

Mutual Information(MI)

To address this issue, the **Mutual Information**(MI)[12] between the features and the target variable has been considered. MI ranges from 0 (no mutual information) and 1 (perfect correlation). Sklearn offers implementation for classification tasks. MI measures the amount of information one variable provides about another and is not limited to linear relationships.

I have used the **Mutual_info_classif**[13] method that basically utilizes the mutual information. It calculates mutual information value for each of independent variables with respect to dependent variable, and selects the ones which have the most information gain. In other words, it basically measures the dependency of features with the target value. The higher score means more dependent variables. Result of this method shows in Fig 12, an illustration of sorted list of features based on the MI Score -the amount of information- of each feature provides about Room_Occupancy_Count.

FIG 12


S1_Light
S2_Light
S3_Light
S5_CO2_Slope
S2_Temp
S4_Light
S1_Sound
S2_Sound
S3_Sound
S5_CO2
S1_Temp
S3_Temp
S4_Temp
S4_Sound
S7_PIR
S6_PIR
Feature
0.0
0.1
0.2
0.3
0.4
0.5
MI_Score

Wrapper methods:

These methods measure the importance of a feature based on its usefulness while training the Machine Learning model on it. There is actually one core difference between Filter and Wrapper methods. The wrapper measures the importance of a subset of features by actually training a model on it.

Embedded/Hybrid:

These methods are a combination of Filter and Wrapper methods; first, use Filter methods to eliminate redundant features and then, Wrapper methods to select a useful subset.

Conclusion:

It all comes down to data; to choose the approach.

I have excluded **Embedded/Hybrid** methods following these reasons:

- The number of features in the dataset is manageable, making the need for Embedded/Hybrid methods less important. They are more beneficial when dealing with a large feature set.
- An Embedded/Hybrid technique involves additional computational complexity and resources.
- The research objective is well-addressed by the choice of wrapper methods.

I have followed the paper's **wrapper** approach of how a reduced feature set compares to the original larger feature set. Initially, models will learn on whole features, next, in each step, one homogeneous sensor will be excluded in the feature set.

Hints:

Feature Selection can lead to better models, which achieve higher performance and are more interpretable. It also takes time; and it is better to consider not investing neither the time, nor the effort. It must always keep in mind three things:

1. Feature Selection will lose information since dropping features and

2. Even if trying all the techniques, Feature Selection can be that no major improvement is seen on the model's performance.

3. Filter methods are computationally less expensive.

Statistical Tests:

Using One-Way Anova I am about to discover if there is a difference in average, across all values in four groups of temperature/light/sound variables.

One-way anova_(Comparing Three+ Groups) hypothesis test will determine if there are differences between the means of several classes.

The Null Hypothesis (H0) is:

- There is no significant difference in the means of the temperature/light/sound variables(four groups)

And Alternative Hypothesis (Ha) is:

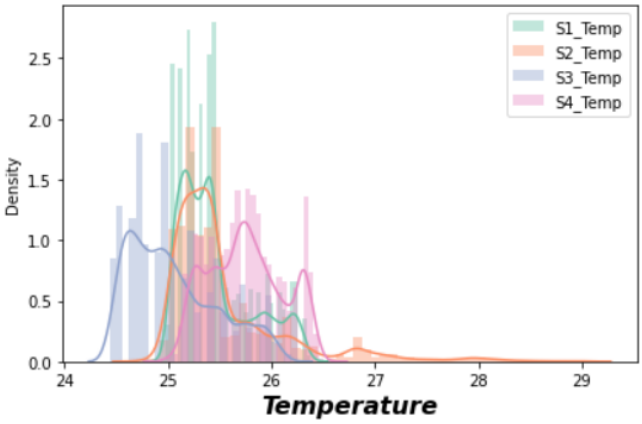
- At least one of the temperature/light/sound variables(four groups) has a different mean

Temperature Sensors

After plotting the density of (S1_Temp, S2_Temp, S3_Temp, S4_Temp), I conducted an ANOVA test, the result indicates f-value with a p-value of 0.0, so the null hypothesis will be rejected, and suggests that there is strong evidence to conclude that there is a significant difference in the means of the (S1_Temp, S2_Temp, S3_Temp, S4_Temp). At least one of the variables has a different average value compared to the others.

The pairwise Tukey's Honestly Significant Difference(HSD)[20] test was used to identify which groups are significantly different from each other.

Fig 13



Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
S1_Temp	S2_Temp	0.092	0.0	0.0761	0.108	True
S1_Temp	S3_Temp	-0.3974	0.0	-0.4133	-0.3815	True
S1_Temp	S4_Temp	0.3001	0.0	0.2842	0.316	True
S2_Temp	S3_Temp	-0.4894	0.0	-0.5053	-0.4735	True
S2_Temp	S4_Temp	0.2081	0.0	0.1922	0.224	True
S3_Temp	S4_Temp	0.6975	0.0	0.6816	0.7134	True

f-value : 4458.430427883466
p-value : 0.0

Fig 13 Table indicates that:

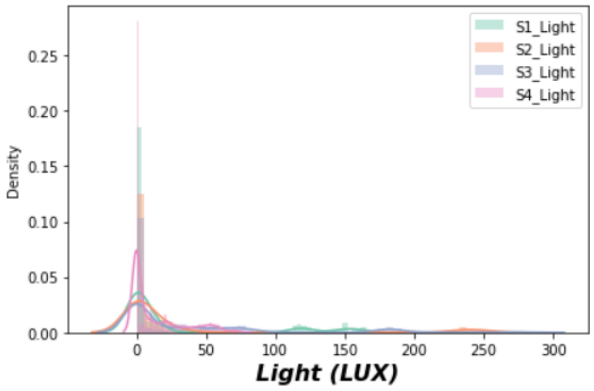
- Some sensors have higher or lower temperatures.
- The "meandiff" column provides the average difference in values between the compared groups. For example, a positive value for a pair (S1_Temp vs. S2_Temp) indicates that the mean temperature of S1_Temp is higher than S2_Temp
- There are significant differences in means between all pairs of temperature sensors.
- It can infer the direction of differences from the sign of the mean differences. For instance, if S1_Temp vs. S2_Temp has a positive mean difference, it suggests that S1_Temp tends to have higher values than S2_Temp.

Light Sensors

After plotting the density of (S1_Light, S2_Light, S3_Light, S4_Light), I conducted an ANOVA test, the result indicates f-value with a extremely low p-value, so the null hypothesis will be rejected, and suggests that there is strong evidence to conclude that there is a significant difference in the means of the (S1_Light, S2_Light, S3_Light, S4_Light). At least one of the variables has a different average value compared to the others.

The pairwise Tukey's Honestly Significant Difference (HSD) test was used to identify which groups are significantly different from each other.

Fig 14



Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
S1_Light	S2_Light	0.5712	0.8646	-1.3156	2.4581	False
S1_Light	S3_Light	8.8034	0.0	6.9166	10.6903	True
S1_Light	S4_Light	-12.2248	0.0	-14.1117	-10.3379	True
S2_Light	S3_Light	8.2322	0.0	6.3454	10.1191	True
S2_Light	S4_Light	-12.796	0.0	-14.6829	-10.9092	True
S3_Light	S4_Light	-21.0282	0.0	-22.9151	-19.1414	True

f-value : 278.39240105576897
p-value : 7.023925657885299e-179

Fig 14 Table indicates that:

- Some Light sensors have higher or lower levels

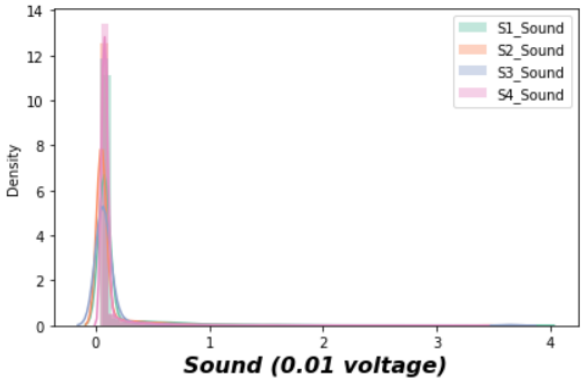
- The adjusted p-values for all pairwise comparisons, except for S1_Light vs. S2_Light, are reported as 0.0, indicating high statistical significance. This implies that there are significant differences in means between the majority of pairs of light sensors.
- The "meandiff" column provides the average difference in values between the compared groups. For example, a positive value for a pair (S1_Light vs. S3_Light) indicates that the mean light level for S1_Light is higher than S3_Light.
- Since the majority of entries in the "reject" column are True, it will be concluded that there are significant differences in means between most pairs of light sensors.
- It can infer the direction of differences from the sign of the mean differences. For instance, if S1_Light vs. S3_Light has a positive mean difference, it suggests that S1_Light tends to have higher light levels than S3_Light.

Sound Sensors

After plotting the density of (S1_Sound, S2_Sound, S3_Sound, S4_Sound), I conducted an ANOVA test, the result indicates f-value with a p-value of 0.0, so the null hypothesis will be rejected, and suggests that there is strong evidence to conclude that there is a significant difference in the means of the (S1_Sound, S2_Sound, S3_Sound, S4_Sound). At least one of the variables has a different average value compared to the others.

The pairwise Tukey's Honestly Significant Difference (HSD) test was used to identify which groups are significantly different from each other.

Fig 15



Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
S1_Sound	S2_Sound	-0.0481	0.0	-0.0589	-0.0373	True
S1_Sound	S3_Sound	-0.0101	0.0778	-0.0208	0.0007	False
S1_Sound	S4_Sound	-0.0643	0.0	-0.0751	-0.0536	True
S2_Sound	S3_Sound	0.0381	0.0	0.0273	0.0488	True
S2_Sound	S4_Sound	-0.0162	0.0006	-0.027	-0.0054	True
S3_Sound	S4_Sound	-0.0543	0.0	-0.0651	-0.0435	True

f-value : 106.04723802697454
p-value : 2.1835783091369963e-68

Fig 15 Table indicates that:

- Some Sounds sensors have higher or lower levels

- The adjusted p-values for all pairwise comparisons, except for S1_Sound vs. S3_Sound, are reported as 0.0, indicating high statistical significance. This implies that there are significant differences in means between the majority of pairs of sound sensors.
- The "meandiff" column provides the average difference in values between the compared groups. For example, a negative value for a pair (S1_Sound vs. S2_Sound) indicates that the mean sound level for S1_Sound is lower than S2_Sound.
- The "reject" column indicates whether to reject the null hypothesis of equal means. Since the majority of entries in the "reject" column are True, there are significant differences in means between most pairs of sound sensors.
- It can infer the direction of differences from the sign of the mean differences. For instance, if S1_Sound vs. S2_Sound has a negative mean difference, it suggests that S1_Sound tends to have lower sound levels than S2_Sound.

Bivariate Statistics

It deals with relationships between two variables, and how the features affect some labels.

There are three types of **Bivariate statistics**:

1. **(NUM/NUM):** Numeric to Numeric relationship
2. **(CAT/NUM):** Categorical to Numeric relationship
3. **(CAT/CAT):** Categorical to Categorical relationship

Bivariate CAT/NUM

Here, I examine how the occupancy impacts sensors values, and discover if there's a significant difference between values of all numeric features across all Room_Occupancy_Count (0 to 3). After plotting the distribution of values of all numeric features across all Room_Occupancy_Count (0 to 3), I used an ANOVA Test to examine how important features are for predicting or understanding room occupancy.

- **(CAT/NUM):** Room_Occupancy_Count **VS** (S1_Temp , S2_Temp , S3_Temp , S4_Temp , S1_Light , S2_Light , S3_Light , S4_Light , S1_Sound , S2_Sound , S3_Sound , S4_Sound , S5_CO2 , S5_CO2_Slope)

Null Hypothesis (H0):

- The means of Room_Occupancy_Count are equal across different levels of the specified features

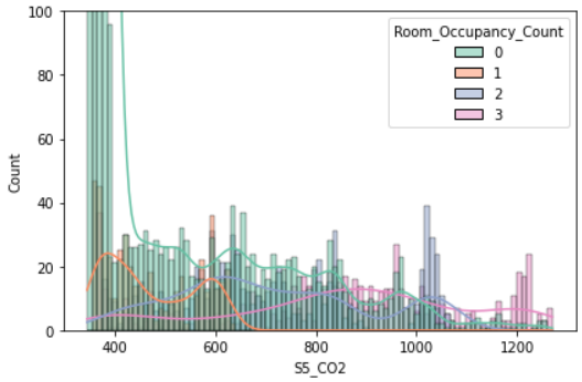
Alternative Hypothesis (Ha):

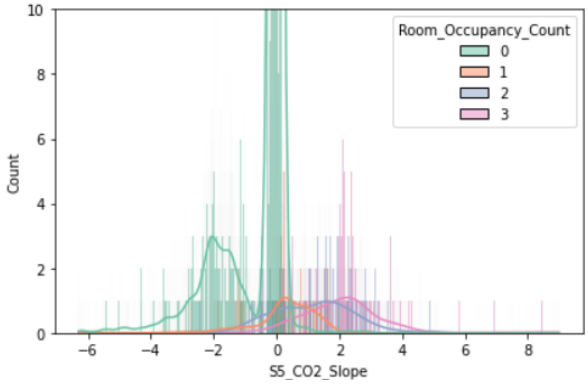
- There is a significant difference in means

One-Way ANOVA Test

(S5_CO2, S5_CO2_Slope) vs Room_Occupancy_Count

Fig 16



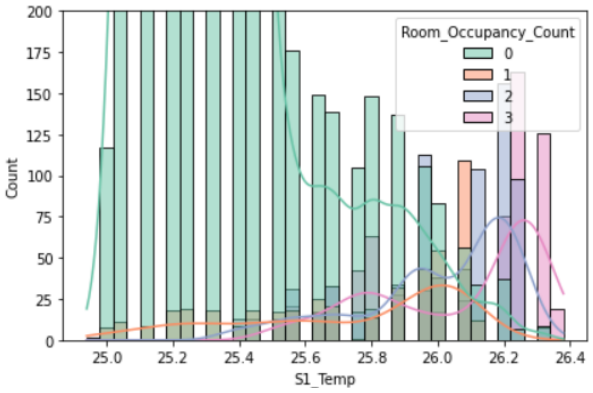


(statistic=2689.7587234168395, p value=0.0)

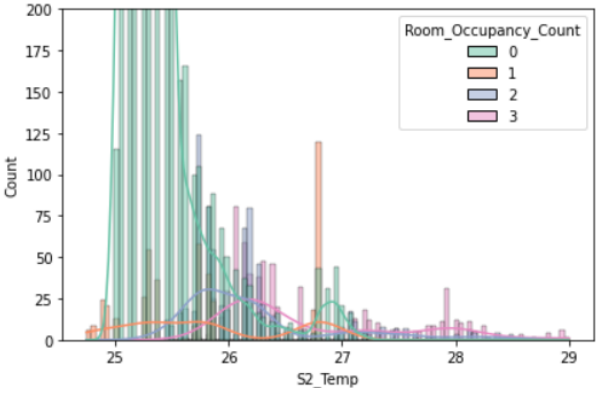
(statistic=1927.1011299664176, p value=0.0)

(S1_Temp, S2_Temp, S3_Temp, S4_Temp) vs Room_Occupancy_Count

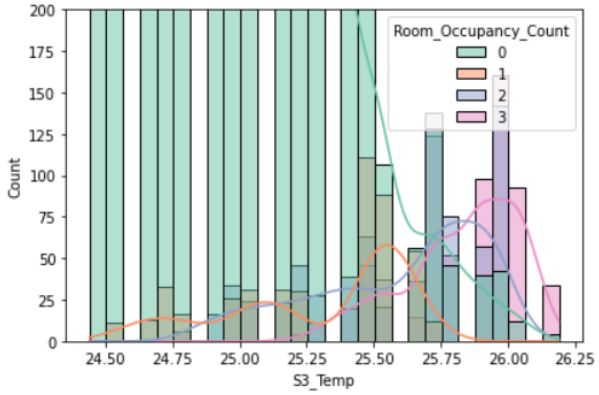
Fig 17



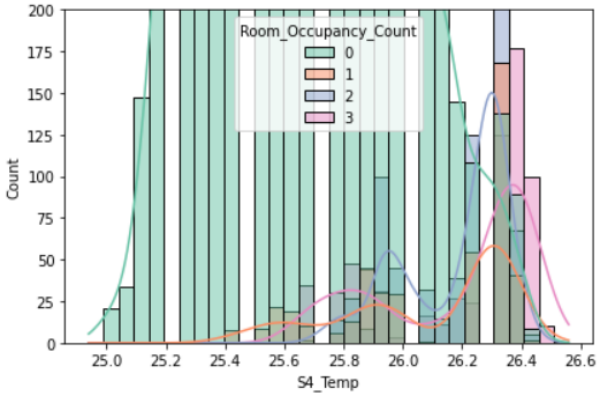
(statistic=3515.813553612012, p value=0.0)



(statistic=2774.965502971935, p value=0.0)



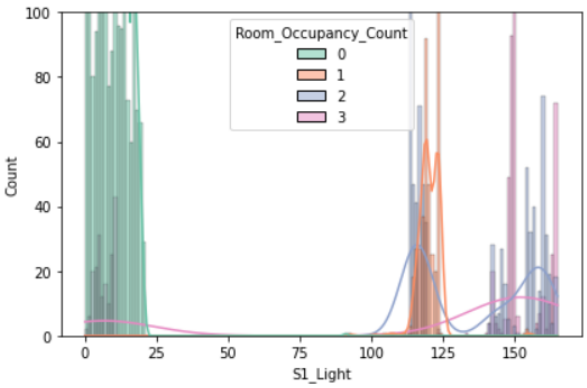
(statistic=2516.2398373741366, p value=0.0)



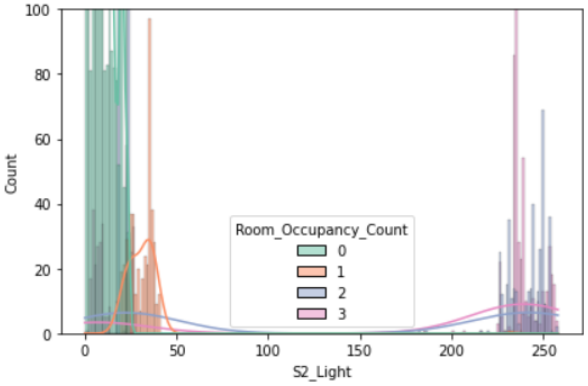
(statistic=1485.6358862104928, p value=0.0)

(S1_Light, S2_Light, S3_Light, S4_Light) vs Room_Occupancy_Count

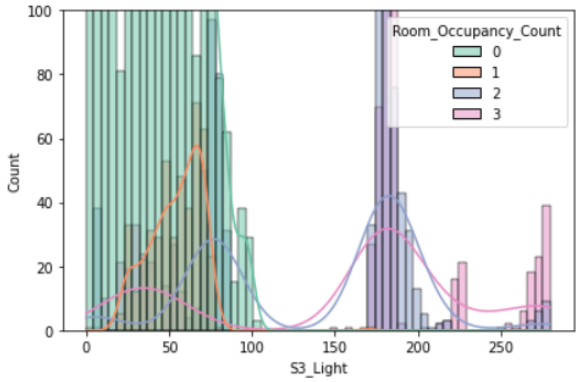
Fig 18



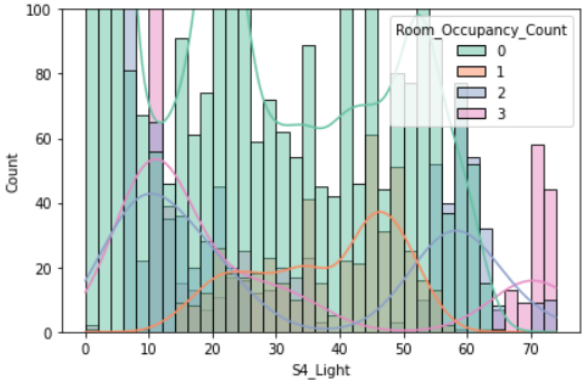
(statistic=22393.540766352362, p value=0.0)



(statistic=5862.045970873254, p value=0.0)



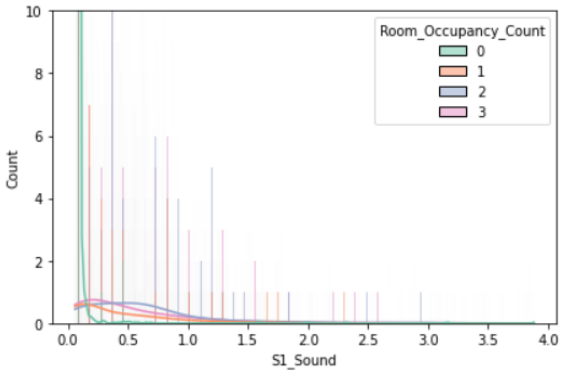
(statistic=6092.5182166646455, p value=0.0)



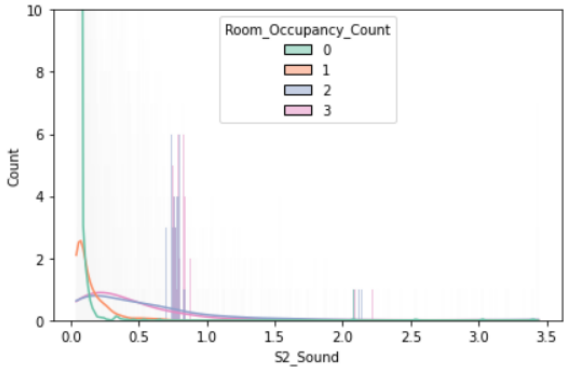
(statistic=785.911759762542, p value=0.0)

(S1_Sound, S2_Sound, S3_Sound, S4_Sound) vs Room_Occupancy_Count

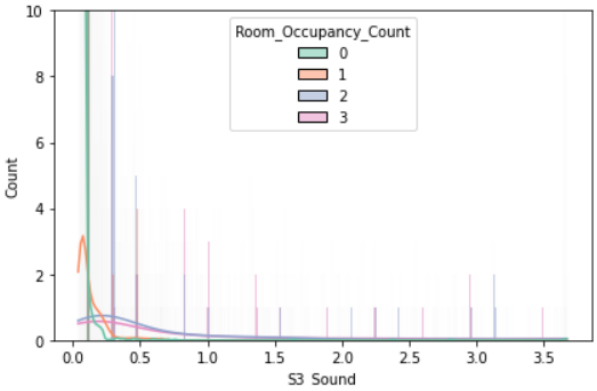
Fig 19



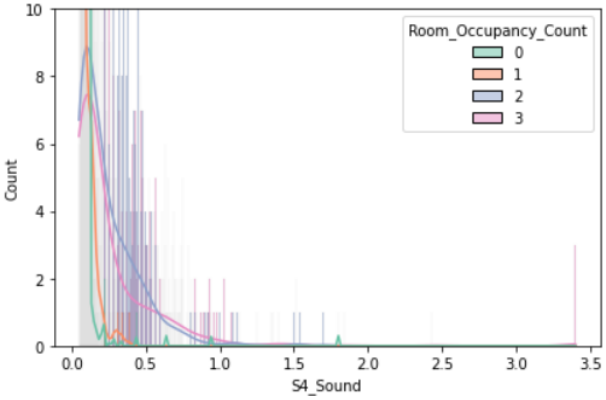
(statistic=1987.3822249923078, p value=0.0)



(statistic=1762.2562608822018, p value=0.0)



(statistic=1413.4851682771603, p value=0.0)



(statistic=988.8313307678422, p value=0.0)

The statistics from Anova Test Result in Fig 16 - Fig 17 - Fig 18 - Fig 19 show:

- A p-value of 0.0 indicates that the null hypothesis should be rejected

- There is strong evidence that there is a significant difference in the means of the all features levels across different room occupancy counts

- This implies that these sensor variables can be important features for predicting or understanding room occupancy.

Bivariate CAT/CAT

Here, I explored the dependency of categorical features to the label Room_Occupancy_Count.

- **(CAT/CAT):** Room_Occupancy_Count VS (Date , Time, S6_PIR, S7_PIR)

Time feature Encoding

It is necessary to transform the Time categorical values into the relevant interval ones. Taking into account how many times, time value is present in relation with a range of time of the day, a new feature created and called “Time_Of_Day”.

The time ranges were divided into the following categories:

- Midnight: 00:00:01 to 07:00:00
- Morning: 07:00:01 to 12:00:00
- Afternoon: 12:00:01 to 17:00:00
- Evening: 17:00:01 to 21:00:00
- Night: 21:00:01 to 23:59:59

Chi-Square test:

For this I used this statistical test to determine the association between two categorical variables. It is based on the difference between the observed frequencies of Room_Occupancy_Count and the expected frequencies. It returns a probability, zero shows complete dependency and one shows complete independency.

The Null Hypothesis (H0) is:

- There is no significant association between Room_Occupancy_Count and each of (Date , Time_Of_Day, S6_PIR, S7_PIR), they are independent, or The distribution of target class is independent of them.

The Alternative Hypothesis (Ha) is:

- There is a significant association between Room_Occupancy_Count and each of (Date , Time, S6_PIR, S7_PIR), they are dependent, or The distribution of target class is dependent on them.

After testing Chi2 on all categorical features, the result shown in Table VII:

Table VII

feature	chi2 result
'Date'	('p value : 0.0', 'Date is correlated to Room_Occupancy_Count')
'Time_Of_Day'	('p value : 0.0', 'Time_Of_Day is correlated to Room_Occupancy_Count')
'S6_PIR'	('p value : 0.0', 'S6_PIR is correlated to Room_Occupancy_Count')
'S7_PIR'	('p value : 0.0', 'S7_PIR is correlated to Room_Occupancy_Count')

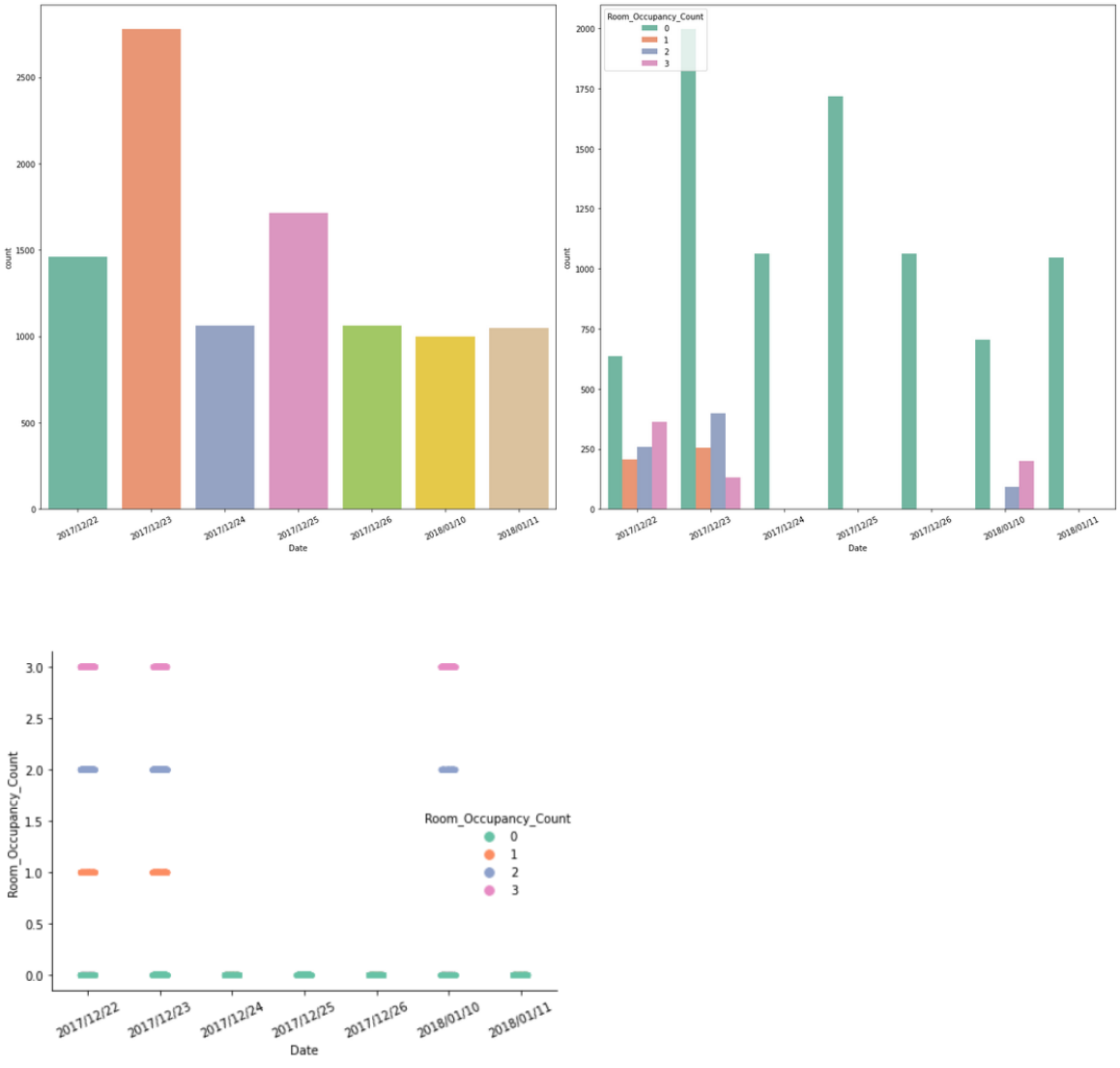
The results implies that:

- A p-value of 0.0 for all Categorical features, indicates that the null hypothesis should be rejected
- The distribution of Room_Occupancy_Count is dependent on Date , Time_Of_Day, S6_PIR, S7_PIR
- These sensor variables can be important features for predicting or understanding room occupancy

Visualization Of Categorical data

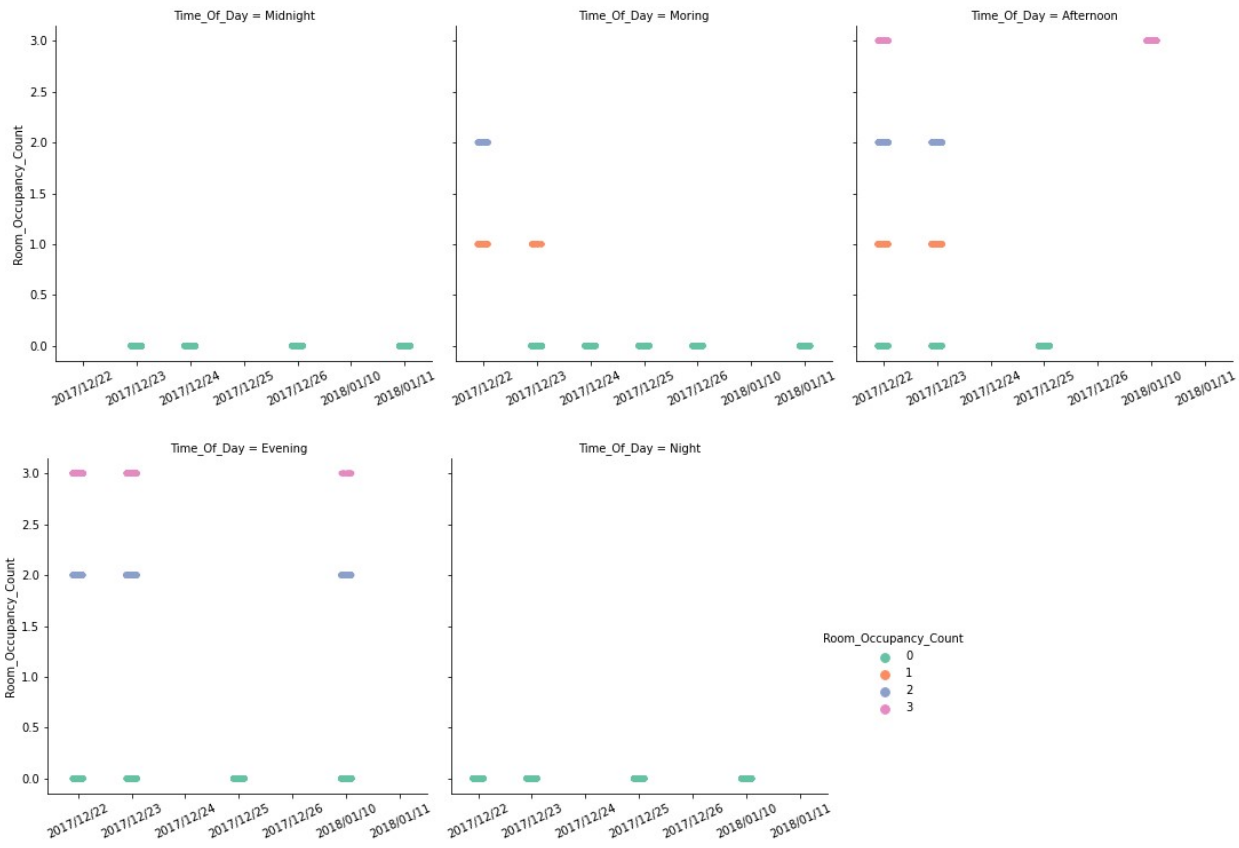
Date vs Room_Occupancy_Count

Fig 20



The Room_Occupancy_Count on the following three dates —"2017/12/23", "2017/12/22" and "2018/01/10" — indicates an occupancy of more than 1.

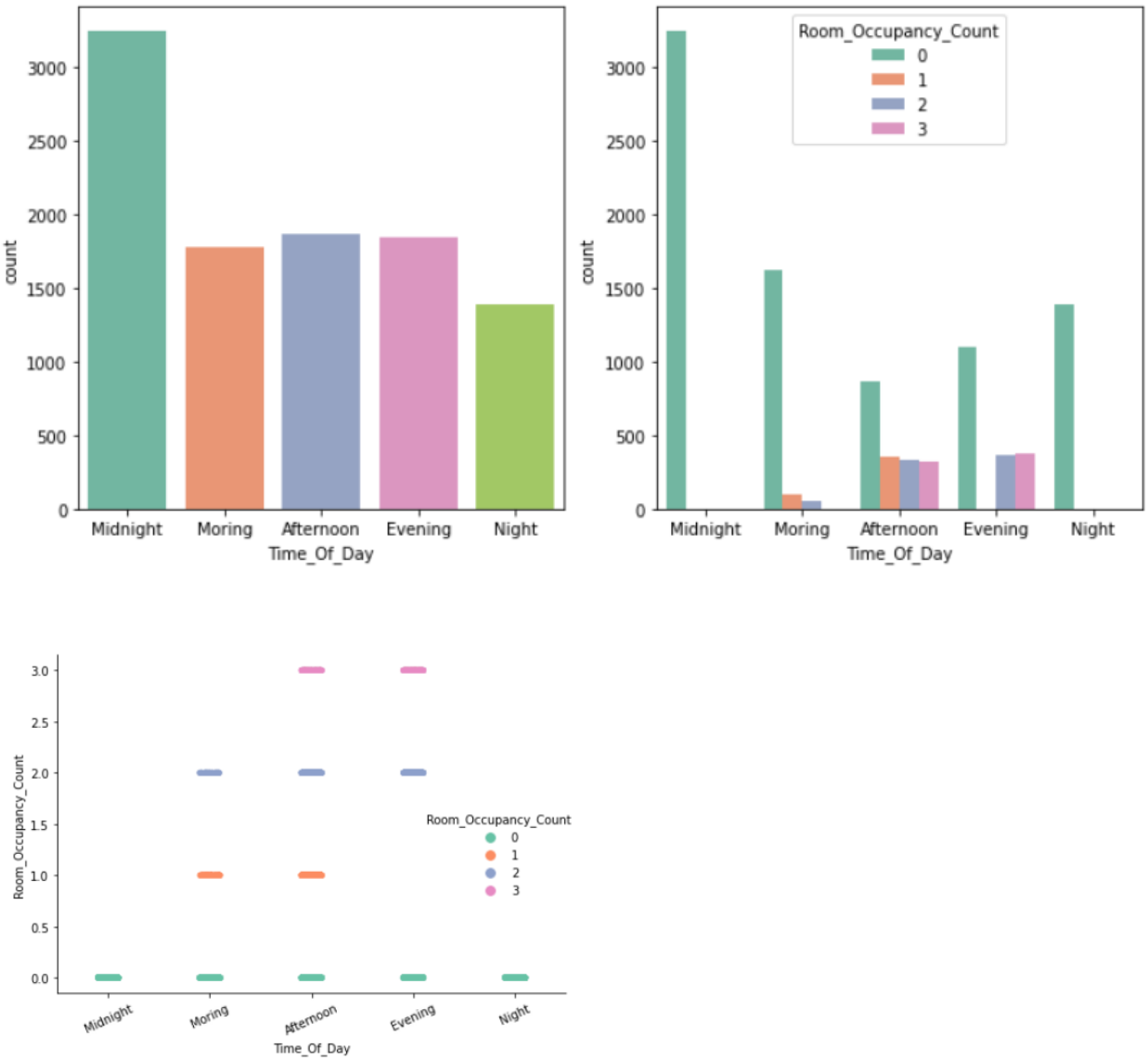
Fig 21



The Room_Occupancy_Count on the following three ‘Time_Of_Day’s —"Morning", "Afternoon", and "Evening" — indicates an occupancy of more than 1.

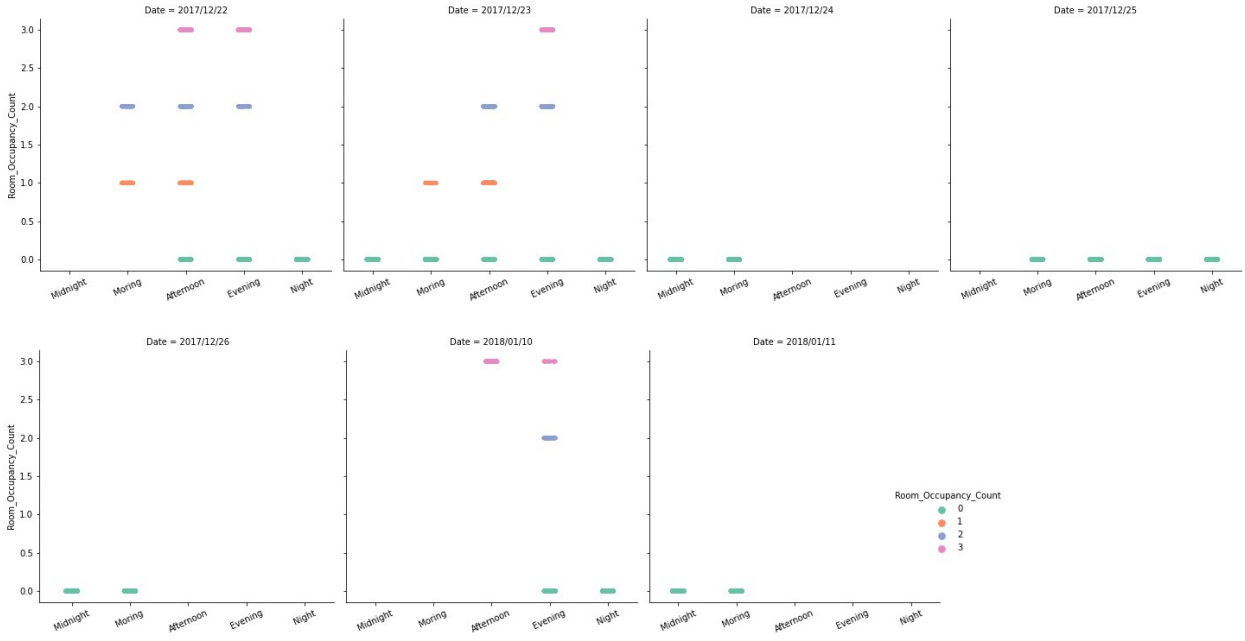
Time vs Room_Occupancy_Count

Fig 22



The Room_Occupancy_Count on the following three 'Time_Of_Day's —"Morning", "Afternoon", and "Evening" — indicates an occupancy of more than 1.

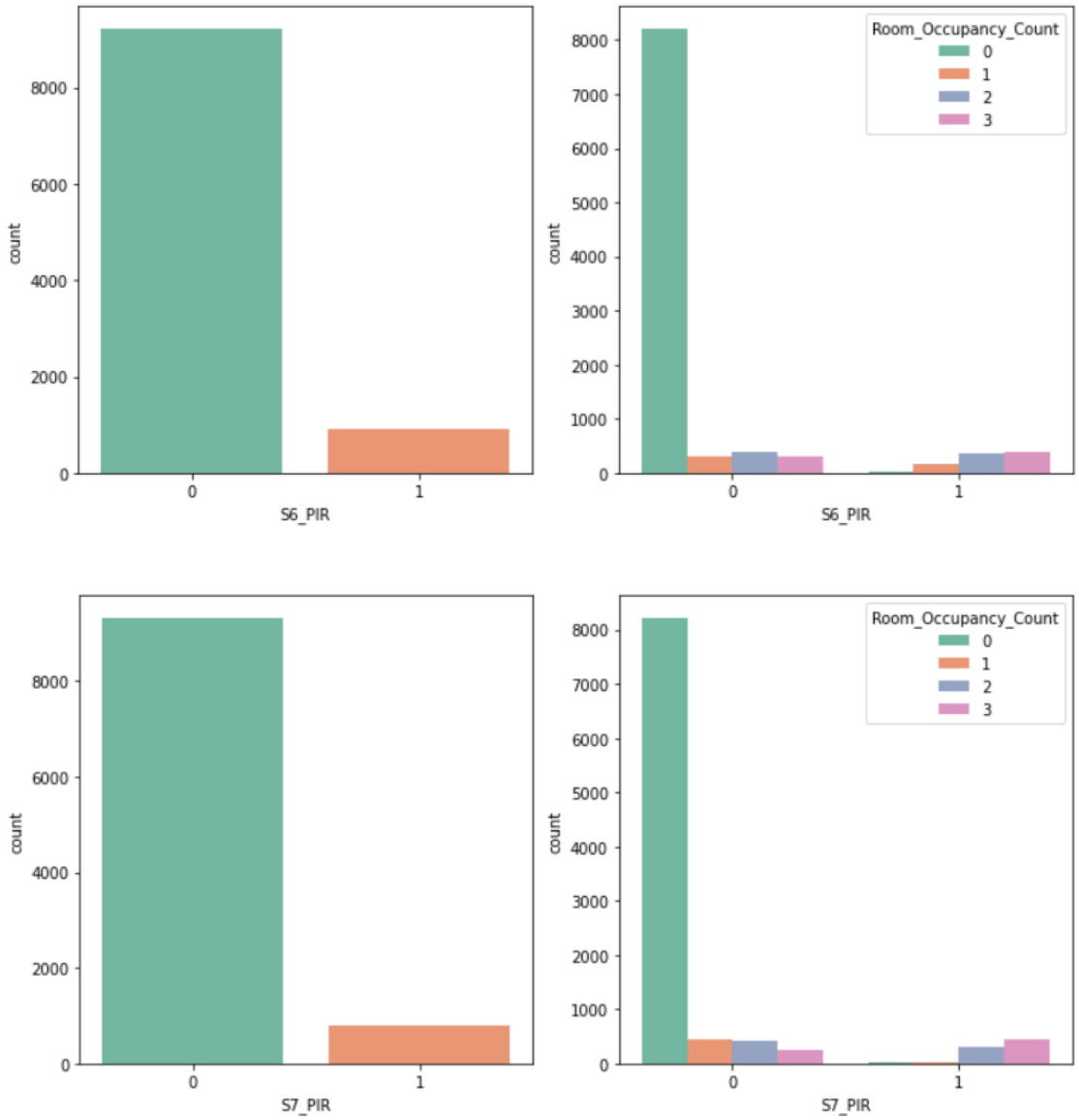
Fig 23



The Room_Occupancy_Count on the following three dates —"2017/12/23", "2017/12/22" and "2018/01/10" — indicates an occupancy of more than 1.

(S6_PIR, S7_PIR) vs Room_Occupancy_Count

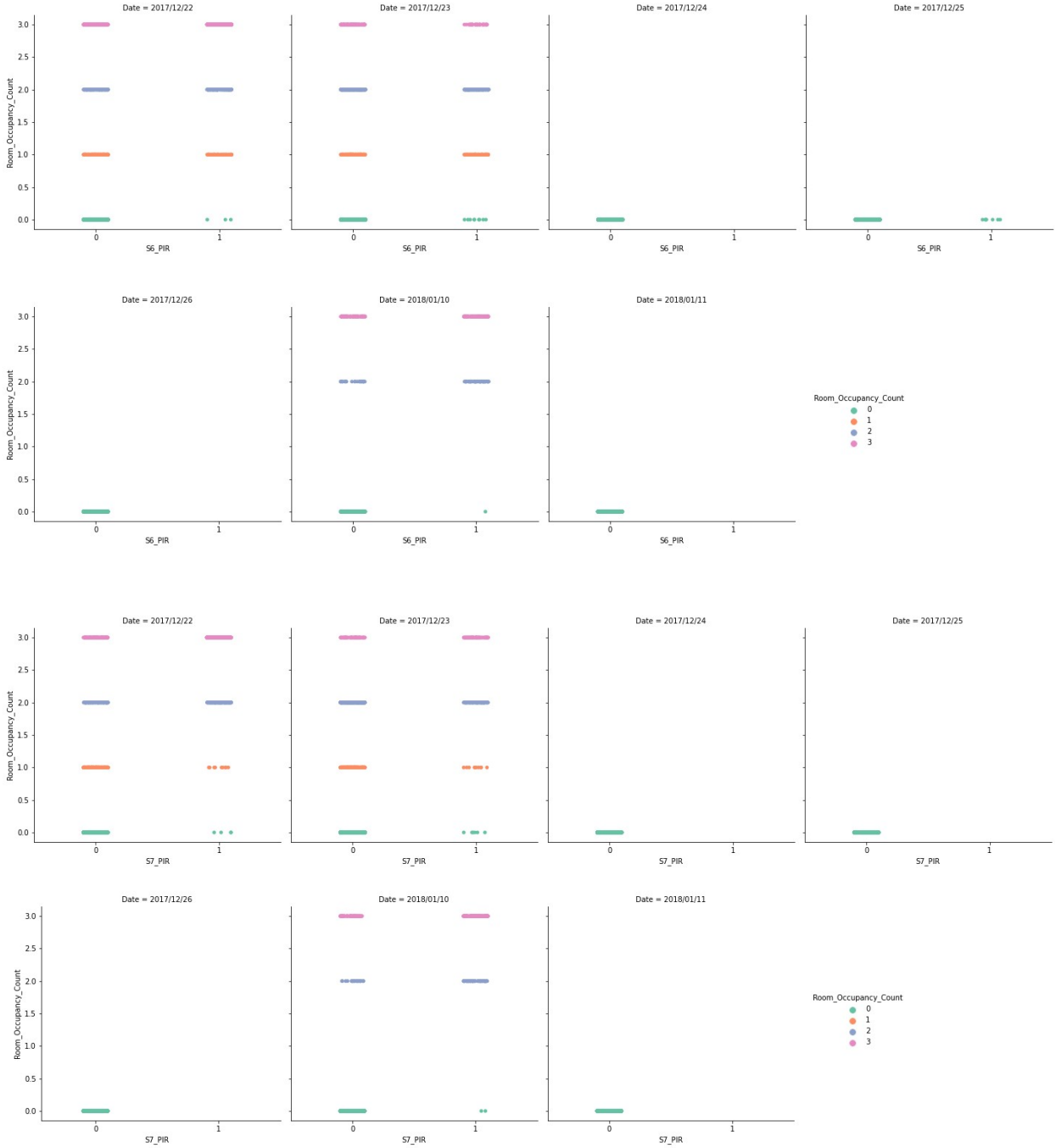
Fig 24



- The distribution of S6_PIR, S7_PIR readings reveals that the majority of instances are captured with a value of 0(absence of motion), rather than 1(presence of motion).

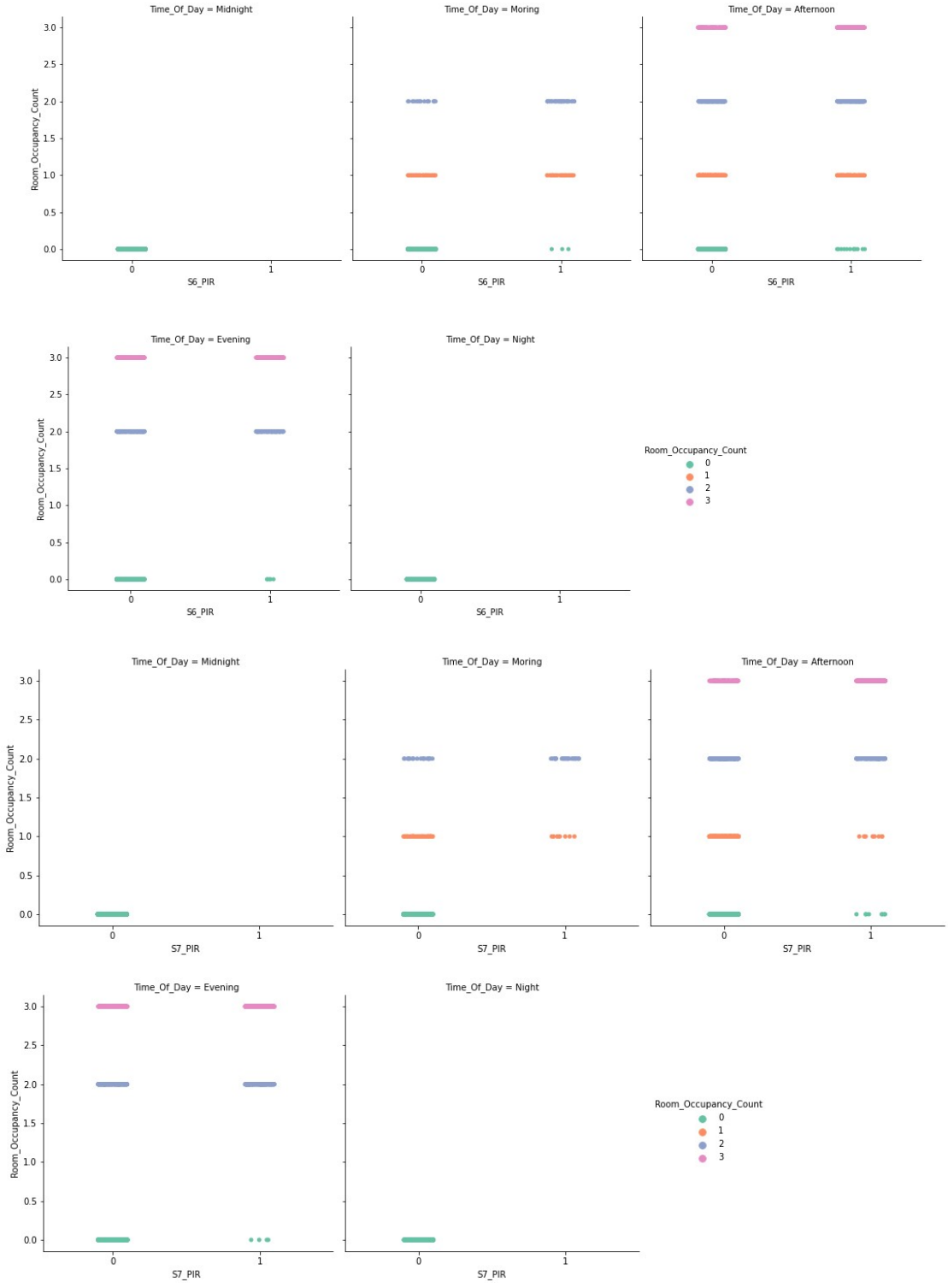
- The majority of instances with zero occupants coincide with the absence of motion, indicating that when the sensor detects no motion, the room is often unoccupied.
- There are a few instances where the sensor detects motion, but there are zero occupants. This could be due to the sensor picking up motion from sources other than human occupants.

Fig 25


Date = 2017/12/22
Date = 2017/12/23
Date = 2017/12/24
Date = 2017/12/25
Date = 2017/12/26
Date = 2018/01/10
Date = 2018/01/11
Room_Occupancy_Count
S6_PIR
S7_PIR
0
1
2
3

The values of S6_PIR and S7_PIR on the following three dates —"2017/12/23", "2017/12/22" and "2018/01/10" — implies that the sensor is capturing instances of motion when there are occupants in the room.

Fig 26


Time_Of_Day = Midnight
Time_Of_Day = Moring
Time_Of_Day = Afternoon
Time_Of_Day = Evening
Time_Of_Day = Night
Room_Occupancy_Count
S6_PIR
S7_PIR
Room_Occupancy_Count
0
1
2
3

The values of S6_PIR and S7_PIR on the following three ‘Time_Of_Day’s—"Morning", "Afternoon" and "Evening" — implies that the sensor is capturing instances of motion when there are occupants in the room.

Modeling and Evaluation

Based on Literature Review and a Comparative Study on Classic Machine Learning Algorithms[22], **the best performed** supervised machine learning models including **Linear** SVM, **RBF** SVM, **random forest** (RF) were built, and then Dimensionality Reduction using PCA was conducted. Model building was done in two phases:

1. **Homogeneous** phase: using one homogeneous sensor at a time. Each sensor group is individually examined in the model.
2. **Heterogeneous** phase: using combined homogeneous sensors, include one sensor group in each step. The focus is on examining the outperforming of the combined sensors.

Metrics:

Model selection and evaluation using tools, such as model_selection.cross_val_score[30], take a scoring parameter[31] that controls what metric they apply to the estimators evaluated. All scorer objects follow the convention that **higher return values are better than lower return values.** The **sklearn.metrics** module implements several score and utility functions to measure **classification** performance. Some of these are restricted to the binary classification case, others also work in the **multiclass** case.

Some metrics like **f1_score** are defined for binary classification tasks, so by default only the positive label is evaluated, assuming that the positive class is labeled 1. In extending to multiclass problems, one way is to average binary metric calculations across the set of classes using the **average** parameter, setting to **'macro'**. It combines precision and recall into a single value. It is calculated using the following formula:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1 score is interpreted as a harmonic mean of precision and recall. In the Room_Occupanvy_Count class, this is the average of the F1 score of each label(0, 1, 2, 3), with weighting average parameter set to 'macro', which calculates metrics for each label, and finds their unweighted mean, where both false positives and false negatives need to be considered.

The **Matthews correlation coefficient**[32] can be used as a measure of the quality of multiclass classifications, by considering each class against the rest (one-vs-all approach). It takes into account true positives, true negatives, false negatives and false positives and is particularly useful when dealing with **imbalanced** datasets. It provides a **balanced measure** even if the classes are of different sizes.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

MCC ranges from -1 to +1, where +1 indicates a perfect prediction, 0 is equivalent to random prediction, and -1 indicates a complete disagreement between prediction and observation.

For emphasizing balanced performance across all classes, MCC might be worth exploring.

The **confusion_matrix** function evaluates classification accuracy by computing the confusion matrix with each row corresponding to the true class.

The **accuracy, f1-score and MCC** play the role of the judge. These along with **Confusion Matrix** are the performance metrics used while training classification models.

At the end, the combination of features that gives the optimal results, according to the evaluation criterion, will be selected.

Cross Validation Evaluation:

Model performance was evaluated using Accuracy, F1 score, MCC, and confusion matrix metrics, considering the explanation of the model output, through Cross-validation.

Finding the evaluation technique is dependent on the characteristic of the dataset:

- **time-series** in nature, where the sequence of observations is crucial
- Existence of **class imbalance**(Room_Occupancy_count =0).

Combination of TimeSeriesSplit and StratifiedKFold

Customizing a combination of **TimeSeriesSplit** and **StratifiedKFold** cross-validation[21] could help:

- The temporal order is preserved during cross-validation (Time-Series Splitting).
- Each fold maintains a balanced distribution of Room_Occupancy_count class (StratifiedKFold)

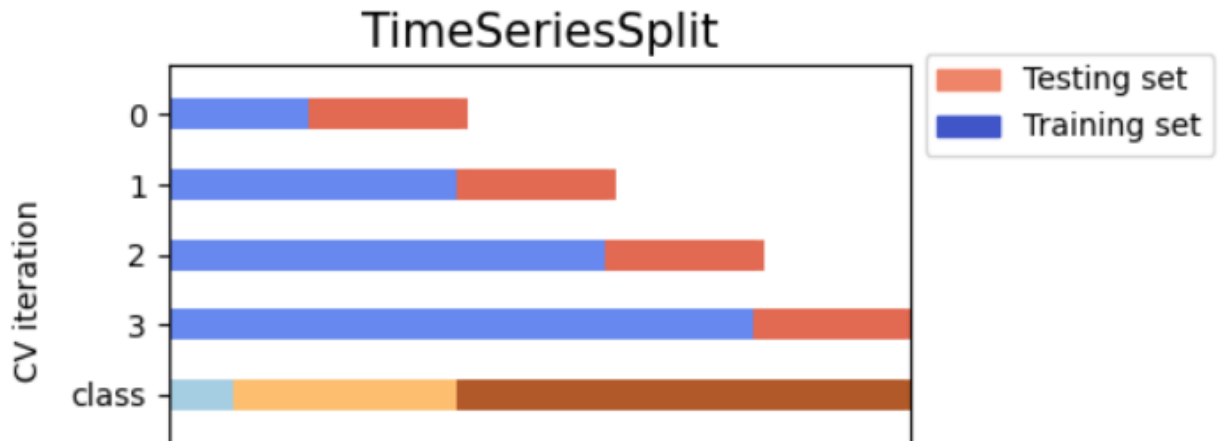
Within each TimeSeriesSplit fold, I have used StratifiedKFold, preserving **time-dependent** sampling and the temporal order of samples, avoiding data points similar to the test data, getting into the training data, while maintaining **stratified** sampling for a balanced distribution sampling of each class.

TimeSeriesSplit:

This cross-validation method has been used in order to temporal order of observations during cross-validation. Fig 27 is an illustration of how this split works:

1. **Data Splitting:** The dataset is divided into "folds", ensuring that each fold contains data points that occurred after the ones in the previous fold.
2. **Iteration Process:** In each iteration, the training set includes past data, and the test set includes future data.
3. **Sequential Nature:** The process continues until each subset has been used as a test set exactly once, ensuring that the model is tested on data points that occurred after the ones used for training.

Fig 27



Stratified KFold:

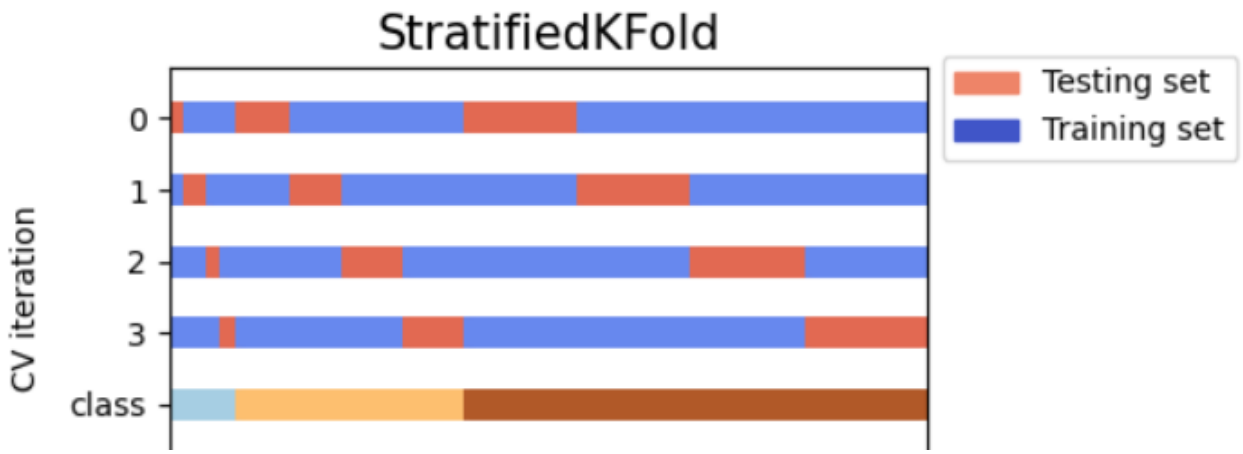
StratifiedKFold is a variation of KFold. First, it shuffles the data, then splits it into n_splits parts. Note that **it only and always shuffles data one time** before splitting. This cross-validation

method will be used to handle class imbalance, which is a variation of the standard k-Fold CV technique and is designed to be effective, preventing from being biased toward the majority class.

Fig 28 is an illustration of how this split works:

1. **Data Splitting:** Split the dataset into k folds, ensuring that each fold contains a balanced representation of target classes.
2. **Iteration Process:** train the model on k - 1 folds while validating on the remaining fold.
    - Save the validation results for each iteration.
    - Repeat the process k times, ensuring that each fold serves as the test set once.
    - Average the validation results to obtain the final performance score.

Fig 28



Time and Computation Complexity:

This combination is computationally expensive; and it takes plenty of time to cross-validate the model. The time depends on the number of samples, features, and the number of splits in the custom cross-validation.

Data Normalization

I have applied the **Normalization** technique for the transformation to ensure that features are on a similar scale. This is crucial for the models that I used which are sensitive to features with large scales. Features with large scales can have a more significant impact on the model learning process, leading to less performance.

I have chosen the **StandardScaler**[24] normalization method, transforming the features so that they have zero mean and a standard deviation of 1. Only the training set features have been used to calculate normalization. The normalization obtained from the training set then has been applied to both the training and testing sets consistently.

Machine Learning Algorithms:

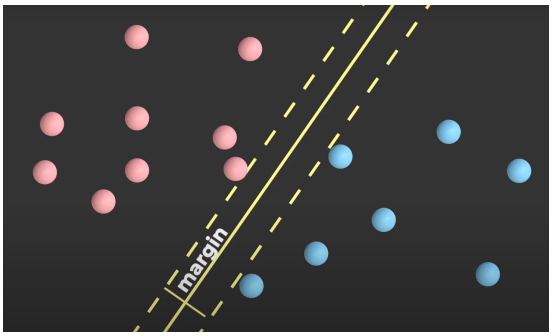
I first built the best performed models on the research paper - SVM and Random Forest - using the supervised learning algorithms and then moved on to the dimensionality reduction using PCA.

Support Vector Machines:

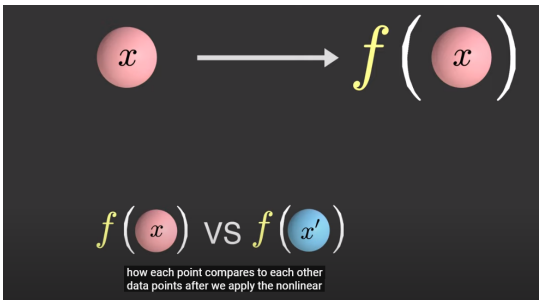
Support Vector Machines[23], is a classification machine learning algorithm that utilizes the **kernel trick**, for a non-linear transformation on points. This technique allows SVM to establish an optimal boundary between classes by mapping points of data into a higher dimensional space. Its objective is to identify the optimal separating hyperplane, maximizing the margin within the training data.

Each n-dimensional feature vector is basically a point in an n-dimensional feature space, where n is the number of features present in the dataset. Each vector belongs to one of the k classes. The algorithm attempts to fit an optimal hyperplane between the two classes with the help of support vectors. Therefore, for k classes, the number of classifiers learned by the algorithm are k(k-1)/2 which are put to a majority vote. It used a one-vs-one scheme for multiclass problems.

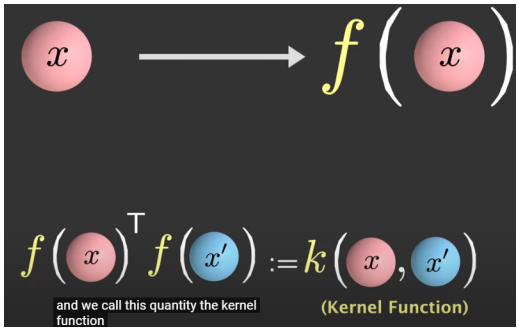
SVM performs the classification test by drawing a hyperplane, a line in 2D and a plane in 3D, in such a way that all points of one class are one side of the hyperplane. It tries to find the one that best separates the two classes, in the sense that it maximizes the distance to points to either classes. This distance is called **margin**, and the points that are exactly on the margin are called **support vectors**.



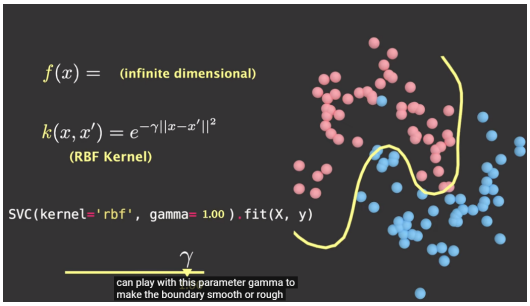
If the dataset can not be separated by the hyperplane, using the **kernel trick** will apply a non-linear transformation beforehand. The algorithm of SVM, just need to know how each point compares to each other data points after applying the non-linear transformation



Mathematically this corresponds to taking the inter-polation ($^{\intercal}$) between f (x) and f (x′), calling this kernel function: f (x)$^{\intercal}$f (x′)



Sometimes it is hard and impossible to find corresponding transformations. The prime example of this is the **Radial Basis Function**(RBF) kernel. It is also infinite dimensional. But the kernel expression is simple. Making the boundary smooth or rough, by playing with parameter of γ (gamma).



There is another parameter 'C', called penalty parameter, that tells the SVM optimization how much to avoid misclassification in each training example. A lower value for C, will result in a larger-margin separating hyperplane if that hyperplane misclassified more points. On the other hand, a higher value of C will result in a smaller-margin hyperplane if it does a better job of classifying all training points correctly.

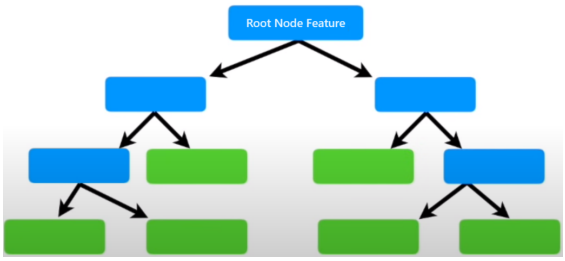
Random Forest:

Random Forests[29] are built from decision trees, those are easy to go , easy to use and easy to interpret, but in practice they are not that awesome. They are not ideal tools for predictive learning, namely inaccuracy. They are great with the data used to create them, but are not flexible to classifying the new samples. Random Forests combine the simplicity of decision trees with flexibility, resulting in a vast improvement in accuracy.

Let's make a Random Forest:

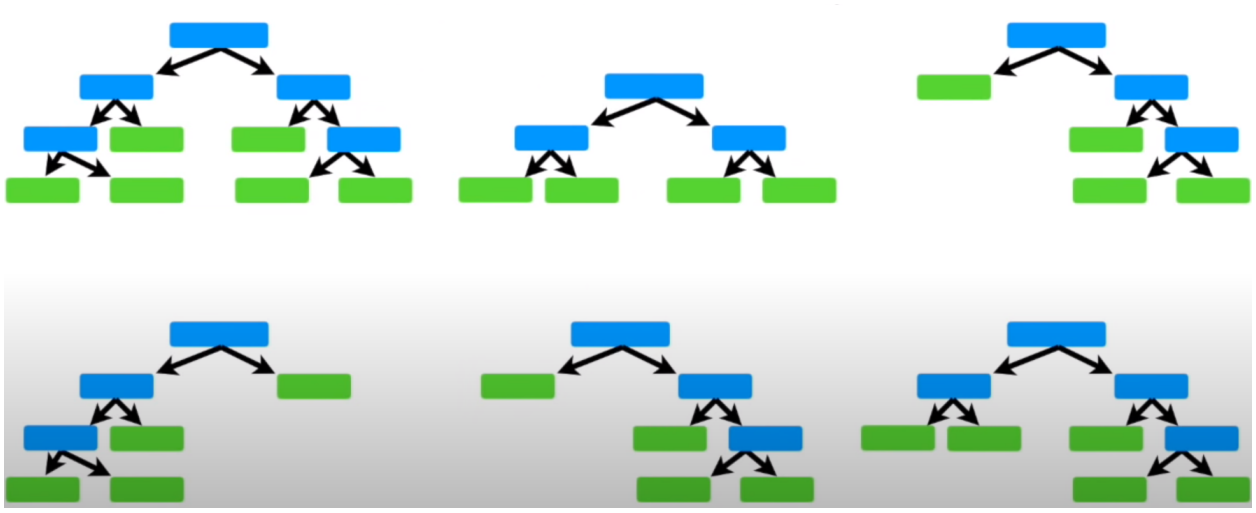
1- create a '**Bootstrapped**' dataset: the same size as the original dataset, by **randomly** selecting samples from the original dataset. The important detail is that picking the same sample is allowed. As a result some entries may not be included into the bootstrapped dataset.

2- Create a decision tree using the bootstrapped dataset, but only using a **random subset**[2] of features at each step. For example randomly select 2 features for the root node. Among them the feature that does the best job separating the samples(e.g. gini index) will be chosen. The same feature can be selected multiple times in a tree. Every time choosing will be done from the full list of features. Just like for the root node, 2 features will be randomly selected as candidates. Building the tree is done as usual, but by only considering a **random subset** of features in each step.

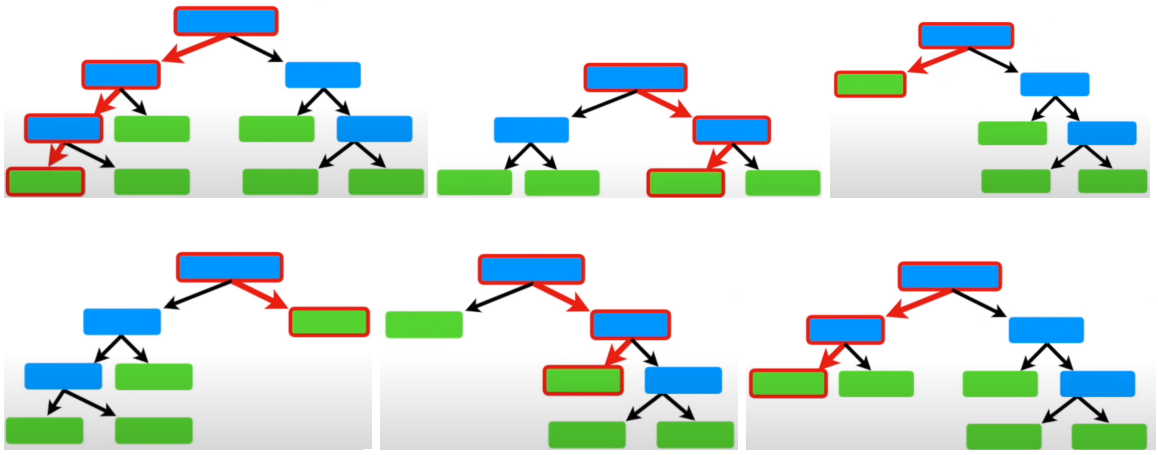


3- Step 1 is repeated, and a new bootstrapped dataset is created, followed by the construction of a tree as step 2. Ideally, this process is repeated 100's of times, resulting in a wide variety of trees but here, only six trees are showcased! The variety is what makes random forests more effective than individual decision trees.

[2] The goal is to introduce **randomness** to decrease the variance of the forest estimator, addressing the high variance and overfitting tendencies of individual decision trees.



How to use the Random Forest? For a specific sample, it predicts the proper class. The data was taken and the first tree was run down on it. By keep tracking of that:



After running the data down all of the trees in Random Forest, it will be to the class which received the **most** votes.

Bootstrapping the data plus using the **aggregate** to make a decision is called '**Bagging**'. Considering the first step of making RF, typically, ⅓ of the original dataset entries, does not end

up in the bootstrapped dataset. These entries are called '**Out-Of-Bag-Dataset**'. As they were not used to create the RF trees, the Random Forest process runs all of these Out-Of-Bag samples through all the RF trees to find its classified labels. The label with the most votes wins, and this Out-Of-Bag sample will be assigned to that label. Ultimately, the accuracy of Random Forest will be measured by the **proportion** of Out-Of-Bag samples that were correctly classified by the Random Forest.

The proportion of Out-Of-Bag samples that were incorrectly classified is the '**Out-Of-Bag-Error**'. While building a Random Forest, the number of features used per step is changing, by testing different numbers of features, and estimating the '**Out-Of-Bag-Error**', the most accurate Random Forest can be chosen. Typically, it starts by using the square root of the number of features and then a few settings above or below that value, will be tried.

Result

Baseline Models

Homogeneous Phase

In the initial supervised learning phase, one homogeneous sensor used for models at a time, the result are detailed in the Table VIII:

Table VIII

SVM(Linear) - TimeSeriesSplit

	Feature	A	F1
0	['Temp{1,2,3,4}']	0.82	0.621
1	['Light{1,2,3,4}']	0.986	0.968
2	['Sound{1,2,3,4}']	0.777	0.509
3	['PIR{6,7}']	0.758	0.442
4	['CO2']	0.692	0.338
5	['Slope']	0.759	0.422
6	['CO2', 'Slope']	0.839	0.622

SVM(RBF) - TimeSeriesSplit

	Feature	A	F1
0	['Temp{1,2,3,4}']	0.833	0.703
1	['Light{1,2,3,4}']	0.986	0.969
2	['Sound{1,2,3,4}']	0.796	0.57
3	['PIR{6,7}']	0.749	0.424
4	['CO2']	0.647	0.342
5	['Slope']	0.755	0.461
6	['CO2', 'Slope']	0.85	0.68

Random Forest - TimeSeriesSplit

	Feature	A	F1
0	['Temp{1,2,3,4}']	0.809	0.714
1	['Light{1,2,3,4}']	0.967	0.939
2	['Sound{1,2,3,4}']	0.8	0.603
3	['PIR{6,7}']	0.748	0.423
4	['CO2']	0.602	0.347
5	['Slope']	0.724	0.473
6	['CO2', 'Slope']	0.809	0.637

The result show the performance metrics for SVM (Linear), SVM (RBF), and Random Forest models on homogeneous sensors:

- The Light sensors is the best-performing feature set across all three models
- SVM(RBF) performs slightly better among the other three models.
- When light features are not considered, Temp sensors are good estimators for the number of occupants accurately.
- Even with as low as Four components, SVM with RBF kernel gives an accuracy of around 83% and a moderate F1 score of 0.703.

Heterogeneous Phase

The next phase involved heterogeneous feature combinations that were generated by iteratively adding one sensor type at a time until the complete dataset was established, light was considered only in the end. The result are detailed in the Table IX:

Table IX

SVM(Linear) - TimeSeriesSplit

	Feature	A	F1	MCC
0	['Light{1,2,3,4}', 'Sound{1,2,3,4}', 'Temp{1,2,3,4}', 'CO2', 'Slope', 'PIR{6,7}']	0.976	0.95	0.954
1	['Sound{1,2,3,4}', 'Temp{1,2,3,4}', 'CO2', 'Slope', 'PIR{6,7}']	0.906	0.821	0.815
2	['Sound{1,2,3,4}', 'Temp{1,2,3,4}', 'CO2', 'Slope']	0.901	0.812	0.807
3	['Temp{1,2,3,4}', 'CO2', 'Slope']	0.86	0.704	0.718

SVM(RBF) - TimeSeriesSplit

	Feature	A	F1	MCC
0	['Light{1,2,3,4}', 'Sound{1,2,3,4}', 'Temp{1,2,3,4}', 'CO2', 'Slope', 'PIR{6,7}']	0.975	0.954	0.958
1	['Sound{1,2,3,4}', 'Temp{1,2,3,4}', 'CO2', 'Slope', 'PIR{6,7}']	0.894	0.82	0.812
2	['Sound{1,2,3,4}', 'Temp{1,2,3,4}', 'CO2', 'Slope']	0.897	0.83	0.819
3	['Temp{1,2,3,4}', 'CO2', 'Slope']	0.854	0.742	0.737

Random Forest - TimeSeriesSplit

	Feature	A	F1	MCC
0	['Light{1,2,3,4}', 'Sound{1,2,3,4}', 'Temp{1,2,3,4}', 'CO2', 'Slope', 'PIR{6,7}']	0.964	0.932	0.935
1	['Sound{1,2,3,4}', 'Temp{1,2,3,4}', 'CO2', 'Slope', 'PIR{6,7}']	0.892	0.813	0.811
2	['Sound{1,2,3,4}', 'Temp{1,2,3,4}', 'CO2', 'Slope']	0.887	0.806	0.803
3	['Temp{1,2,3,4}', 'CO2', 'Slope']	0.847	0.739	0.728

Table IX shows the per-formance metrics for SVM (Linear), SVM (RBF), and Random Forest models on heterogeneous sensors :

- SVM with RBF kernel performed better than linear SVM for most cases.
- As expected, the complete dataset (First rows) which includes all the sensors including , performs the best at estimating the number of occupants accurately.
- When complete dataset is considered, **SVM(RBF) performs slightly better among the other three models.**
- When light features are not considered, **SVM(Linear) performs slightly better among the other three models.**
- **F1 Score** and **MCC** are both metrics that consider both false positives and false negatives. However, MCC is considered more robust for imbalanced datasets.
- The confusion matrices for the best case in both the feature sets are shown in Table X and Table XI

Table X

CONFUSION MATRIX FOR **LINEAR SVM** CASE FOR THE COMPLETE DATASET **DEVOID OF LIGHT FEATURES**

Actual\Predicted	Predicted 0	Predicted 1	Predicted 2	Predicted 3
Actual 0	7199	44	37	28
Actual 1	104	336	19	0
Actual 2	62	47	507	132
Actual 3	24	4	150	516

Table XI

CONFUSION MATRIX FOR **SVM WITH RBF KERNEL** CASE FOR THE **COMPLETE DATASET**

Actual\Predicted	Predicted 0	Predicted 1	Predicted 2	Predicted 3
Actual 0	7280	2	3	23
Actual 1	0	453	6	0
Actual 2	0	0	718	30
Actual 3	11	0	68	615

The **Precision-Recall** is a useful measure of success of prediction because Room_Occupancy_Count is **imbalanced**. If the cost of false negatives and False Positives are both important, the interpretation of precision and recall becomes crucial.

Precision: is the ratio of correctly predicted positive observations(**Correct Occupancy**) to the total predicted positives(**Correct + Incorrect Occupancy of all classes**). It focuses on the accuracy of the model in correctly identifying occupied rooms among the predicted occupied instances(**avoiding unnecessary energy consumption**).

- Precision=True Positives ／(True Positives+False Positives)

Recall (Sensitivity): is the ratio of correctly predicted positive observations(**Correct Occupancy**) to all observations in the **actual** class. It focuses on the ability to capture the actual occupied instances(**fewer discomfort for occupants)**.

- Recall=True Positives ／(True Positives+False Negatives)

False Positives (Type I Error):

  - Precision is important because it measures how accurate the model is when it predicts an occupied room. A high precision means that when the model predicts occupancy, it is likely correct, reducing the chance of **unnecessary energy consumption**.

False Negatives (Type II Error):

  - Recall is even more crucial in this scenario. High recall means the model is effective in capturing the majority of actual occupied instances. This is essential for preventing underestimation of occupancy and **discomfort for occupants**, which is not desirable.

Aim for high recall while maintaining a reasonable precision. This ensures that the model is effective in identifying most of the occupied instances, minimizing the risk of energy inefficiency and discomfort caused by underestimation.

Looking at the result, **F1 Score** and **MCC** metrics that consider both false positives and false negatives, indicate best performance for

- complete dataset SVM(RBF) model

- Dataset without light features SVM(Linear)

Dimensionality Reduction(PCA):

As the results show, there is a bias of highest performance towards light sensors. It can be explained by the practice of individuals turning on lights upon arrival and off upon departure. However, relying on light sensors can lead to false positives when individuals leave the room with lights still on. Hence, despite its highest performance, lights should be excluded.

Performing **Principal Component Analysis**(PCA) on the feature set without lights, is another attempt to exclude light and also reduce the dimension from 12 to a significantly smaller value.

Table XII

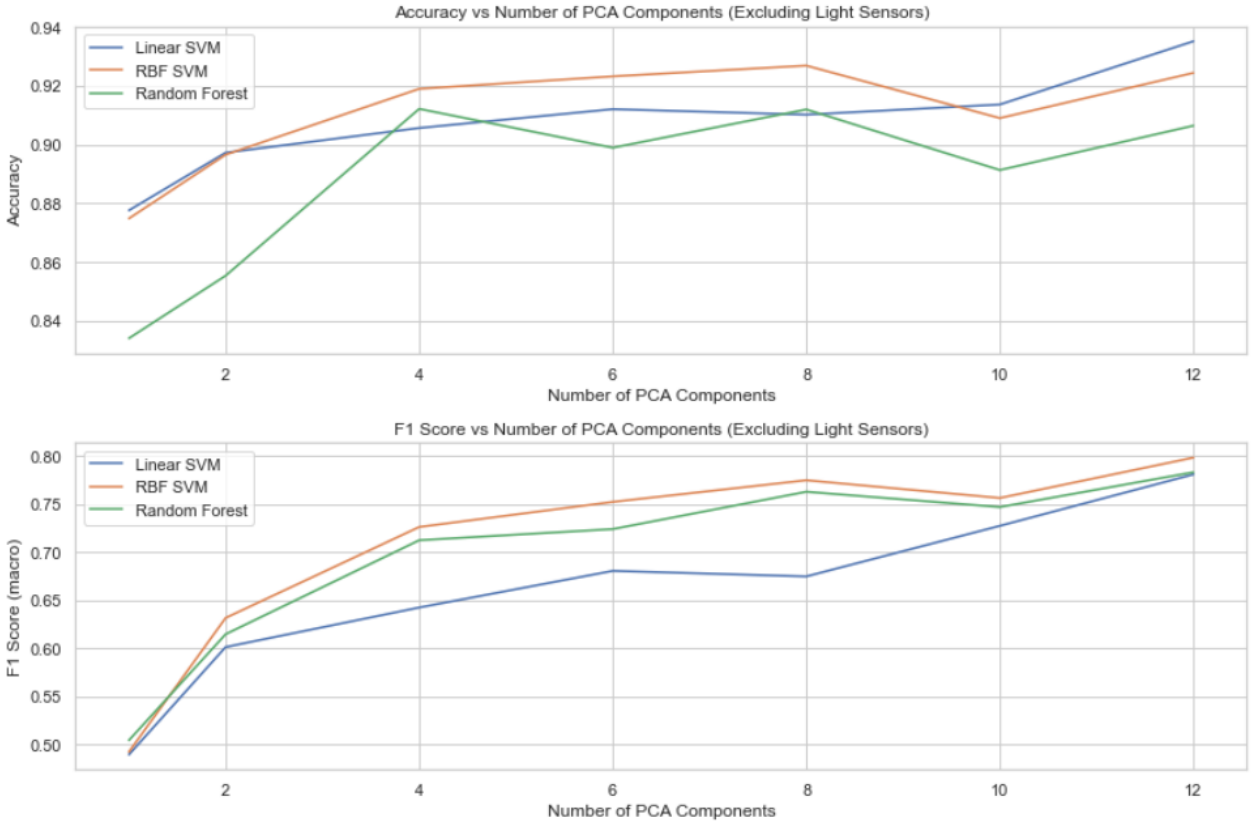


Table XII shows the variation in accuracy and F1 score of all the five ML models respectively with the number of PCA components. It shows that even with as low as four components, SVM

with RBF kernel gives an accuracy of 92% and a moderate F1 score of 0.72. Random Forest also shows a similar performance.

Tuning Hyperparameter:

Tuning hyperparameters for the baseline models involves searching for the best combination of hyperparameter values that optimize the model's performances. I performed hyperparameter tuning using GridSearchCV[33] for the Random Forest model.

Common hyperparameters for Random Forest model include:

- the number of trees(n_estimators)
- the maximum depth of the trees (max_depth)
- the minimum samples required to split an internal node (min_samples_split).

I adjusted the 'n_estimators' parameter to [50, 100, 200] trees, 'max_depth' to [None, 10, 20], and 'min_samples_split' to [2, 5, 10], for tuning. The best hyperparameters have been extracted using grid_search.best_params_, on the train set.

After finding the best hyperparameters, It has been proceeded on the validation set and then evaluated on the test sets.

```
RandomForest Best Hyperparameters: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 200}
Accuracy: 0.9634748272458046, F1 Score: 0.8887821976919197
```

Then I have tuned the final RF model using the best hyperparameters to optimize the model performance. Here is the metrics:

Tuned Random Forest

	Feature	A	F1	MCC
0	['Sound{1,2,3,4}', 'Temp{1,2,3,4}', 'CO2', 'Slope', 'PIR{6,7}']	0.891	0.813	0.811
1	['Sound{1,2,3,4}', 'Temp{1,2,3,4}', 'CO2', 'Slope']	0.889	0.807	0.806
2	['Temp{1,2,3,4}', 'CO2', 'Slope']	0.846	0.736	0.728

Analyzing the performance metrics of the tuned Random Forest model and comparing to the baseline RF model, reveals comparable results. This suggests that the baseline model, without extensive tuning, is proving to be both reliable and robust in its ability to estimate occupancy accurately. This finding proves confidence in the effectiveness of the baseline model.

Conclusion And Future Work

I explored the task of replicating Room Occupancy Estimation, focusing on the application of machine learning models, Support Vector Machines(SVMs) with linear and Radial Basis Function(RBF) kernels, as well as Random Forests, those with best reliable performance in the research paper. The dataset used for this analysis contains information from various sensors, including temperature, light, sound, PIR (Passive Infrared), CO2 levels.

I explored different feature sets based on homogeneous sensors, investigated performance with different feature combinations, utilized a custom TimeSeriesSplit strategy with nested StratifiedKFold for model evaluation, evaluated SVM (Linear), SVM (RBF), and Random Forest models, using accuracy, F1 score, and Matthews Correlation Coefficient(MCC), performed Principal Component Analysis(PCA) on the feature set without lights, and at last tuned Random Forest model using the best hyperparameter.

The finding delivered reliable results:

Feature Importance:

- ☑ Different combinations of features resulted in varied model performance.
- ☑ Certain sensors, such as temperature and CO2, played a significant role in predicting room occupancy.
- ☑ The Light is the best-performing feature set across all three models. But it was excluded from the feature sets, because relying on light can lead to false positives when individuals leave the room with lights still on.

Model Performance:

- ☑ SVM (Linear) and SVM (RBF) exhibited competitive performance
- ☑ Random Forest also provided reliable results, showcasing the power of ensemble methods.
- ☑ For a complete feature set the results show a promising 97.5% accuracy of occupancy estimation with a high F1 score of 0.954 using SVM with RBF kernel.

Cross-Validation Strategy:

- ☑ Custom TimeSeriesSplit combined with nested StratifiedKFold addressed time-series nature and class imbalance.

Dimensionality Reduction:

- ☑ Even with as low as four components, SVM with RBF kernel gives an accuracy of 92% and a moderate F1 score of 0.72

Hyperparameter Tuning:

- ☑ Utilized GridSearchCV for hyperparameter tuning.
- ☑ Tuned parameters for Random Forest model.
- ☑ Analyzing the performance metrics of the tuned Random Forest model and comparing it to the baseline RF model, reveals comparable results, and proves confidence in the effectiveness of the baseline model.

In conclusion, the Room Occupancy Estimation was approached by employing different machine learning models, feature sets, and cross-validation techniques. The custom TimeSeriesSplit strategy provided a robust evaluation, and findings suggested competitive performance for all SVM (Linear) and SVM (RBF), and Random Forest ML models, delivered reliable results for developing accurate and reliable room occupancy estimation.

Moving forward, consideration of other algorithms and ensemble methods could be explored for further improvement.

Repository Navigation

https://github.com/masram-ml/rpstry-bdap-occup-estmt-

References

[1] Adarsh Pal Singh, Vivek Jain, Sachin Chaudhari, Frank Alexander Kraemer, Stefan Werner and Vishal Garg, "Machine Learning-Based Occupancy Estimation Using Multivariate Sensor Nodes," in 2018 IEEE Globecom Workshops (GC Wkshps), 2018.

[2] https://archive.ics.uci.edu/dataset/864/room+occupancy+estimation

[3] F. Pedregosa et al., “Scikit-learn: Machine learning in python,” J. Mach. Learn. Res., vol. 12, pp. 2825–2830, Nov. 2011.

[4] P. Perner, Machine Learning and Data Mining in Pattern Recognition: 10th International Conference, MLDM 2014, St.Petersburg, Russia, July 21-24, 2014, Proceedings. Springer International Publishing, 2014.

[5] B. Dong et al., “An information technology enabled sustainability test-bed (itest) for occupancy detection through an environmental sensing network,” Energy and Buildings, vol. 42, no. 7, pp. 1038– 1046, 2010.

[6] Z. Yang, N. Li, B. Becerik-Gerber, and M. Orosz, “A multisensor based occupancy estimation model for supporting demand driven HVAC operations,” in Proc. Symp Simulation Architecture and Urban Des. (SimAUD), San Diego, CA, USA, 2012, pp. 2:1–2:8.

[7] M. Masood, Y. Soh, and V. Chang, “Real-time occupancy estimation using environmental parameters,” in Int. Joint Conf. on Neural Netw. (IJCNN), Jul. 2015, pp. 1–8.

[8] Z. Chen et al., “Building occupancy estimation with environmental sensors via CDBLSTM,” IEEE Trans. Ind. Electron., vol. 64, no. 12, pp. 9549–9559, Dec 2017.

[9] A. Dey et al., “Namatad: Inferring occupancy from building sensors using machine learning,” in IEEE 3rd World Forum Internet of Things (WF-IoT), Dec. 2016, pp. 478–483.

[10] Whatley, M. (2022). One-Way ANOVA and the Chi-Square Test of Independence. In: Introduction to Quantitative Analysis for International Educators. Springer Texts in Education. Springer, Cham. https://doi.org/10.1007/978-3-030-93831-4_5

[11] https://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection, https://medium.com/analytics-vidhya/feature-selection-73bc12a9b39e , https://medium.com/analytics-vidhya/feature-selection-85539d6a2a88 , https://medium.com/analytics-vidhya/feature-selection-embedded-methods-a7940036973f

[12]https://www.freecodecamp.org/news/how-machines-make-predictions-finding-correlations-in-complex-data-dfd9f0d87889/

[13]https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html

[14] https://scikit-learn.org/stable/

[15] https://seaborn.pydata.org/

[16] https://matplotlib.org/

[17] https://scipy.org/

[18] https://docs.python.org/3/library/statistics.html

[19] https://www.statsmodels.org/stable/index.html

[20]https://www.statsmodels.org/dev/generated/statsmodels.stats.multicomp.pairwise_tukeyhsd.html

[21] https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right , https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation , https://scikit-learn.org/stable/auto_examples/model_selection/plot_cv_indices.html

[22]https://medium.com/@dannymvarghese/comparative-study-on-classic-machine-learning-algorithms-part-2-5ab58b683ec0

[23]https://scikit-learn.org/stable/modules/svm.html

[24]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[25]https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html

[26]https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html#sklearn.feature_selection.f_classif

[27]https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest

[28]https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html#sklearn.feature_selection.chi2

[29]https://scikit-learn.org/stable/modules/ensemble.html

[30]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

[31]https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter

[32]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html#sklearn.metrics.matthews_corrcoef

[33]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html