IE 7300: STATISTICAL LEARNING FOR ENGINERRING

ROOM OCCUPANCY ESTIMATION

Group 8

Members-

Aadarsh Siddha

siddha.a@northeastern.edu

Akshita Singh

singh.akshi@northeastern.edu

Praneith Ranganath

ranganth.p@northeastern.edu

Sophie Chun Hui Yang

yang.sophi@northeastern.edu

Yashasvi Sharma

sharma.yasha@northeastern.edu

INSTRUCTOR: RAMIN MOHAMMADI

DATE: 19th APRIL 2024

Abstract

In modern sustainable building management, understanding and optimizing indoor environments for comfort and energy efficiency are crucial objectives. This study focuses on predicting room occupancy using non-intrusive environmental sensors like CO2, temperature, sound, and light sensors. By accurately estimating the number of occupants in a room, energy-saving measures can be implemented, reducing both energy consumption and carbon footprint. The primary goal of this project is to develop a predictive model for estimating room occupancy, enabling real-time occupancy information to drive intelligent heating, ventilation, air conditioning, and lighting systems in residential and commercial buildings.

Introduction

The sustainable management of buildings has become increasingly important in contemporary times, with a strong emphasis on optimizing indoor environments for both comfort and energy efficiency. A key aspect of this management involves accurately estimating room occupancy, as it enables the implementation of demand-driven systems for heating, ventilation, air conditioning, and lighting. Traditionally, such systems operated on fixed schedules, leading to inefficient energy usage. However, with advancements in sensor technology, particularly non-intrusive environmental sensors, it is now possible to predict room occupancy in real-time. This predictive capability allows for the development of intelligent building systems that adjust dynamically based on occupancy levels, conserving energy and enhancing occupant comfort.

Data Description

The dataset used in this project is sourced from the UCI Machine Learning Repository (https://archive.ics.uci.edu/dataset/864/room+occupancy+estimation).

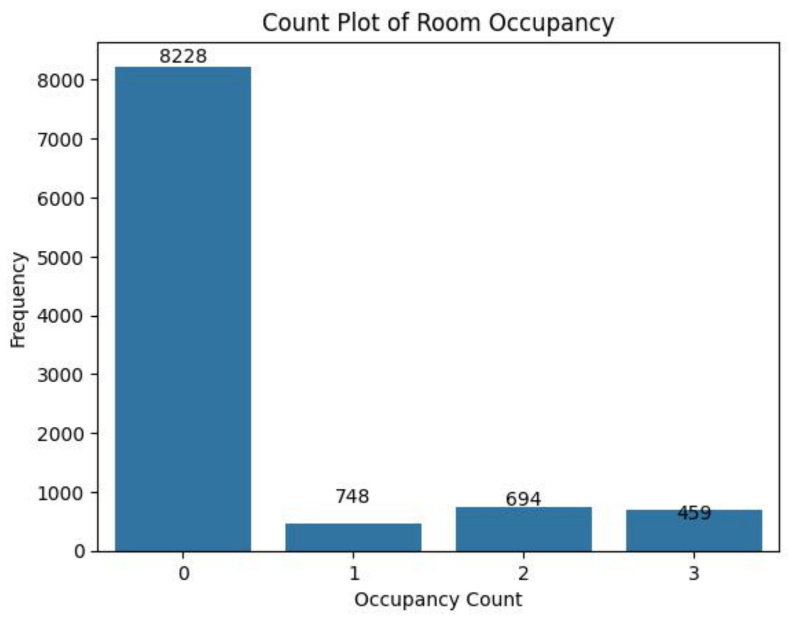
It captures room occupancy in a 6m x 4.6m room with 7 sensor nodes and an edge node. Data was collected wirelessly every 30 seconds, featuring sensors for temperature, light, sound, CO2, and PIR. The experiment spanned 4 days, with occupancy ranging from 0 to 3 people, validated manually. It contains 10,129 data points and 18 columns.

Column	**Datatype**	**Description**
Date	Object	Date of the experiment (yy/mm/dd format)
Time	Object	Time of the experiment (00:00:00)
S1_Temp, S2_Temp, S3_Temp, S4_Temp	Float64	Temperature readings in Celsius from Sensor nodes 1 through 4.
S1_Light, S2_Light, S3_Light, S4_Light	Int64	Light intensity readings in Lux from Sensor nodes 1 through 4.
S1_Sound, S2_Sound, S3_Sound, S4_Sound	Float64	Sound sensor readings in Volts from Sensor nodes 1 through 4 (amplifier output read by ADC).
S5_CO2	Int64	CO2 concentration in parts per million (PPM) from Sensor Node 5.
S5_CO2_Slope	Float64	Slope of CO2 values taken in a sliding window, which represents the rate of change of CO2 concentration.
S6_PIR, S7_PIR	Int64	Binary values conveying motion detection from Sensor nodes 6 and Value of 1 indicates motion detected, while 0 indicates no motion.
Room_Occupancy_Count	Int64	Count of room occupancy (ranging from 0 to 3).

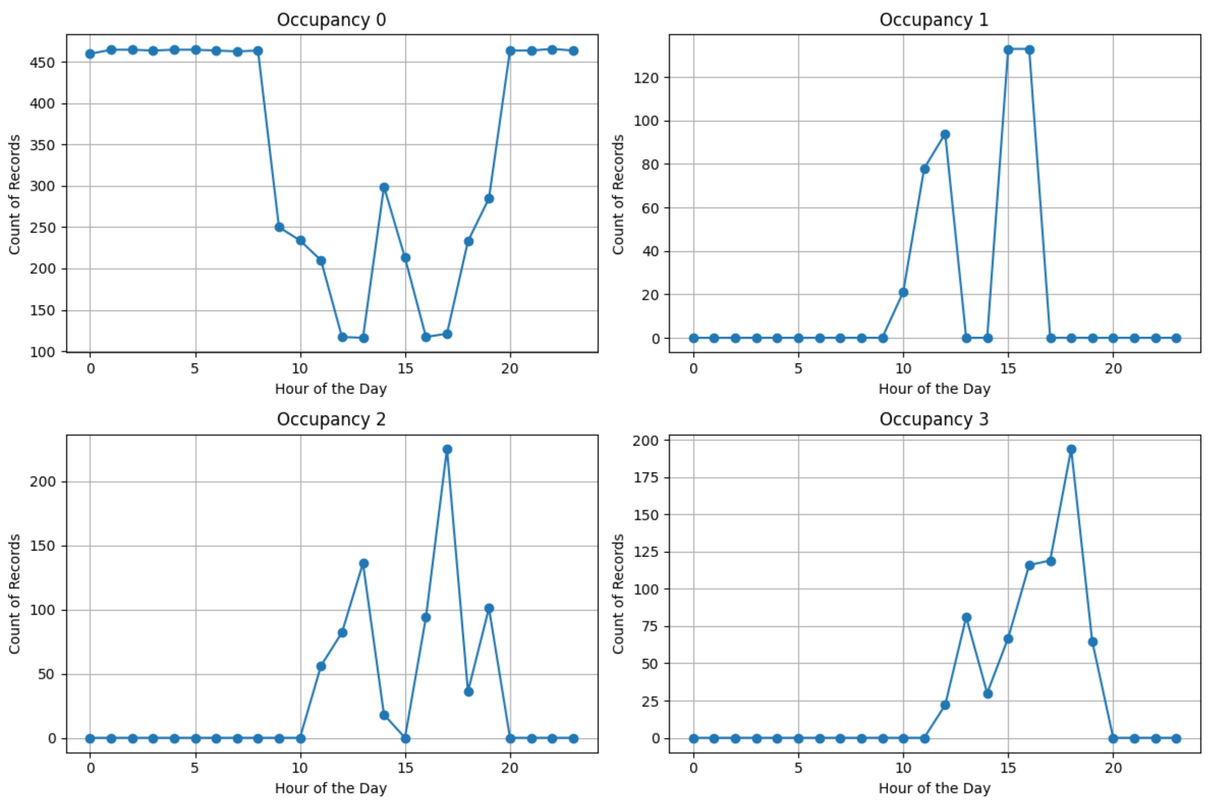
Exploratory Data Analysis (EDA):

The initial exploration of the data involved examining the data types and summary statistics. We then proceeded to check for null values, finding that there were none present in any column.

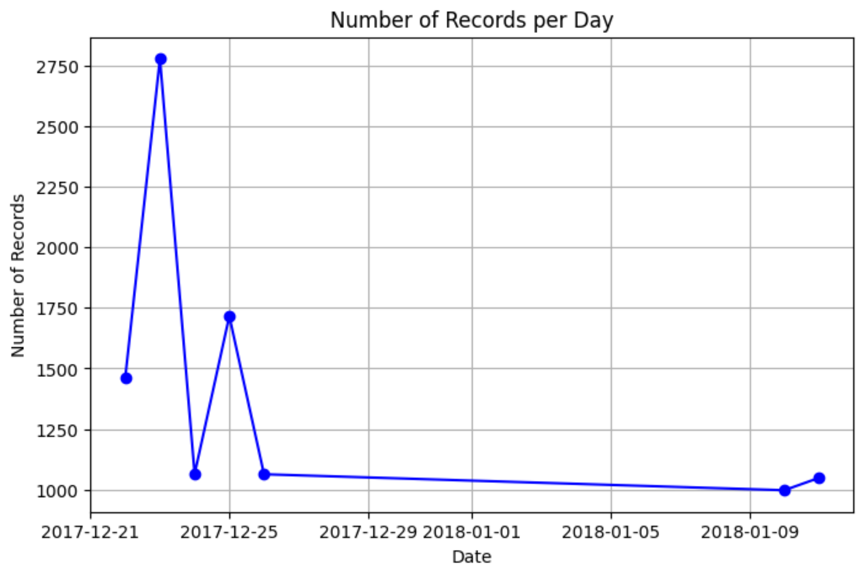
The countplot of the class variable highlights a significant class imbalance. Specifically, there are 8228 data points for class 0, while classes 1, 2, and 3 have 748, 694, and 459 data points, respectively. This imbalance necessitates balancing the data in the training set before applying any modeling techniques to ensure fair representation of all classes and prevent bias towards the majority class.



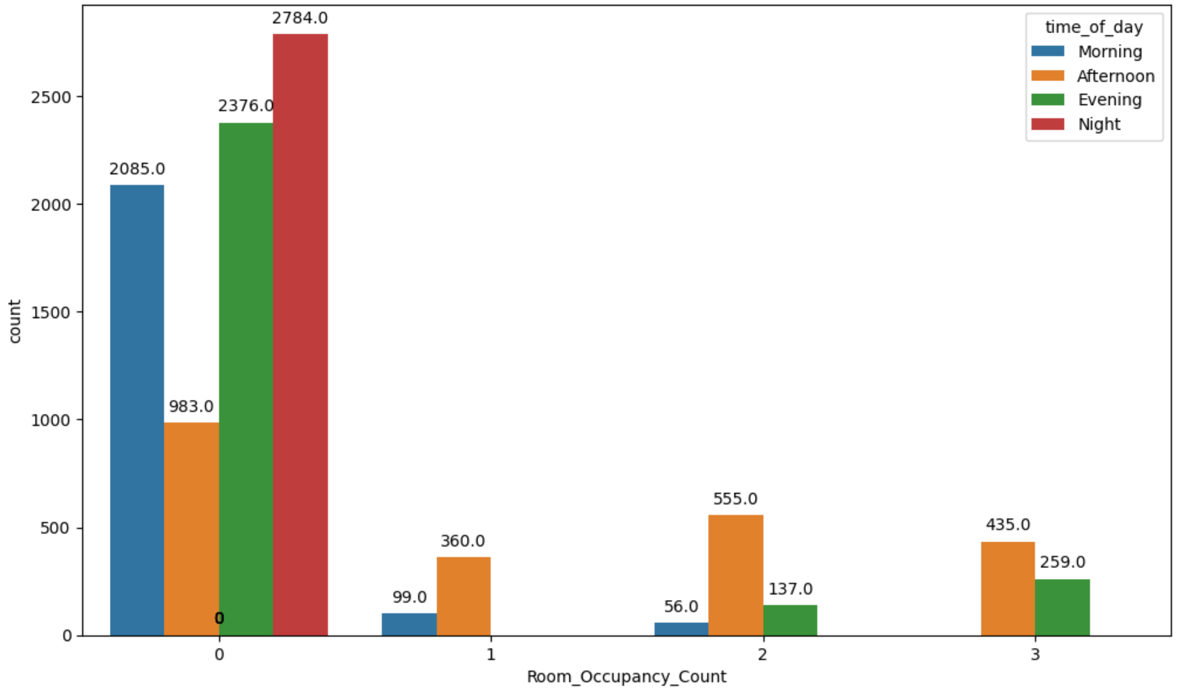
Subsequently, we created two additional features, 'hour_of_the_day' and 'day_of_the_week', derived from the 'Date' and 'Time' columns. These new features are intended to facilitate further exploratory data analysis (EDA) by providing additional temporal information, enabling insights into potential patterns or trends based on time of day and day of the week.



In our analysis of the class distribution within the dataset, visualization techniques were employed to discern patterns based on the time of day. It was observed that Class 0 predominantly has the highest number of records during the nighttime hours. Conversely, Classes 1, 2, and 3 exhibit a higher frequency of records during daytime hours.

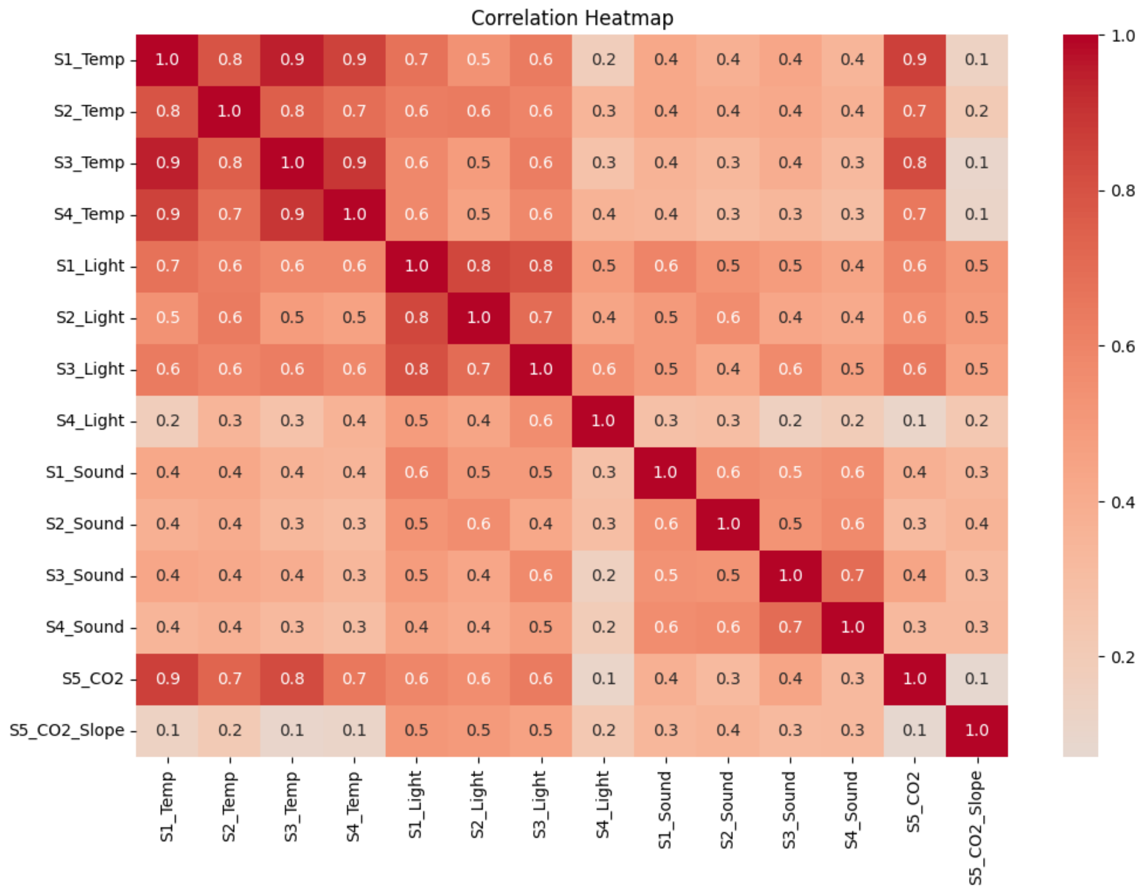


Upon visualizing the distribution of records, it is evident that there is a notably high concentration of records on a single day. This peak in data accumulation suggests that this day is significantly influencing the volume of records collected and may affect our models.



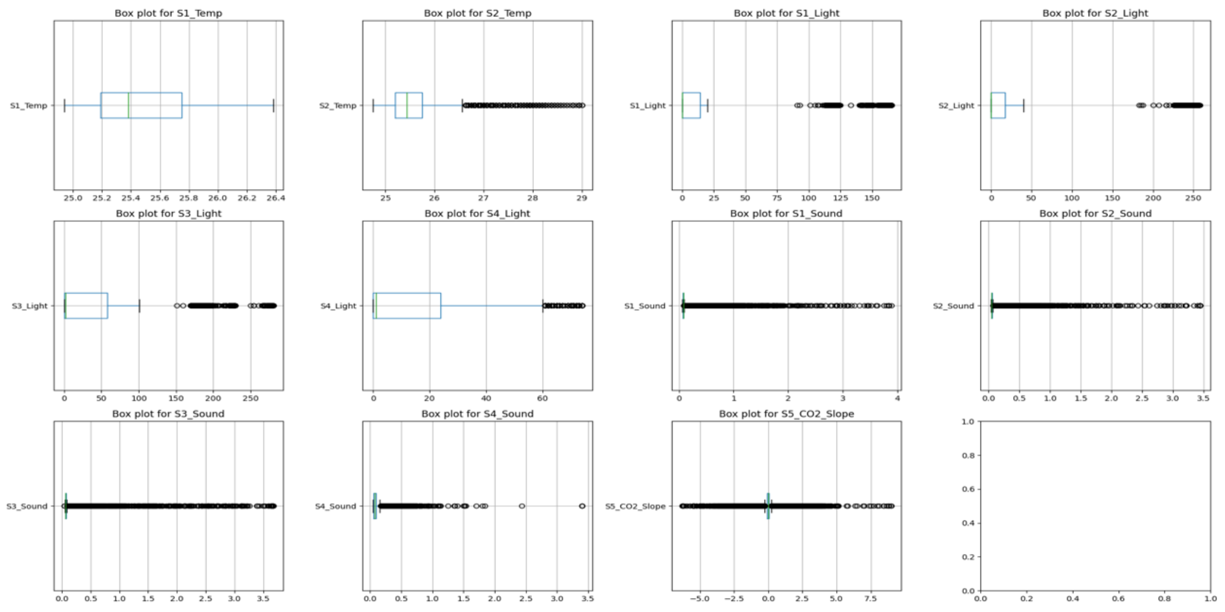
As we have seen in the graph, there are 0 data values for class 0 during Afternoon, Evening, Morning and a lot of values for Night only. For our learning purposes we don't want the effect of hour of day to skew the results.

After removing date and time-related columns, we examined the correlation between the numerical features. The correlation heatmap revealed several highly correlated features. Consequently, we opted to drop features with correlations of 0.9 and above which were 'S3_Temp', 'S4_Temp' and 'S5_CO2'.

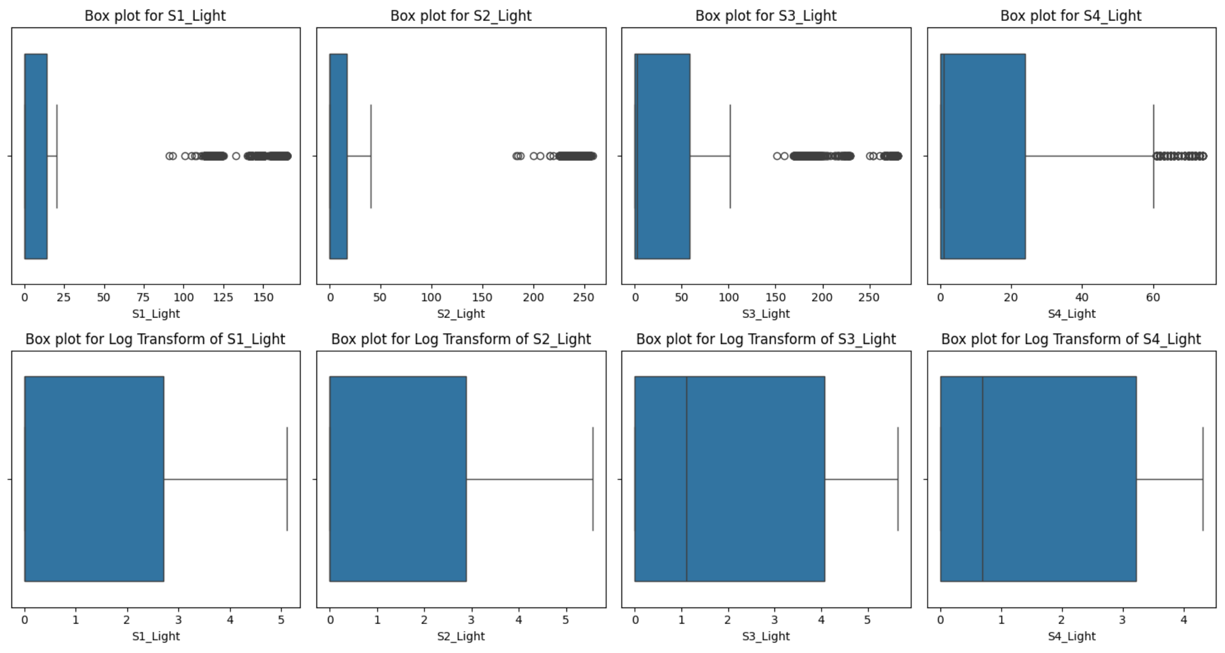


After that we checked for duplicates in the data. We identified nearly 1300 duplicate rows, which we then eliminated, resulting in a final dataset containing 8828 rows.

Next, we conducted outlier detection by plotting box plots of the numerical features. These visualizations revealed the presence of outliers in the dataset, indicating the need for further investigation and potential outlier treatment methods.

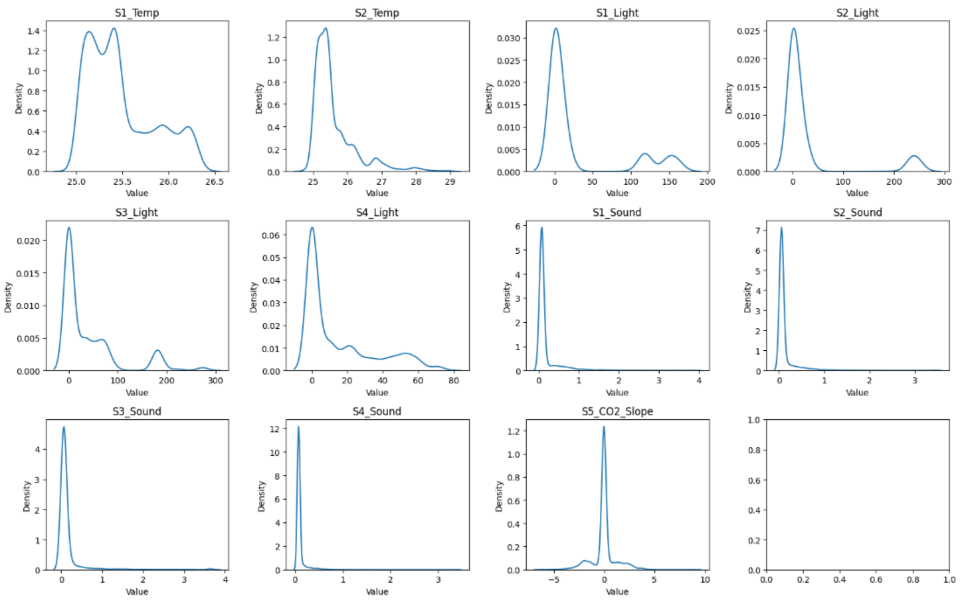


To address the issue of excessive outliers in the light sensor readings, we employed several transformations, including square root, cube root, and logarithmic transformations. These transformations are effective in reducing the influence of outliers by compressing the range of extreme values. By transforming the data using these methods, we aimed to achieve a more balanced distribution and mitigate the impact of outliers on our analysis and modeling efforts. The diagram below illustrates the box plot before and after applying the log transformation.

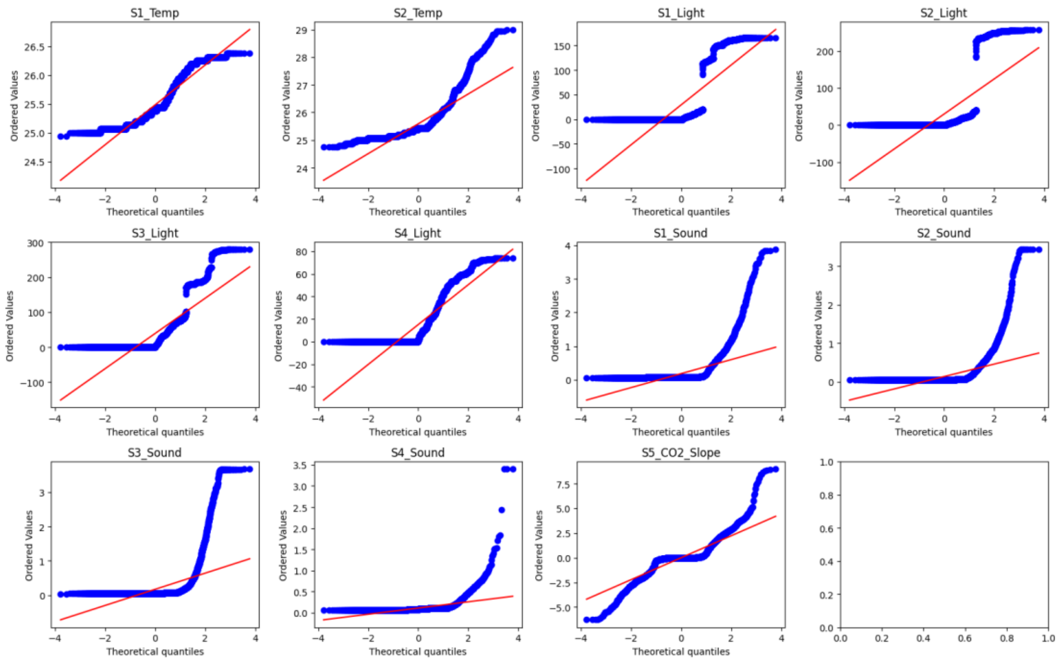


To assess the distribution of our features, particularly because certain models assume Gaussian distribution, we generated density plots for all the numerical features. The density plots visually represent the distribution of each feature, aiding in identifying any deviations from a Gaussian distribution.

Upon inspection, we observed that the sound sensor readings exhibit significant left skewness.

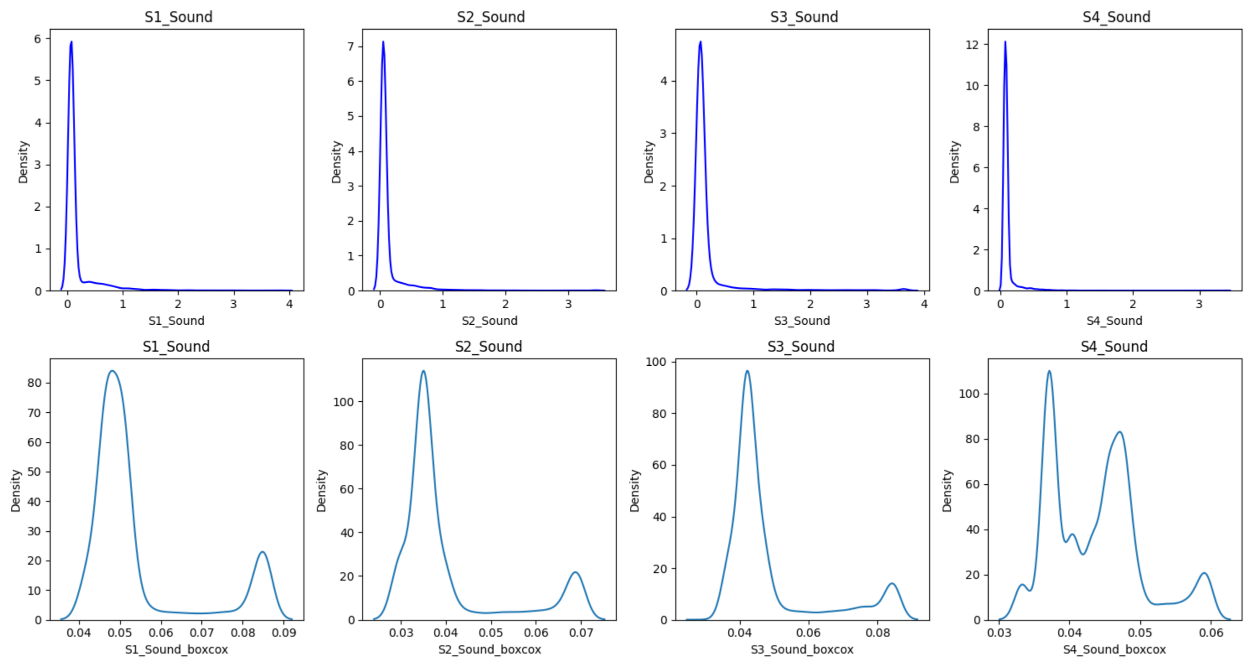


To delve deeper into the distribution characteristics, we utilized Q-Q plots, which provide a graphical comparison between the observed quantiles of a dataset and the quantiles of a theoretical Gaussian distribution. From these plots, we noted that 'S1_Temp' and 'S5_CO2_Slope' closely align with the theoretical line, suggesting a Gaussian distribution.



To address the non-Gaussian distribution of the sound sensor readings, we applied the Box-Cox transformation. Following this transformation, we observed that the transformed sound sensor features exhibited a closer approximation to a Gaussian distribution. However, despite this transformation, the 'S4_Sound' sensor readings continued to deviate significantly from a normal distribution.

Given this we made the decision to drop the 'S4_Sound' feature from further analysis. Further we also dropped the S2_temp as there were too many outliers.



Model Implementation

<u>Naive Bayes</u>

Naive Bayes was selected for its simplicity and effectiveness in handling categorical data. We assume the conditional independence of features (sensors) so that the effect of an attribute value on a given class is independent of the values of other attributes. Note, we already dropped features with a correlation greater than 0.9 during EDA.

For each class c, the model estimates the parameters of the Gaussian distribution: mean (μ), variance (σ^2), and the prior probability (π) uses Laplace smoothing. These parameters are computed below.

$$\mu_c = \frac{1}{N_c}\sum_{i=1}^{N_c} x_i,$$
$$\sigma_c^2 = \left(\frac{1}{N_c}\sum_{i=1}^{N_c}(x_i - \mu_c)^2\right) + \epsilon$$

N_c : the number of samples in class c

x_i : the feature vectors belonging to class c

ε : small constant added to avoid division by zero.

The prior probability π formula with Laplace Smoothing that we used:

$$\pi_c = \frac{N_c + 1}{N + K}$$

N: the total number of raining samples

K: the number of classes

For Laplace smoothing constant, we used 1 which ensured non-zero probabilities. To predict the class of a new sample x, the classifier calculates the posterior probability for each class c and selects the class with the highest probability.

$$P(c \mid x) \propto P(x \mid c)\pi_c$$
$$P(x \mid c) = \prod_{j=1}^{D} P(x_j \mid c)$$

D: the number of features

For each feature j in class c:

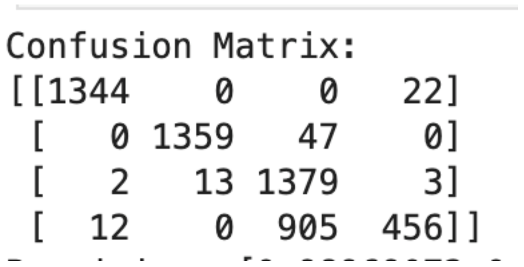
$$P(x_j \mid c) = \frac{1}{\sqrt{2\pi\sigma_{cj}^2}} \exp\left(-\frac{(x_j - \mu_{cj})^2}{2\sigma_{cj}^2}\right)$$

For numerical stability, the model calculates the log of the posterior probabilities:

$$\log P(c \mid x) = \log \pi_c + \sum_{j=1}^{D} \log P(x_j \mid c)$$

Finally, the class with the highest log posterior probability is chosen as the predicted class.

Naive Bayes Confusion Matrix without PCA-



```
Confusion Matrix:
[[1344    0    0   22]
 [   0 1359   47    0]
 [   2   13 1379    3]
 [  12    0  905  456]]
```

Support Vector Machines

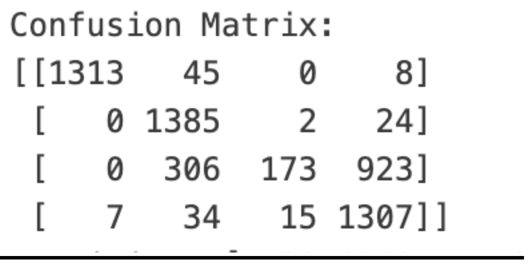
Support Vector Machine (SVM) is a powerful supervised learning algorithm used for classification tasks. It aims to find the optimal hyperplane that separates different classes by maximizing the margin between them. The basic idea is to identify the support vectors, which are the data points closest to the hyperplane.

SVM with Gradient Descent (GD) involves optimizing a cost function using gradient descent to find the parameters (weights) of the hyperplane. The cost function typically includes a regularization term to prevent overfitting and a hinge loss term that penalizes misclassifications. The GD formulas involve iteratively updating the weights to minimize the cost function until convergence.

$$J(w) = \lambda||w||2 + 1/n\sum ni{=}1\max(0, 1-yi(w.xi+b))$$

But the drawback of SVM is the computational cost making it extremely slow and time consuming on high amounts of data. Our dataset with around 30000 rows after scaling falls in the sweet spot where this operates.

SVM Confusion Matrix without PCA-



```
Confusion Matrix:
[[1313   45    0    8]
 [   0 1385    2   24]
 [   0  306  173  923]
 [   7   34   15 1307]]
```

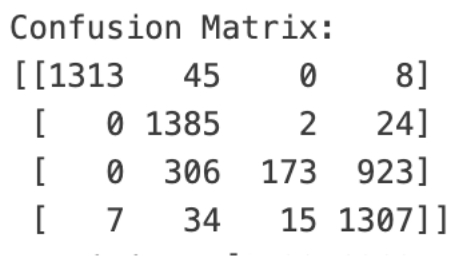
Logistic Regression

We have chosen Logistic Regression as one of the classifier models because our data is Independently and Identically Distributed. Also, we have transformed any outliers or dropped them. Moreover, we are using one-vs-all for which logistic regression is a good choice.

Cost Function:

$$-(y\log(p) + (1-y)\log(1-p))$$

Logistic Regression Confusion Matrix without PCA-



```
Confusion Matrix:
[[1313   45    0    8]
 [   0 1385    2   24]
 [   0  306  173  923]
 [   7   34   15 1307]]
```

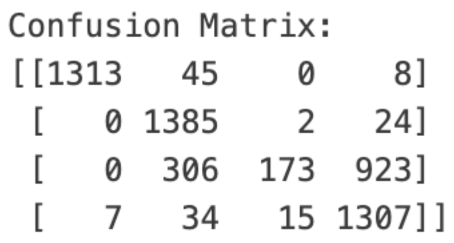
Dimensionality Reduction:

PCA

Principal Component Analysis (PCA) is a powerful statistical technique used to reduce the number of variables within a dataset while retaining most of its variance. In our project, applying PCA to our dataset enabled us to capture approximately 94% of the variance with just 7 principal components. Prior to implementing PCA, we conducted a thorough examination to ensure the suitability of our data for this method. This preliminary analysis included verifying the numerical nature of the columns, assessing the dimensionality of the dataset, and evaluating the linearity of the data values. Following these assessments and a comprehensive Exploratory Data Analysis (EDA), we confirmed that PCA was applicable to our transformed data.

Principal Component	Cumulative Variance Captured
PC 1	56.28%
PC 2	68.58%
PC 3	75.88%
PC 4	82.17%
PC 5	87.65%
PC 6	92.12%
PC 7	94.36%
PC 8	96.07%
PC 9	97.63%
PC 10	98.84%
PC 11	99.55%
PC 12	100.00%

Support Vector Machines With PCA-



```
Confusion Matrix:
[[1313   45    0    8]
 [   0 1385    2   24]
 [   0  306  173  923]
 [   7   34   15 1307]]
```

Conclusion

	Precision				Recall				
	Class				Class				
Model	0	1	2	3	0	1	2	3	**Accuracy**
Base Line Model	0.24	0.25	0.26	0.24	0.24	0.25	0.25	0.25	0.25
Naive Bayes	0.98	0.99	0.59	0.94	0.98	0.96	0.98	0.33	0.81
SVM without PCA	0.99	0.78	0.91	0.57	0.96	0.98	0.12	0.95	0.75
SVM with PCA	0.99	0.78	0.91	0.57	0.96	0.98	0.12	0.95	0.75
Logistic Regression	0.99	0.99	0.92	0.96	0.99	0.99	0.97	0.92	0.97

In this project, we evaluated the performance of several machine learning models for classification tasks. The baseline model, though simple, provides a reference point for comparison. However, more advanced models outperform the baseline across all metrics.

The Naive Bayes model incorporated Laplace smoothing to effectively handle zero-frequency issues, proving to be reasonably effective as evidenced by the evaluation metrics. This adjustment highlights the model's simplicity and reliance on basic assumptions, allowing Naive Bayes to successfully capture the underlying patterns in the data. The natural fit of this model with our dataset likely stems from the straightforward nature of the relationships between features, which align well with the model's assumptions. Overall, Naive Bayes demonstrates exceptional precision and recall for classes 0 and 1 but struggles with class 3, indicating potential class imbalance issues or the need for further feature engineering.

Support Vector Machines (SVM) both with and without Principal Component Analysis (PCA) exhibit similar precision and recall scores, indicating that dimensionality reduction does not significantly impact performance in this context. However, the overall accuracy of SVM models is lower compared to Naive Bayes and Logistic Regression.

Logistic Regression emerges as the top performer in terms of overall accuracy, precision, and recall across all classes. Its robust performance suggests its suitability for this classification task, especially in scenarios where interpretability is important.

In conclusion, while each model has its strengths and weaknesses, Logistic Regression emerges as the most suitable choice for this classification task, offering a balance between performance and interpretability.