“ ROOM OCCUPANCY PREDICTION ”

IE 7300 – FINAL PROJECT REPORT

GROUP 4

Kirthana Shri Chandra Sekar

Anjali Dayaram Kshirsagar

Neeraj Rangwani

Samarth Saxena

Abstract:

Effective room occupancy monitoring is important for optimizing energy consumption and ensuring space utilization efficiency in various domains such as smart buildings and hospitality. This project delves into estimating the exact number of occupants in a room utilizing Machine Learning techniques. The study involves a dataset comprising environmental sensor data, encompassing attributes like temperature, light, sound, CO2 and PIR levels. Machine Learning algorithms including Support Vector Machines (SVM), Logistic Regression, and Neural Networks are employed for predictive modeling. A range of performance metrics, including Accuracy, Precision, Recall, and F1 score were used to assess the effectiveness of our models. Principal Component Analysis (PCA) was also implemented to compare the model performance after dimensionality reduction. The outcomes from the experiments revealed a high precision score of 98.7% in predicting the room's occupancy count. This project gives the significance of Machine Learning in optimizing space usage, contributing to smarter infrastructure planning and resource management.

Introduction:

In the field of building management, real-time occupancy data is important for optimizing intelligent HVAC systems and advanced lighting, contributing significantly to energy conservation and occupant comfort. Leveraging Internet of Things (IoT) sensors for environmental monitoring, this project explores the potential of classical Machine Learning (ML) and Neural Networks to accurately estimate room occupancy. The challenge lies in harnessing IoT data and ML techniques for an economical and non-intrusive solution that aligns with energy efficiency and occupant well-being objectives. The primary project goal is to employ ML for precise room occupancy estimation, reducing energy consumption and promoting sustainability. Specific objectives include identifying the best-performing model, determining influential sensor features, and interpreting models for insights into the role of sensor data in occupancy estimation. This research contributes to advancing sustainable building practices and energy conservation.

Data Description:

Source : The dataset used in this study originates from the UCI Machine Learning repository and is credited to Adarsh Pal Singh, Vivek Jain, Sachin Chaudhari, Frank Alexander Kraemer, Stefan Werner, and Vishal Garg. The dataset is prominently featured in the research paper titled "Machine Learning-Based Occupancy Estimation Using Multivariate Sensor Nodes," presented at the 2018 IEEE Globecom Workshops (GC Wkshps).
The dataset can be accessed at the following link: UCI Machine Learning Repository - Room Occupancy Estimation

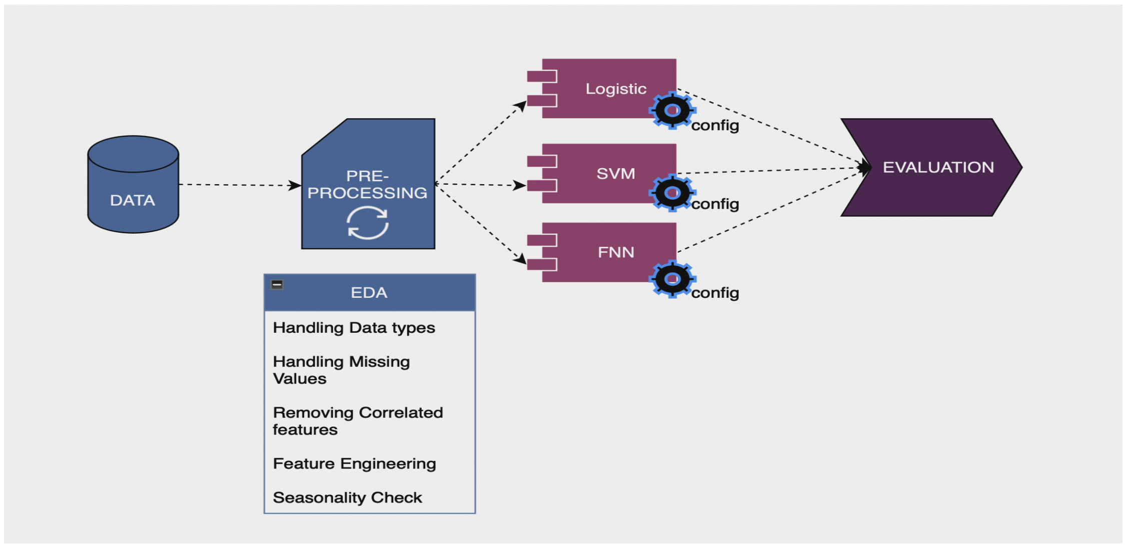
The data was gathered over a 7-day duration, during which the heating, ventilation, and air conditioning (HVAC) systems in the designated building space remained inactive. Occupancy levels fluctuated between 0 and 3 individuals, providing insights into the dynamics of the environment under real-world conditions. Measurements were recorded every 30 seconds, facilitated by wireless transceivers.

The dataset encompasses **10,129** instances with **19** distinct features, offering a comprehensive perspective on the environment of the room. The features include time series data on temperature, light, CO2 levels, PIR (Passive Infrared Rays), and sound, systematically collected from sensor nodes strategically positioned in a 6m x 4.6m room.

Methods:

The methodology to address the occupancy prediction problem involves a three-step process.

1. A comprehensive data preprocessing and exploratory data analysis (EDA) stage is implemented, encompassing cleaning, outlier handling, feature engineering, and standardization. This is crucial for ensuring data integrity and preparing features for modeling.
2. Data was split into Training (60%), Validation (20%) and Test sets (20%). Standardization and sampling techniques were applied.
3. A diverse set of models, including Logistic Regression, Support Vector Machines (SVM), and Fully Connected Feedforward Neural Networks (FNN), are employed. Logistic Regression and SVM were also tested with PCA dataframe to check performance. These models undergo rigorous training, validation, and fine-tuning processes.
4. Lastly, the models are tested using appropriate evaluation metrics such as accuracy, precision, recall, F1-score to assess their performance and identify the most effective solution for accurate occupancy prediction.



Data Understanding :

➢ 19 features :	Continuous numerical features: 14
	Categorical features: 2
	Datetime features: 2
	Target Variable: 1
➢ 4 classes :	Room Occupancy count: [0,1,2,3]

Understanding the Models:

Logistic Regression:
Logistic regression is a widely employed statistical model for binary classification tasks. It utilizes the sigmoid function to map a linear combination of input features to a probability between 0 and 1, representing the likelihood of the positive class. By optimizing parameters through gradient descent and minimizing the difference between predicted probabilities and actual labels, the model learns associations iteratively. It is versatile in nature, as logistic regression extends seamlessly to multi-class classification (using softmax function). Regularization techniques can be incorporated to enhance generalization and prevent overfitting. In essence, logistic regression with gradient descent is a straightforward yet potent algorithm, iteratively learning the association between input features and class probabilities in binary and multi-class scenarios.

SVM:
Support Vector Machines (SVM) are a class of powerful classification algorithms. At their core, SVMs work by finding a hyperplane, acting as a decision boundary, that effectively separates data points into two categories in binary classification tasks. The choice between Hard Margin SVM, which aims for a perfect separation, and Soft Margin SVM, which allows for controlled misclassifications through the parameter C, depends on the nature of the data. The optimization process involves tuning parameters such as C and alpha values, where C balances the trade-off between a smooth decision boundary and permissible misclassifications, and alpha values determine the significance of individual data points in defining the hyperplane. For multiclass classification, SVMs employ the one-vs-all approach, training separate binary classifiers for each class. In essence, SVMs offer a flexible and robust framework for classification tasks, utilizing hyperplanes and various parameters to achieve accurate and adaptable decision boundaries.

FNN:
A Feedforward Neural Network (FNN) is a method in neural network architecture employed for classification or regression tasks. It operates through layers of interconnected neurons, where information flows in a unidirectional manner—from input to output. The fundamental concept underlying FNN is that of a fully connected neural network, meaning each neuron in one layer is connected to every neuron in the subsequent layer. To enhance the learning process, FNN employs

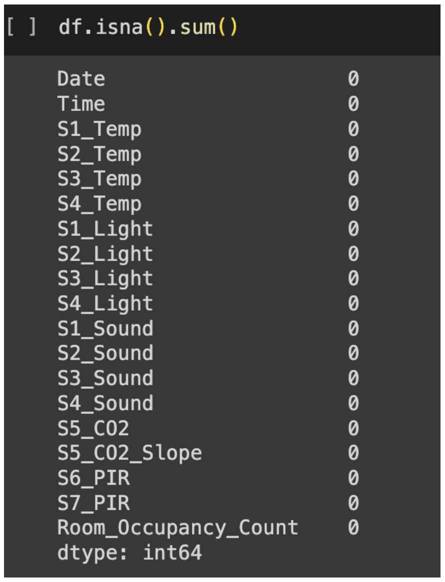
backpropagation, a mechanism that updates the model based on the error calculated during the prediction stage, utilizing the chain rule for efficient adjustments. Additionally, to expedite computations, the mini-batch technique can be utilized, which processes subsets of data, contributing to the network's overall efficiency.

Exploratory Data Analysis (EDA):

EDA is a crucial step in understanding and summarizing the main characteristics of a dataset using statistical and visualization techniques. EDA helps in identifying missing values, outliers, and understanding the distribution of variables, enabling better-informed decisions in subsequent modeling or analysis tasks.

We initiated our analysis by importing essential libraries and loading the dataset, investigating its dimensions through the utilization of the ".shape" function. It is also a good practice to know the number of records and columns and their corresponding data types (.dtypes), along with finding whether they contain null values or not.

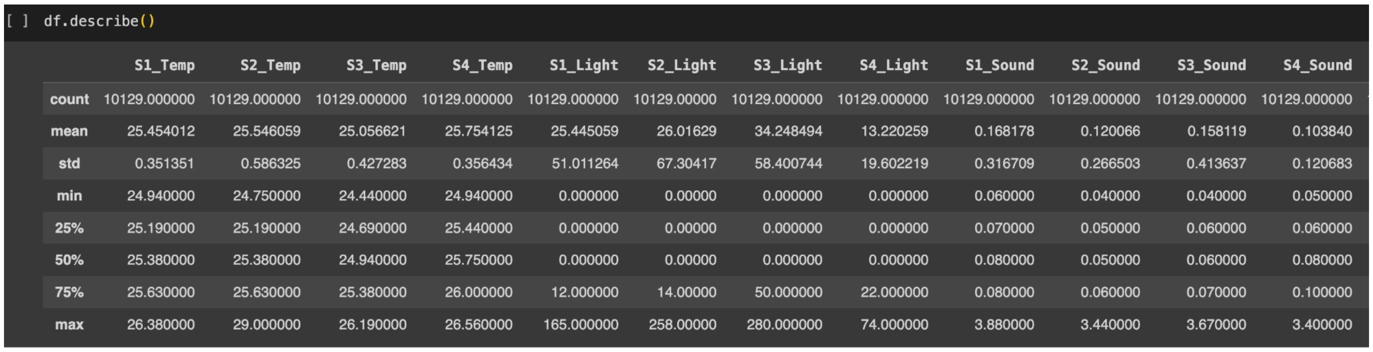
No missing values were present in the dataset -



```
[ ]  df.isna().sum()

     Date                    0
     Time                    0
     S1_Temp                 0
     S2_Temp                 0
     S3_Temp                 0
     S4_Temp                 0
     S1_Light                0
     S2_Light                0
     S3_Light                0
     S4_Light                0
     S1_Sound                0
     S2_Sound                0
     S3_Sound                0
     S4_Sound                0
     S5_CO2                  0
     S5_CO2_Slope            0
     S6_PIR                  0
     S7_PIR                  0
     Room_Occupancy_Count    0
     dtype: int64
```

We used the .describe() function to understand distribution and statistical information for each numerical column in our dataset.

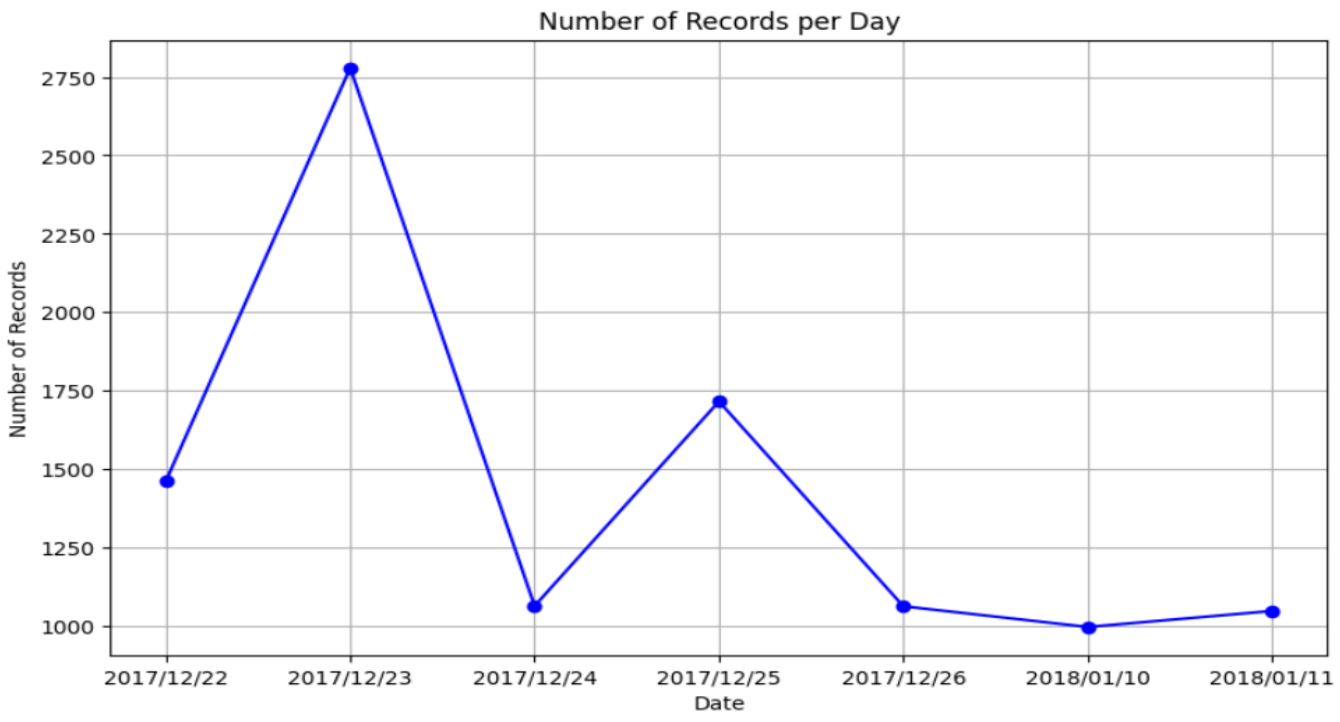


```
[ ]  df.describe()
```

	S1_Temp	S2_Temp	S3_Temp	S4_Temp	S1_Light	S2_Light	S3_Light	S4_Light	S1_Sound	S2_Sound	S3_Sound	S4_Sound
count	10129.000000	10129.000000	10129.000000	10129.000000	10129.000000	10129.00000	10129.000000	10129.000000	10129.000000	10129.000000	10129.000000	10129.000000
mean	25.454012	25.546059	25.056621	25.754125	25.445059	26.01629	34.248494	13.220259	0.168178	0.120066	0.158119	0.103840
std	0.351351	0.586325	0.427283	0.356434	51.011264	67.30417	58.400744	19.602219	0.316709	0.266503	0.413637	0.120683
min	24.940000	24.750000	24.440000	24.940000	0.000000	0.00000	0.000000	0.000000	0.060000	0.040000	0.040000	0.050000
25%	25.190000	25.190000	24.690000	25.440000	0.000000	0.00000	0.000000	0.000000	0.070000	0.050000	0.060000	0.060000
50%	25.380000	25.380000	24.940000	25.750000	0.000000	0.00000	0.000000	0.000000	0.080000	0.050000	0.060000	0.080000
75%	25.630000	25.630000	25.380000	26.000000	12.000000	14.00000	50.000000	22.000000	0.080000	0.060000	0.070000	0.100000
max	26.380000	29.000000	26.190000	26.560000	165.000000	258.00000	280.000000	74.000000	3.880000	3.440000	3.670000	3.400000

Our exploration proceeded with the following insightful visualizations:

➢ We created a Line chart to display the distribution of records across the seven distinct date values present in the dataset.

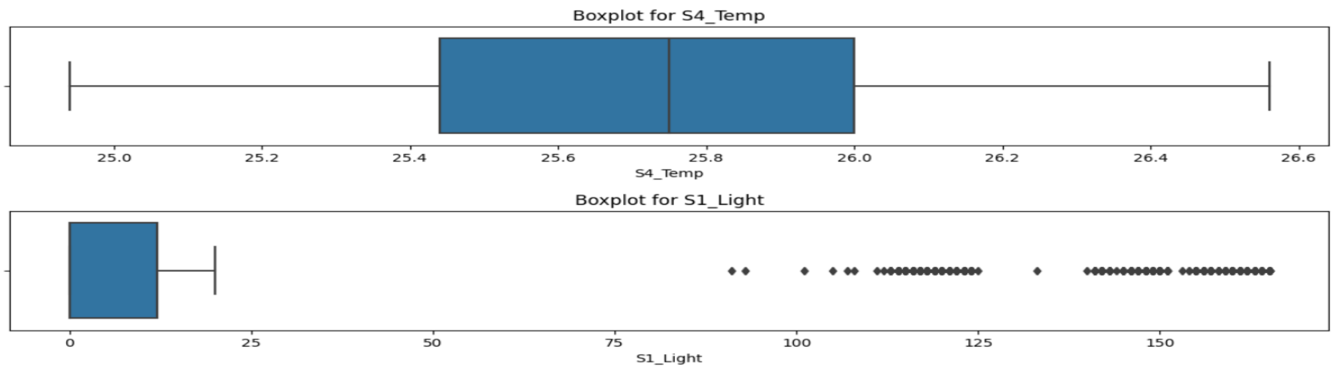


This plot tells us that there is no seasonality effect since only 7 unique days data is present in the dataset.

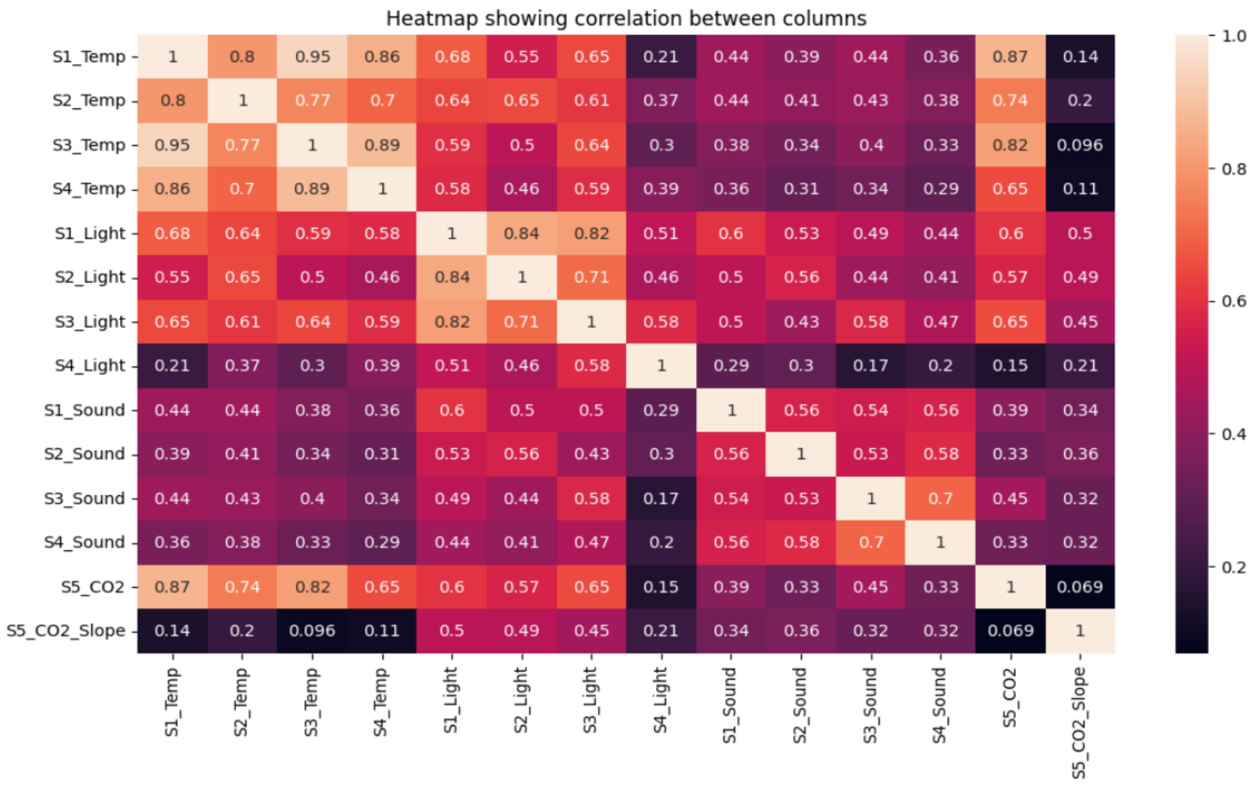
➢ Subsequently, we engineered a new "Date_time" column by combining the date and time columns, categorizing room occupancy during different time segments ('Morning,' 'Afternoon,' 'Evening,' 'Night') and we converted this "Time_of_Day" column to categorical form using LabelEncoder. We then dropped the Date and Time columns since they violate the independence assumption due to their temporal nature. By excluding these features, we aim to adhere to the

IID assumption and facilitate the application of ML algorithms, which generally perform optimally when this assumption holds true.

➢ The box plot is a standardized way of displaying the distribution of data based on the five number summary: Minimum, First quartile, Median, Third quartile, Maximum. These are instrumental in finding the outliers in the dataset (if any) as well as the range of each column.



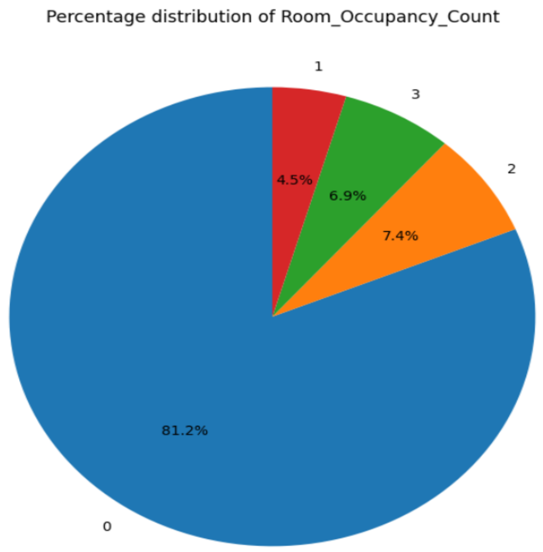
➢ Feature correlation analysis, conducted through pandas' ".corr()" function and visualized via a heatmap in seaborn, aimed to eliminate correlated variables, thereby enhancing model performance. We removed the categorical features and the target variable before performing the correlation analysis.



	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
S1_Temp	1	0.8	0.95	0.86	0.68	0.55	0.65	0.21	0.44	0.39	0.44	0.36	0.87	0.14
S2_Temp	0.8	1	0.77	0.7	0.64	0.65	0.61	0.37	0.44	0.41	0.43	0.38	0.74	0.2
S3_Temp	0.95	0.77	1	0.89	0.59	0.5	0.64	0.3	0.38	0.34	0.4	0.33	0.82	0.096
S4_Temp	0.86	0.7	0.89	1	0.58	0.46	0.59	0.39	0.36	0.31	0.34	0.29	0.65	0.11
S1_Light	0.68	0.64	0.59	0.58	1	0.84	0.82	0.51	0.6	0.53	0.49	0.44	0.6	0.5
S2_Light	0.55	0.65	0.5	0.46	0.84	1	0.71	0.46	0.5	0.56	0.44	0.41	0.57	0.49
S3_Light	0.65	0.61	0.64	0.59	0.82	0.71	1	0.58	0.5	0.43	0.58	0.47	0.65	0.45
S4_Light	0.21	0.37	0.3	0.39	0.51	0.46	0.58	1	0.29	0.3	0.17	0.2	0.15	0.21
S1_Sound	0.44	0.44	0.38	0.36	0.6	0.5	0.5	0.29	1	0.56	0.54	0.56	0.39	0.34
S2_Sound	0.39	0.41	0.34	0.31	0.53	0.56	0.43	0.3	0.56	1	0.53	0.58	0.33	0.36
S3_Sound	0.44	0.43	0.4	0.34	0.49	0.44	0.58	0.17	0.54	0.53	1	0.7	0.45	0.32
S4_Sound	0.36	0.38	0.33	0.29	0.44	0.41	0.47	0.2	0.56	0.58	0.7	1	0.33	0.32
S5_CO2	0.87	0.74	0.82	0.65	0.6	0.57	0.65	0.15	0.39	0.33	0.45	0.33	1	0.069
S5_CO2_Slope	0.14	0.2	0.096	0.11	0.5	0.49	0.45	0.21	0.34	0.36	0.32	0.32	0.069	1

High correlation among features S1_temp, S3_temp, and S4_temp, suggesting their similar impact on the target variable, thus warranting the removal of two features (S1_temp and S3_temp) to streamline the dataset's dimensionality.

➢ Visualization of the class distribution via a pie chart for our categorical target variable "Room_Occupancy_Count".

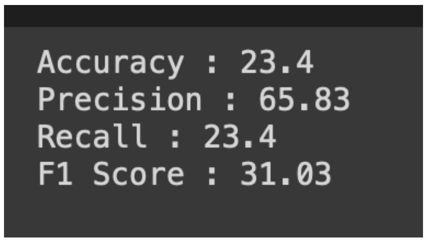


The plot showed that there was a class imbalance in our dataset since more than 80% of the records belong to '0' class. To address this imbalance, techniques such as Undersampling and Oversampling are employed in the train dataset to ensure equal importance across all classes for predicting the target variable.

Results:

Baseline Model:

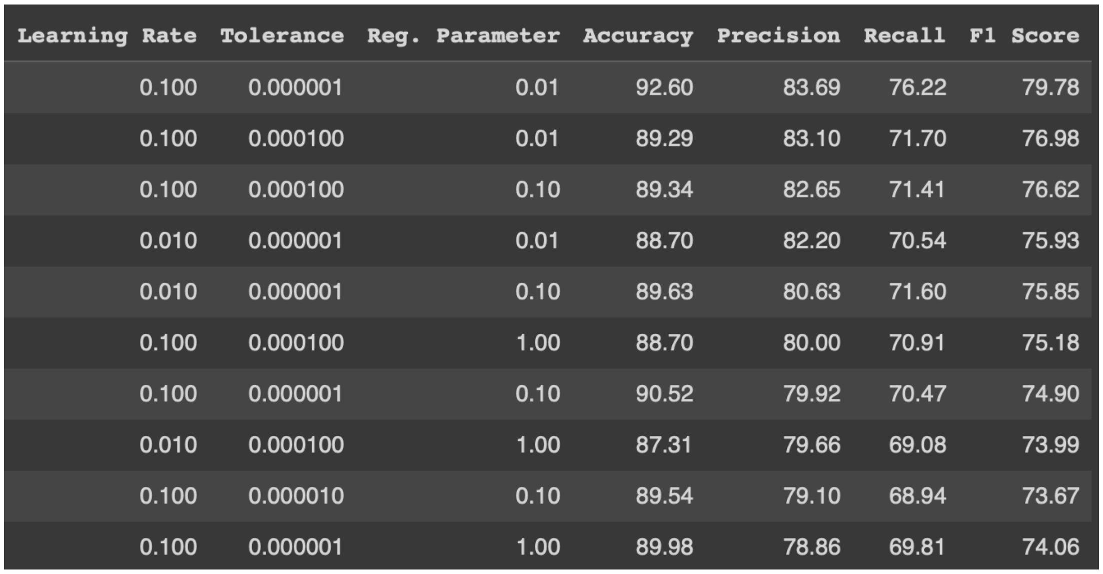
We created a baseline model that randomly gives class output value between 0 and 3. The results of this model on the test set were:



```
Accuracy : 23.4
Precision : 65.83
Recall : 23.4
F1 Score : 31.03
```

Logistic Regression:
Used softmax function to get probability values for each class, and whichever has the highest value means the record belongs to that class. Performed grid search with different range of hyperparameter values - Learning rate (0.001, 0.01, 0.1), Tolerance (0.0001, 0.00001, 0.000001) and Regularization parameter (0.01, 0.1, 1) to obtain best performing model. With PCA, we used 8 principal components for our dataframe since they explained > 85% of cumulative variance.

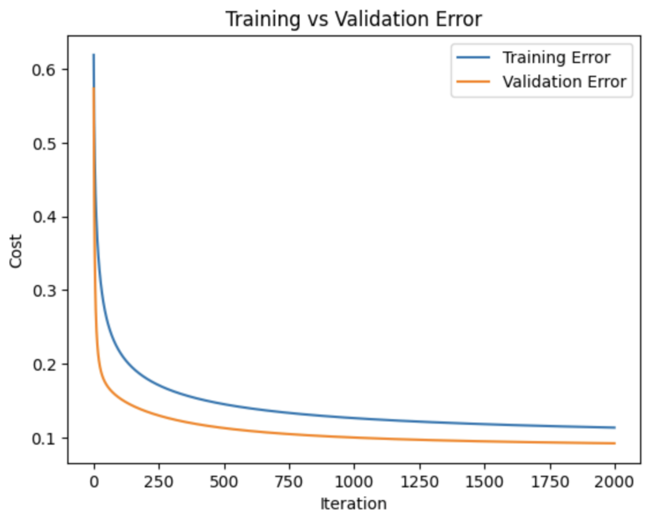
With PCA:
Top 10 hyperparameter combination results sorted in the descending order of precision



Learning Rate	Tolerance	Reg. Parameter	Accuracy	Precision	Recall	F1 Score
0.100	0.000001	0.01	92.60	83.69	76.22	79.78
0.100	0.000100	0.01	89.29	83.10	71.70	76.98
0.100	0.000100	0.10	89.34	82.65	71.41	76.62
0.010	0.000001	0.01	88.70	82.20	70.54	75.93
0.010	0.000001	0.10	89.63	80.63	71.60	75.85
0.100	0.000100	1.00	88.70	80.00	70.91	75.18
0.100	0.000001	0.10	90.52	79.92	70.47	74.90
0.010	0.000100	1.00	87.31	79.66	69.08	73.99
0.100	0.000010	0.10	89.54	79.10	68.94	73.67
0.100	0.000001	1.00	89.98	78.86	69.81	74.06

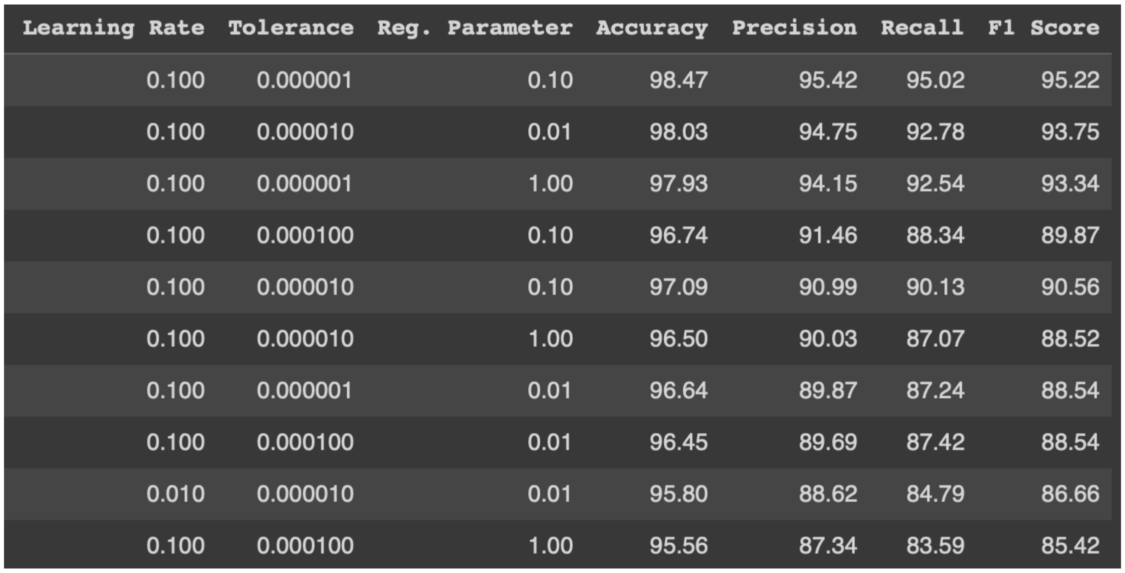
Highest precision score achieved with PCA is - 84%

Corresponding Training vs Validation error plot for highest precision value model



Without PCA:

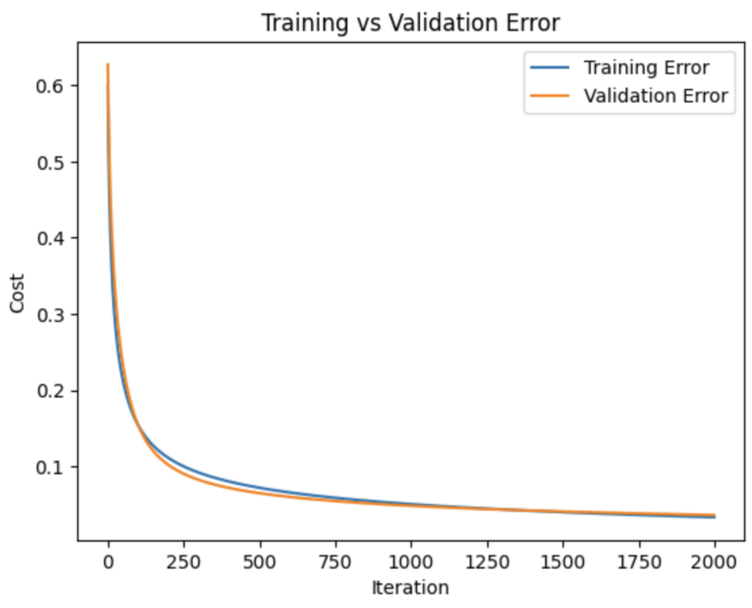
Top 10 hyperparameter combination results sorted in the descending order of precision



Learning Rate	Tolerance	Reg. Parameter	Accuracy	Precision	Recall	F1 Score
0.100	0.000001	0.10	98.47	95.42	95.02	95.22
0.100	0.000010	0.01	98.03	94.75	92.78	93.75
0.100	0.000001	1.00	97.93	94.15	92.54	93.34
0.100	0.000100	0.10	96.74	91.46	88.34	89.87
0.100	0.000010	0.10	97.09	90.99	90.13	90.56
0.100	0.000010	1.00	96.50	90.03	87.07	88.52
0.100	0.000001	0.01	96.64	89.87	87.24	88.54
0.100	0.000100	0.01	96.45	89.69	87.42	88.54
0.010	0.000010	0.01	95.80	88.62	84.79	86.66
0.100	0.000100	1.00	95.56	87.34	83.59	85.42

Highest precision score achieved with PCA is - 95%

Corresponding Training vs Validation error plot for highest precision value model

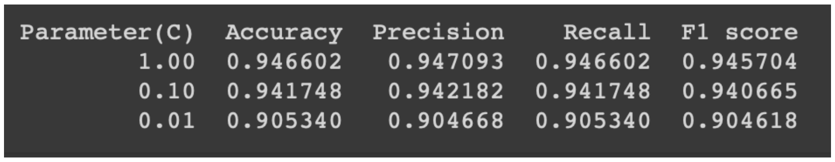


The error pattern for the training and validation set was almost the same showing there was no overfitting issue in our model. Also, the Logistic Regression model performed much better without PCA, which shows the importance of features in our data and the information they provide.

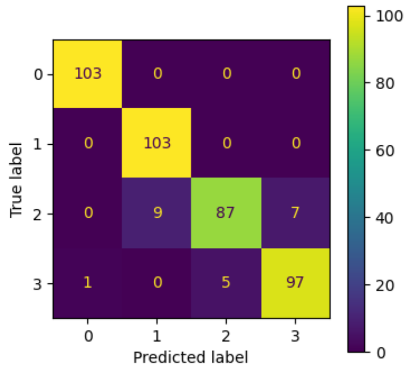
SVM:
The model is initialized with a regularization parameter C, and its fit method takes input features X and corresponding labels y. The optimization process aims to maximize the dual SVM objective using the Sequential Least Squares Quadratic Programming (SLSQP) method. The resulting optimized alphas are utilized to compute the decision boundary parameters, including the weight vector w and the bias term b. Support vectors, crucial for defining the decision boundary, are identified based on the optimized alphas exceeding a small threshold epsilon. The model stores the key attributes, such as the optimized alphas, support vectors, support labels, weight vector, and bias term, providing a comprehensive representation of the trained Soft Margin SVM for subsequent classification tasks.

Two independent Soft Margin SVM models are trained: one without PCA on the original feature space and another with PCA-transformed data. Both models are trained for different C values, providing a comprehensive exploration of the impact of regularization and PCA transformation on the SVM's learning process and subsequent classification performance.

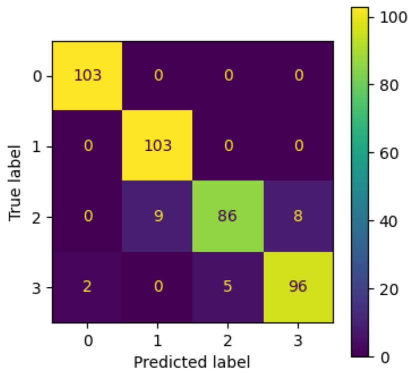
<u>Without PCA</u>:
Results sorted in the descending order of precision


```
Parameter(C)  Accuracy  Precision    Recall  F1 score
        1.00  0.946602   0.947093  0.946602  0.945704
        0.10  0.941748   0.942182  0.941748  0.940665
        0.01  0.905340   0.904668  0.905340  0.904618
```

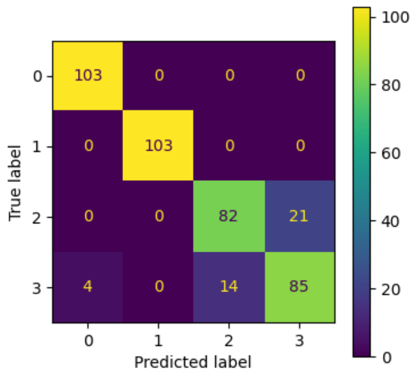
C =1.00



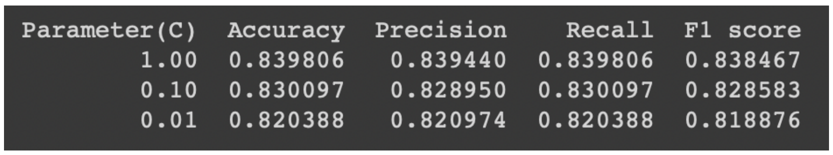
C=0.10



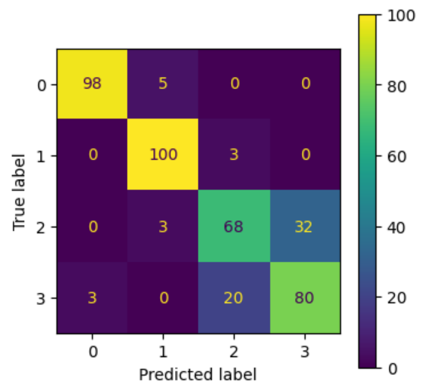
C=0.01



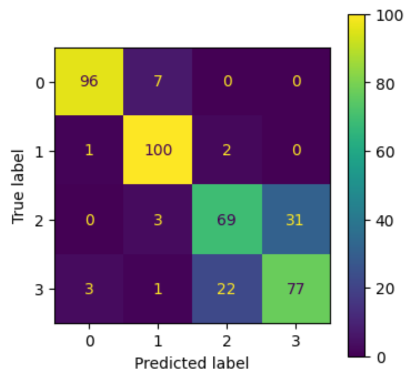
<u>With PCA</u>:
Results sorted in the descending order of precision


```
Parameter(C)  Accuracy  Precision    Recall  F1 score
        1.00  0.839806   0.839440  0.839806  0.838467
        0.10  0.830097   0.828950  0.830097  0.828583
        0.01  0.820388   0.820974  0.820388  0.818876
```

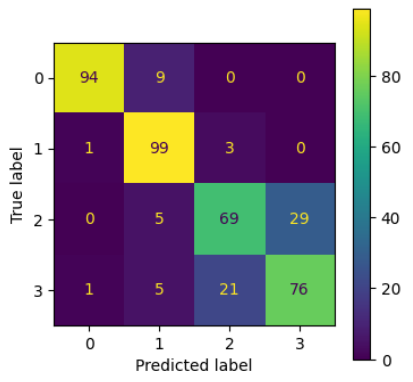
C =1.00



C=0.10



C=0.01



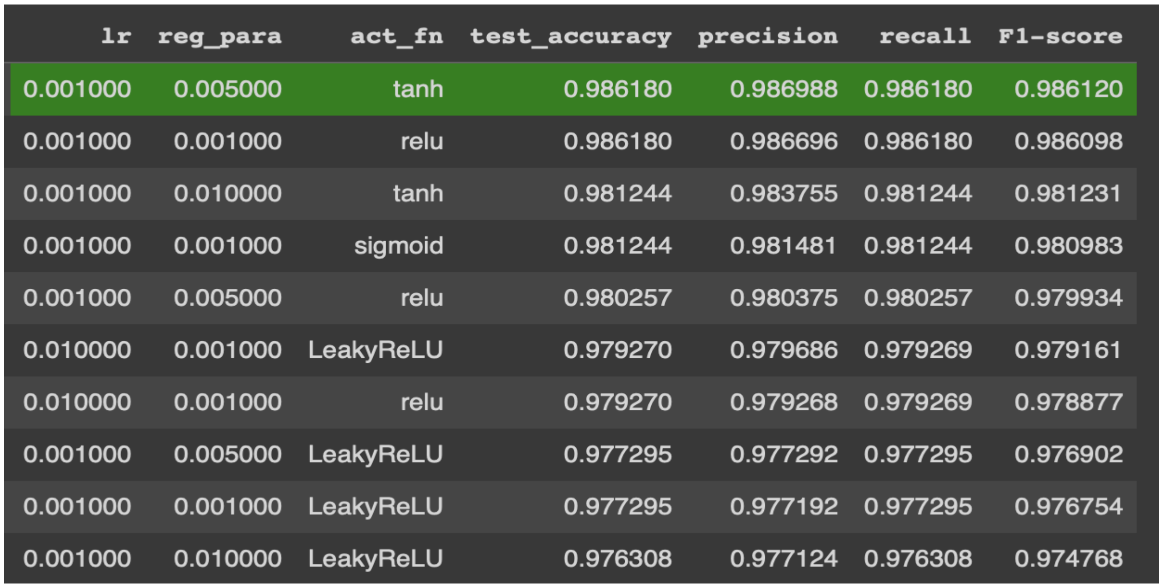
The model without PCA demonstrates superior performance compared to the PCA-transformed model. Without PCA, the SVM achieves higher precision, recall, accuracy, and F1-score, particularly at C=1.0. The PCA transformation leads to a noticeable decline in precision and overall model performance. The impact of regularization (C values) is consistent in both scenarios, with C=1.0 consistently providing optimal results. The choice between using PCA or not depends on the dataset characteristics, with the non-PCA model showcasing better classification performance in this context.

FNN:

The Neural Networks model function defines a 3-layer Feedforward Neural Network (FNN) with 16 neurons in the input layer, 15 neurons in the hidden layer (activated by act_fn), and 4 neurons in the softmax-activated output layer for multiclass classification. Training employs the Adam optimizer with different learning rates (lr- 0.01,0.001,0.0001), activation functions ('relu', 'sigmoid', 'tanh', 'LeakyReLU') and L2 regularization (0.01,0.001,0.0001) to prevent overfitting. The model undergoes 20 epochs with a batch size of 32, utilizing sparse categorical cross-entropy loss and accuracy as the evaluation metric for each configuration.

The printed classification report offers a detailed summary, including metrics for each class. Weighted computation considers class imbalance, providing a comprehensive understanding of the model's predictive capabilities and generalization performance. Hyperparameter adjustments, such as learning rate and regularization strength, offer optimization possibilities.

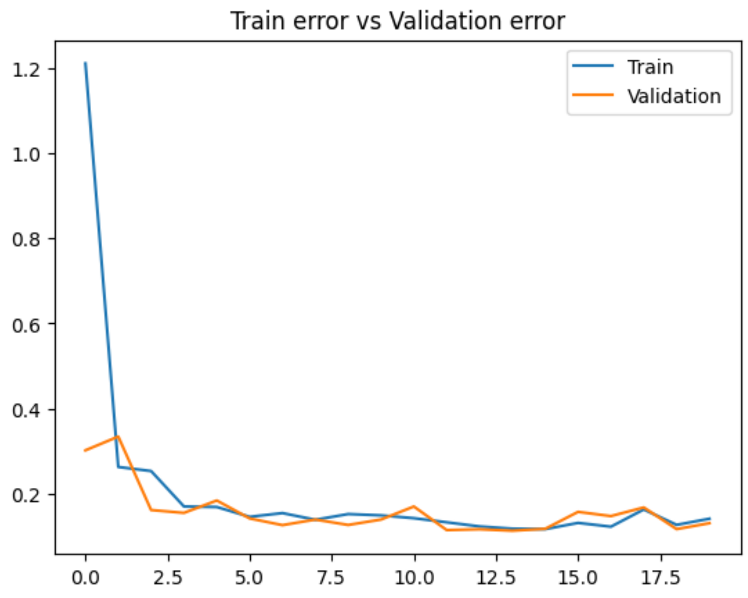
Top 10 hyperparameter combination results sorted in the descending order of precision:



lr	reg_para	act_fn	test_accuracy	precision	recall	F1-score
0.001000	0.005000	tanh	0.986180	0.986988	0.986180	0.986120
0.001000	0.001000	relu	0.986180	0.986696	0.986180	0.986098
0.001000	0.010000	tanh	0.981244	0.983755	0.981244	0.981231
0.001000	0.001000	sigmoid	0.981244	0.981481	0.981244	0.980983
0.001000	0.005000	relu	0.980257	0.980375	0.980257	0.979934
0.010000	0.001000	LeakyReLU	0.979270	0.979686	0.979269	0.979161
0.010000	0.001000	relu	0.979270	0.979268	0.979269	0.978877
0.001000	0.005000	LeakyReLU	0.977295	0.977292	0.977295	0.976902
0.001000	0.001000	LeakyReLU	0.977295	0.977192	0.977295	0.976754
0.001000	0.010000	LeakyReLU	0.976308	0.977124	0.976308	0.974768

The models, employing a learning rate of 0.001 and regularization parameters ranging from 0.001 to 0.005, and all activation functions demonstrated very good performance, indicating their suitability for the classification.

Train vs Validation error graph of 20 epochs for model with highest precision score:
(Learning rate- 0.001, Regularization parameter - 0.005, Activation function - tanh)



Stable training and validation errors with minimal fluctuations after the first epoch, signified a well-balanced learning model. This effectiveness in learning from training data while generalizing to new data yielded 98% accuracy on the test set.

Discussion:

Interpretation:

The findings indicate the efficacy of machine learning in accurately estimating room occupancy. All models performed much better than the random baseline model for all evaluation metrics. The best evaluation metric scores for each model is shown below.

Model	Precision	Accuracy	Recall	F1 Score
Logistic Regression without PCA	95.42%	98.47%	95.02%	95.22%
Logistic Regression with PCA	83.69%	92.60%	76.22%	79.78%
SVM without PCA	94.71%	94.66%	94.66%	94.57%
SVM with PCA	83.94%	83.98%	83.98%	83.85%
Neural Networks	98.7%	98.62%	98.62%	98.61%

The high precision scores show the models' ability to understand occupancy patterns based on available sensor data. This suggests the models' strong predictive ability, promising reliability in real-time occupancy estimation tasks.

Limitations:

Although the models exhibited high accuracy, they might face limitations in scenarios involving sudden or irregular changes in occupancy patterns. Our study focused on a specific dataset and environmental conditions which could limit the models' adaptability to diverse settings or unforeseen circumstances.

Future Scope:

In future, we aim to use cross validation for the SVM model which would help to generalize better on the data which was not possible currently due to high time complexity and computational factors.