

**Machine Learning**

**10204350  
H/618/7438**

**Section (1)**

**Assignment 2 ML Project**

**Submitted to**

Dr. Rami Al Ouran

**Submitted on**

January 30th, 2024

**Submitted by**

Marwan Tarek Shafiq Al Farah

**Student ID**

21110011

**Fall 2023 – 2024**

**Table of Content**

[***Introduction*** 3](#_Toc156143210)

[Problem Addressed and its Importance 3](#_Toc156143211)

[Dataset: Source, Collection Method, Attributes, Size, and Domain 3](#_Toc156143212)

[Learning Problem Being Solved 6](#_Toc156143213)

[Data Preprocessing 6](#_Toc156143214)

[***Methods*** 8](#_Toc156143215)

[Model Selection and Justification 8](#_Toc156143216)

[Testing Machine Learning Applications 10](#_Toc156143217)

[Machine Learning Algorithms Used 10](#_Toc156143218)

[***Evaluation*** 16](#_Toc156143219)

[Performance Measures Used for Model Evaluation 16](#_Toc156143220)

[Rationale for Using Specific Performance Metrics 16](#_Toc156143221)

[Model Enhancements Based on Performance Measures 17](#_Toc156143222)

[***Results and Discussion*** 19](#_Toc156143223)

[Result Reliability and Algorithm Fit 19](#_Toc156143224)

[Effectiveness of Algorithm Applications 21](#_Toc156143225)

[Algorithm Strengths and Weaknesses 22](#_Toc156143226)

[Future Enhancements and Project Limitations 24](#_Toc156143227)

[***References*** 26](#_Toc156143228)

# ***Introduction***

Problem Addressed and its Importance (Singh et al., 2018)

In this assignment, we will address the problem of accurately estimating room occupancy using a collection of sensors, which is an important issue in the context of smart building management, since accurate occupancy prediction can significantly enhance energy efficiency in buildings. By using these predictions, it becomes possible to optimize heating, ventilation, air conditioning, and lighting systems, leading to reduced energy consumption and cost savings, which is not only positive economically, but also environmentally significant, as buildings account for a substantial portion of energy use globally, meaning that this approach contributes to creating smarter, more sustainable building management systems.

Dataset: Source, Collection Method, Attributes, Size, and Domain (Singh et al., 2018)

* **Source:** The dataset originates from the research study “Machine Learning-based Occupancy Estimation Using Multivariate Sensor Nodes,” which focuses on estimating room occupancy using a variety of sensors.
* **Collection Method:** Data was collected using multiple low-cost, non-intrusive sensors in a star configuration, capturing environmental conditions over 4 days (based on the research paper) / 7 days (based on the dates in the dataset).
* **Attributes:** The dataset contains 19 columns, which include:
* **Date and Time:** Representing the date and the timestamp of the data collection.
* **S1\_Temp to S4\_Temp:** Temperature readings from four different sensors.
* **S1\_Light to S4\_Light:** Illumination levels from four different sensors.
* **S1\_Sound to S4\_Sound:** Sound levels from four different sensors.
* **S5\_CO2:** CO2 concentration levels.
* **S5\_CO2\_Slope:** The rate of change of CO2 concentration.
* **S6\_PIR and S7\_PIR:** Motion detection readings from two Passive Infrared (PIR) sensors.
* **Room\_Occupancy\_Count:** The count of occupants in the room, which is the target variable for occupancy estimation.
* **Size:** The dataset comprises 10,129 rows, each representing a unique set of sensor readings along with the corresponding occupancy count, along with 18 columns for the features and 1 for the target variable, meaning that the dataset is 10,129 x 19.
* **Domain:** This dataset is within the domain of environmental monitoring and smart building management, specifically in the field of using sensor data for estimating the occupancy of indoor spaces.

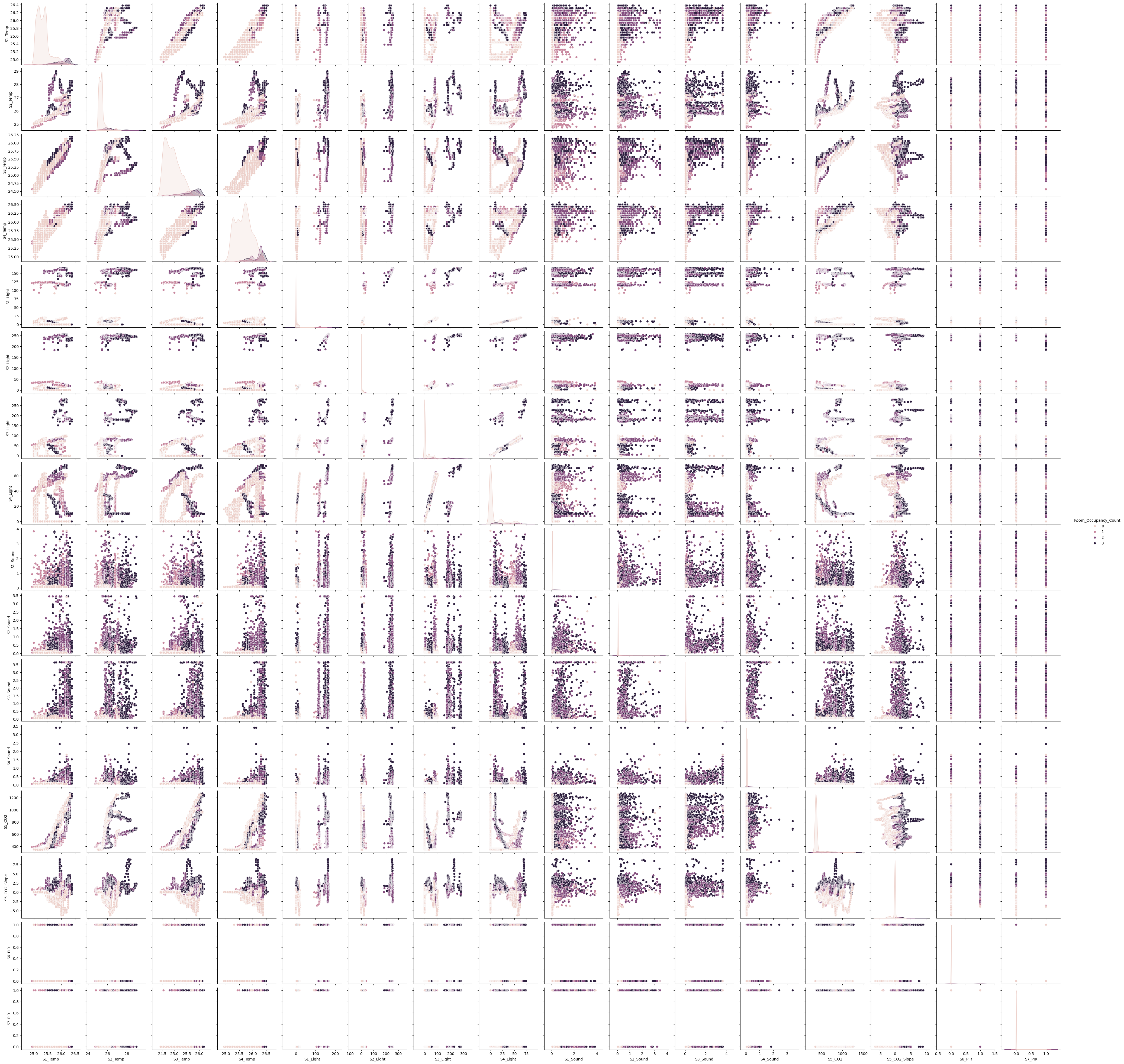
**EDA**

With the use of df.info() and df.describe(), statistical summaries for each sensor’s readings, including mean, standard deviation, minimum and maximum values, and various percentiles, provide a foundational understanding of the data’s distribution and variability. Also, they also helped us identify the data types of each feature, along with checking for any null values.

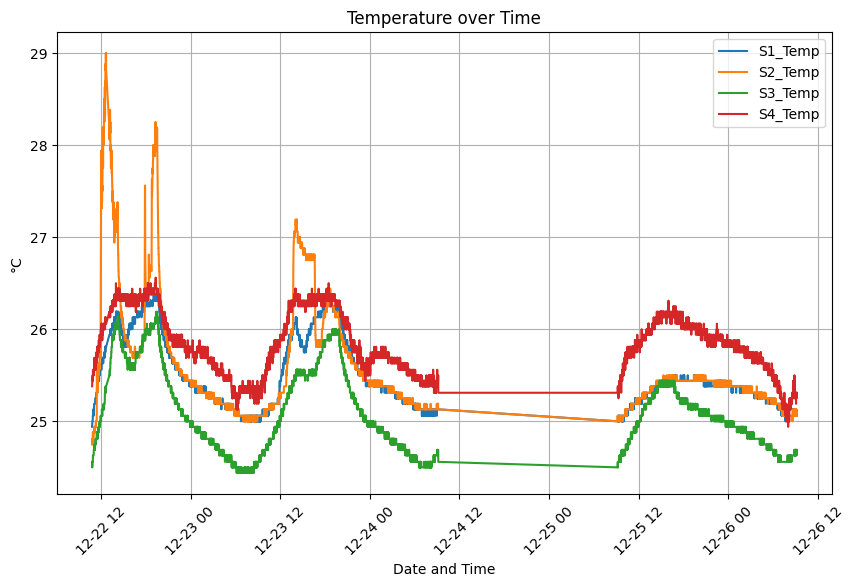
A graph with different colored bars

Description automatically generated

We used a bar chart to visualize out target variable “Room Occupancy Count”, and found out that the dataset is imbalanced, and that most data points are when the count was 0.

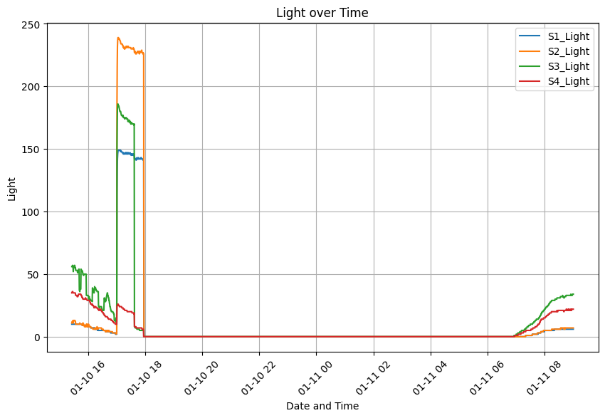
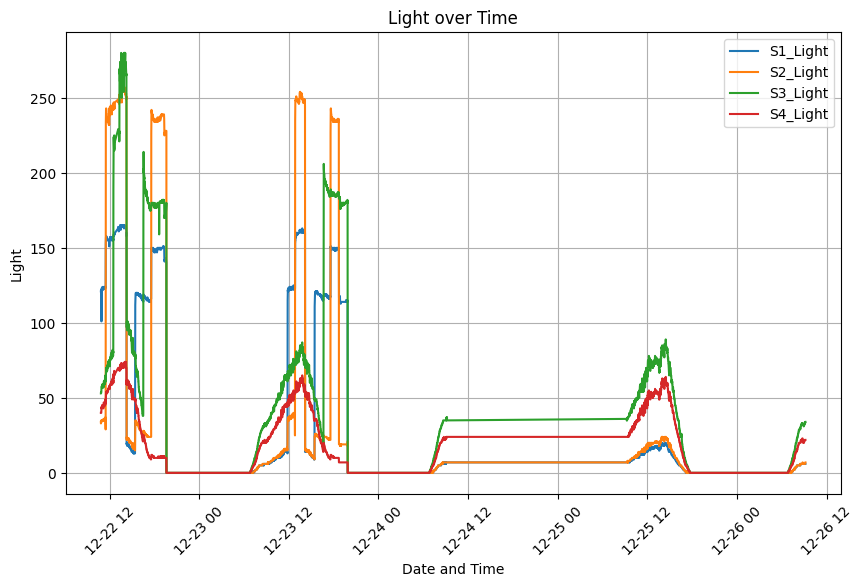


As can be seen from this pairplot, most of the points are scattered randomly, but S1\_temp, S2\_temp, and S4\_temp seem to have a near linear relationship. Additionally, in a chart that has any combination of 2 of these three sensors, we can see that a lower reading on both of the sensors, usually indicated a less occupied room.

A graph of a temperature

Description automatically generated

These charts confirm our conclusion from the pairplot, as we can see S1\_temp, S3\_temp and S4\_temp being very close to each other, while S2\_temp having very different results from the other 3 of them.

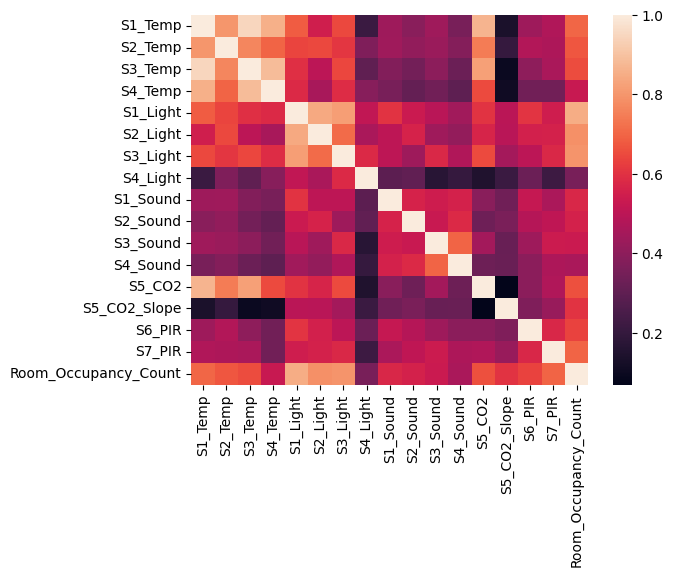


A graph with different colored lines

Description automatically generatedA graph with different colored lines

Description automatically generated

As can be seen from both the light and sound sensors, they do not have a strong correlation between each other, and we cannot draw any conclusions based on their behaviors.



The correlation heatmap further confirms that the strongest correlation in the features can be observed between S1\_temp and S3\_temp, followed by S1\_temp and S4\_temp along with S3\_temp and S4\_temp.

Learning Problem Being Solved (Singh et al., 2018)

The learning problem tackled involves using machine learning techniques to explore and analyze sensor data (CO2 levels, temperature, light, sound, and motion) and predict room occupancy. The goal is to predict whether a room is occupied based on the data from the different sensors. The problem is a supervised learning problem, specifically a classification problem since the output should be a categorical output (0, 1, 2, or 3). The project will apply various machine learning models, including XGBoost, Gradient Boosting, Random Forest, and Support Vector Machines, to solve this classification problem. The goal is to select and fine-tune the right combination of features, hyperparameters, and algorithms to accurately and reliably predict occupancy levels. The performance of these models is evaluated using metrics like accuracy, precision, recall and F1 score.

Data Preprocessing

1. **Feature-Target Separation:** The initial step in our data preparation process involved seperate the dataset into features (X) and the target variable (y). We achieved this by dropping the Room\_Occupancy\_Count column from our dataset to form the features (X), and isolated this column to represent our target variable (y). This separation is fundamental in supervised learning models, as it clearly delineates the input variables that the model will learn from and the output variable it will predict.
2. **Time Feature Transformation:** This transformation converts time from ‘hours:minutes:seconds’ format into number of seconds. It simplifies the time representation and makes it a continuous numerical variable, which is easier for models to interpret to understand temporal patterns, since time in its original format is complex for models to process.
3. **Date Encoding:** The Date column, initially in a string format, was transformed using Label Encoding, converting these dates into a series of numeric values, since most machine learning models cannot handle date formats directly. Encoding them into numeric values is necessary to include the date information in the model.
4. **Feature Scaling:** To standardize the range of our independent variables, we employed the StandardScaler, which normalized the feature values to have a mean of 0 and a standard deviation of 1. This normalization is particularly important for models sensitive to the scale of input features, as it ensures that all features contribute equally to the model’s learning process.
5. **Train-Test Split:** For the purpose of evaluating our model’s performance on unseen data, we split our dataset into training and test sets, using 80% of the data for training and reserving 20% for testing.
6. **Additional Preprocessing Exploration:** We explored various other preprocessing methods, including Random Undersampling, Cluster Centroids Undersampling, SMOTE, Sequential Feature Selector, and Select K Best with hyperparameter tuning. We also experimented with transforming our data using np.log. However, these techniques did not result in a significant improvement in our models’ performance. In some instances, they led to increased computational time with no to very minimal benefit, or in the worst case, weakened the model’s performance, therefore, they were not utilized in the final code.

# ***Methods***

Model Selection and Justification

1. **Random Forest Classifier:** (IBM Corporation, 2021; Shafi, 2023; R, 2024)

* **Nature of Classification Task:** As an ensemble of decision trees classifiers, Random Forest is suitable for classification tasks. It works by aggregating the predictions of multiple decision trees to improve accuracy and control over-fitting.
* **Handling Complex Data:** Sensor data exhibit complex structures with interactions between variables. Random Forest can capture these interactions without needing explicit feature engineering.
* **Robustness to Noise and Overfitting:** It is less prone to overfitting compared to individual decision trees by taking the mode of the results of each tree, a crucial aspect when dealing with noisy sensor data.
* **Nature of Data:** Random Forest is robust to outliers and non-linear data, making it suitable for real-world sensor data.
* **Feature Handling:** It effectively handles a large number of input features and maintains accuracy even when some of the features may not be relevant, which may be the case when dealing with sensor data.
* **Interpretability:** Provides insights into feature importance, aiding in understanding which sensors contribute most to occupancy detection.
* **Flexibility:** Offers flexibility with numerous hyperparameters like the number of trees, depth of trees, criterion, maximum features, etc., allowing fine-tuning for the specific dataset.

1. **Gradient Boosting Classifier:** (Nelson, 2020; GeeksforGeeks, 2022; Saini, 2024)

* **Sequential Improvement for Classification:** This is a classification model that builds trees sequentially one at a time, unlike random forests, with each tree trying to correct the errors made by previously trained trees by improving decision boundaries.
* **Handling of Non-Linear Relationships:** Its ability to model complex and non-linear relationships is essential in sensor data, where such patterns are common.
* **Adaptability to Imbalanced Data:** Given the imbalance in occupancy data (more non-occupied), it can be used for such scenarios, and make accurate predictions.
* **Flexibility:** Offers flexibility with numerous hyperparameters like the number of trees, depth of trees, learning rate, etc., allowing fine-tuning for the specific dataset.

1. **Support Vector Machine (SVM):** (Stecanella, 2017; Gandhi, 2018; Saini, 2021)

* **Optimal Hyperplane for Classification:** SVC is designed for classification tasks, aiming to find the best hyperplane that best separates the classes in the feature space.
* **Versatility with Different Kernel Functions:** The use of kernel functions and kernel trick allows SVC to adapt to various data distributions, a valuable feature for the diverse range of sensor data patterns.
* **Effectiveness in High-Dimensional Spaces:** Sensor data can be high-dimensional, and SVC excels in such environments due to its reliance on support vectors rather than the dimensionality of the data.
* **Kernel Trick:** It allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space, enhancing its ability to capture complex relationships.
* **Robustness:** SVC is known for its robustness and effectiveness in achieving high accuracy.
* **Flexibility:** It offers flexibility with numerous hyperparameters like the kernel, c, gamma, etc., allowing fine-tuning for the specific dataset.

1. **XGBoost Classifier:** (T, 2021; Kazmi, 2023; Tuychiev, 2023)

* **Advanced GB for Classification:** XGBoost is an advanced implementation of gradient boosting with improved accuracy and efficiency, making it very suitable for classification tasks for room occupancy for sensor data.
* **Regularization and Generalization:** It incorporates regularization terms to prevent overfitting, crucial for achieving generalizable classification outcomes in sensor data.
* **Scalability and Speed:** Known for delivering high performance and speed, its design allows for parallel processing, making it scalable and faster, which is a significant advantage for processing large volumes of sensor data.
* **Flexibility:** Offers flexibility with numerous hyperparameters like the number of trees, depth of trees, learning rate, minimum child weight, etc., allowing fine-tuning for the specific dataset.
* **Handling Missing Values:** XGBoost has the ability to handle missing values.

**The Role of Grid Search in Model Selection and Optimization**

Grid Search CV plays an important role in model optimization. It systematically works through multiple combinations of parameter tunes, cross-validating as it goes to determine which tune gives the best performance. Here’s how it applies to our models:

* **Hyperparameter Tuning:** Each of the models selected has hyperparameters that significantly impact their performance. Grid Search helps in finding the optimal values for these hyperparameters by giving it all the combinations that should be tested.
* **Improving Model Accuracy:** By exhaustively searching through a specified parameter space, Grid Search ensures that the model is not just fitted to the data but is optimized for the highest accuracy.
* **Avoiding Overfitting:** Through cross-validation, Grid Search also aids in preventing overfitting. By using different subsets of the data for training and validation, it ensures that the model generalizes well to unseen data.

In conclusion, the chosen classification algorithms are well-suited for the task of room occupancy estimation from sensor data due to their inherent capabilities in handling classification tasks, complex data patterns, and high-dimensional spaces.

Testing Machine Learning Applications

It is crucial that we test our models using holdout test data in addition to k-fold cross-validation, which enables us to evaluate the model’s performance across various data subsets. This ensures that our models are not overfitting and can be applied to new, unseen data.

1. **Data Preparation:** The dataset is split into a training set that is used for both training and cross-validation, and a holdout test set that is reserved for the final evaluation.
2. **Hyperparameter Tuning with Grid Search CV:** Grid Search CV is integrated with k-fold cross-validation to optimize the hyperparameters for each model. This step ensures that the models are not just tested but also tuned for optimal performance. The best parameters identified by Grid Search are used for the final model training.
3. **K-Fold Cross-Validation Setup:** The training dataset is divided into ‘k’ equal subsets. Usually, figures like 5 or 10 are used for ‘k,’ but in this case, we’ll use 5. The training set is made up of the leftover k-1 subsets after one of the ‘k’ subsets is utilized as the validation set for each iteration.
4. **Model Training and Validation:** In each iteration, the model is trained on the k-1 training subsets. The trained model is then validated on the subset reserved for validation and the score of the model is calculated.
5. **Performance Aggregation:** After completing the iterations, the scores from each fold are aggregated to produce an overall performance measure. This usually involves calculating the mean and standard deviation of the scores across all folds. Finding the most efficient model and hyperparameters is aided by comparing the performance of several models and their hyperparameters using the aggregated results.
6. **Final Model Evaluation:** The final step involves evaluating the tuned models on the holdout test set, which is an important step as it provides an unbiased evaluation of the model’s performance on unseen data, then metrics like accuracy, precision, recall, F1 score are calculated to assess the final performance of each model.

Machine Learning Algorithms Used

1. **Random Forest:** (IBM Corporation, 2021; Shafi, 2023; R, 2024)

It is an ensemble machine learning algorithm. In ensemble learning, several models are combined to enhance overall performance. The idea of ensembling is embodied in the Random Forest algorithm, which creates a “forest” by constructing and combining many decision trees, where every tree is trained on different subsets of the data and features, to ensure diversity and robustness in the model, and then makes its own prediction, and the Random Forest’s final result is the mode of all of these predictions, where a decision tree is a structure that resembles a flowchart, with each leaf node being a class, each internal node representing a test, and each branch representing the test’s result.

**How Random Forest Works for Classification**

* **Creation of the Forest**

1. **Bootstrap Aggregating (Bagging):** Random Forest applies the principle of bagging, where it creates multiple subsets of the original dataset with replacement. Each subset is used to train a separate decision tree.
2. **Feature Randomness:** When splitting a node during the construction of the tree, the Random Forest algorithm selects a random subset of features. This randomness ensures that the trees in the forest are different from each other.
3. **Training Individual Trees:** Each tree is grown to the largest extent possible without pruning. The unpruned trees in a Random Forest tend to overfit their training set, but the ensemble approach compensates for this overfitting.

* **Prediction Process**
  + 1. **Passing through Trees:** Input is passed through each of the trees in the forest.
    2. **Individual Tree Prediction:** Each tree provides its prediction. In classification, it’s the predicted class; in regression, it’s a numerical value.
    3. **Aggregation of Predictions:** For classification, the final prediction is the mode of all the predictions. For regression, it’s the mean of the predictions.

Many sectors, including but not limited to finance, healthcare, stock market research, and e-commerce, use random forests. Based on data collected by several sensors, Random Forest is used in our example to predict the rooms occupancy count.

**Limitations**

* **Complexity:** The model can be quite complex and require significant computational resources, especially with a large number of trees.
* **Interpretability:** Although it can provide information about feature importance, a Random Forest is harder to interpret and visualize unlike a single decision tree.
* **Performance:** For tasks with very high-dimensional data or text data, other algorithms might perform better.

**Pseudocode:**

RandomForestClassifier(n\_estimators, max\_features, max\_depth, criterion):  
 Create an empty list for the forest of trees  
 For each estimator in n\_estimators:  
 Select a random subset of the training data  
 If max\_features is specified, select a random subset of features  
 Create a decision tree with its nodes and leaves using the selected data and features  
 If max\_depth is specified, limit the depth of the tree  
 Use the criterion for splitting nodes  
 Add the tree to the forest  
 Return the forest of trees  
To predict a new sample:  
 Let each tree in the forest make a prediction  
 Combine the predictions (majority voting for classification)  
 Return the combined prediction

1. **Gradient Boosting:** (Nelson, 2020; GeeksforGeeks, 2022; Saini, 2024)

Unlike Random Forests, which create multiple trees at once, Gradient Boosting builds them sequentially, with each tree trying to correct the mistakes of the previous ones, which is a of a type of ensemble learning technique called “Boosting”. It is the process of combining multiple weak learners into one strong learner by having each succeeding model learn from the mistakes of the others, improving the overall predictive accuracy. In Gradient Boosting, the term “Gradient” refers to the optimization technique Gradient Descent’s use in minimizing the loss function.

**How Gradient Boosting Works for Classification**

* **Creating the Initial Model:** Gradient Boosting starts with a base model, often a simple decision tree. This model makes initial predictions, which are likely to be imperfect, leaving residuals.
* **Iteratively Improving the Model**

1. **Building Sequential Models:** New models are added, each focusing on the residuals left by the previous models.
2. **Learning Rate:** A learning rate is applied to slow down the learning process. This prevents overfitting and allows for more robust generalization.
3. **Gradient Descent Application:** The algorithm uses gradient descent to determine how the parameters of the model should be adjusted to minimize the loss function.

* **Prediction Process:** Similar to Random Forests,once all the models are built, predictions are made by aggregating the outcomes of all individual trees.

Gradient Boosting has been successfully applied in many industries, including risk management, biology for gene classification, and in technology for ranking algorithms used by search engines. In our example, Gradient Boosting is used to predict the room occupancy count based on data from multiple sensors.

**Limitations**

* **Prone to Overfitting:** If not properly tuned, Gradient Boosting models can overfit, especially on noisy datasets.
* **Computationally Intensive:** The sequential nature of boosting can be more computationally intensive compared to algorithms that build models in parallel, like Random Forests.
* **Parameter Tuning:** It requires careful tuning of several parameters (like the number of trees, depth of trees, learning rate), which can be time-consuming.

**Pseudocode:**

GradientBoosting(learning\_rate, max\_depth, n\_estimators):  
 Initialize the model with a constant prediction (e.g., mean of target)  
 For each estimator in n\_estimators:  
 Compute the negative gradient of the loss function  
 Fit a decision tree to this gradient  
 If max\_depth is specified, limit the depth of the tree  
 Update model with the sum of the previous model and the scaled output of the tree  
 Scale the output by the learning\_rate  
 Return the final model  
To predict a new sample:  
 Sum the predictions from all trees  
 Return the final prediction

1. **Support Vector Machine:** (Stecanella, 2017; Gandhi, 2018; Saini, 2021)

SVM is a machine learning that is special due to its ability to find the optimal boundary (hyperplane) that best divides a dataset into different classes in a dataset. The strength of SVM lies in its ability to handle both linear and nonlinear data.

**Margin and Support Vectors**

* **Margin:** Refers to the distance between the hyperplane and the nearest data points from each class. SVM seeks to maximize this margin to enhance model generalizability.
* **Support Vectors:** The data points closest to the hyperplane, which influence its position and orientation. These points are pivotal in constructing the SVM model.

**How SVM Works for Classification**

* **Creating the Model:** Depending on the nature of the data, a kernel is chosen. The most common kernels are linear, Radial Basis Function (RBF), and polynomial.
* **Linear Kernel:** Used for linearly separable data.
* **RBF Kernel:** Useful for non-linear data separation in higher dimensions.
* **Polynomial Kernel:** Effective for data where relationships between features are polynomial in nature.
* **Training the Model**
* **Constructing Hyperplanes:** SVM constructs hyperplanes in a multidimensional space to separate different classes.
* **Maximizing Margin:** The optimal hyperplane is the one that maximizes the margin between the closest points of different classes, known as support vectors.
* **Handling Non-Linearity:** For non-linear kernels, the data is projected into a higher-dimensional space where it is easier to find a separating hyperplane.
* **Optimization Process**
* **Solving an Optimization Problem:** SVM involves solving a quadratic optimization problem to find the hyperplane with the maximum margin.
* **Regularization Parameter (C):** This parameter controls the trade-off between achieving a low error on the training data and maximizing the margin. A higher value of C can lead to overfitting.
* **Prediction Process**
* **Classification of New Data:** For a new data point, the SVM model predicts the class based on which side of the hyperplane the point lies.
* **Decision Function:** The decision function, based on the chosen kernel and the support vectors, determines the class of the new data point.

**Kernel-Specific Considerations**

* **Linear SVM:** Straightforward with less computational complexity but limited to linearly separable data.
* **RBF SVM:** More flexible for non-linear data, sensitive to the gamma parameter which defines the influence of a single training example.
* **Polynomial SVM:** Effective for complex datasets, involves tuning the degree of the polynomial, which can increase computational complexity.

SVMs are used in Bioinformatics in protein classification and cancer classification. Also, SVMs are used in image recognition and financial analysis, specifically for predicting market movements and credit scoring. In our example, SVM is used to predict the room occupancy count based on data from multiple sensors.

**Limitations**

* **Computational Efficiency:** For large datasets, SVM can be computationally intensive, particularly when selecting and tuning kernel parameters.
* **Interpretability:** SVM models, especially with non-linear kernels, can be challenging to interpret compared to simpler models like decision trees.
* **Sensitivity to Parameters:** The performance of SVM is heavily dependent on the choice of kernel and its parameters, as well as the regularization parameter C.

**Pseudocode:**

SVM(C, gamma, kernel):  
 Choose the kernel function based on the kernel parameter  
 Transform the input data into the chosen kernel space  
 Initialize the support vector machine model  
 Use the C parameter to set the regularization strength  
 If using RBF kernel, use gamma to define the influence of a single training example  
 Train the model to find the hyperplane that best separates the classes  
 Optimize model to maximize margin between classes to minimize misclassifications  
 Identify support vectors as the data points nearest to the hyperplane  
To predict a new sample:  
 Transform the sample into the kernel space  
 Use the trained model to predict the class of the sample  
 Return the predicted class

1. **eXtreme Gradient Boosting (XGBoost):** (T, 2021; Kazmi, 2023; Tuychiev, 2023)

It is based on the ideas of gradient boosting, and it is an enhanced implementation of gradient boosting. A key distinction of XGBoost is its focus on computational efficiency and model performance. It introduces several optimizations in both algorithmic design and engineering implementation to minimize the model’s complexity, reduce overfitting, and to utilize the computational resources effectively.

**How XGBoost Works for Classification**

* **Creating the Initial Model:** XGBoost starts with a simple model, often a decision tree, as its base learner. This initial model makes the first set of predictions, which is likely to include misclassifications.
* **Iteratively Improving the Model**
* **Adding Trees Sequentially:** Unlike Random Forest, XGBoost adds trees one at a time. Each new tree corrects the errors made by the previously added one.
* **Gradient Boosting Framework:** The model uses a gradient boosting framework, where it focuses on minimizing the loss function by using gradient descent.
* **Regularization:** XGBoost introduces regularization (L1 and L2) in its objective function to control overfitting, making it more robust than standard gradient boosting.
* **Optimization of Computational Resources:** XGBoost is designed to be highly efficient, scalable, and fast. It uses techniques like approximate tree learning and sparsity awareness.
* **Prediction Process**
* **Aggregation of Tree Outputs:** For a given input, XGBoost makes predictions by summing the outputs of all the individual trees.
* **Final Prediction**: In classification, the summed score is transformed (using a logistic function, for example) to predict the probability for each class. The final prediction is the class with the highest probability.

XGBoost is highly successful in various sectors. In financial services, it’s used for credit scoring and fraud detection. In healthcare, it assists in disease prediction. Retailers use it for customer behavior analysis and inventory forecasting. In our example, XGBoost is used to predict the room occupancy count based on data from multiple sensors.

**Limitations**

* **Complexity and Overfitting:** While powerful, XGBoost models can become complex and may overfit if not properly regularized and tuned.
* **Computational Resources:** For very large datasets, XGBoost can be resource-intensive, requiring considerable memory and processing power.

**Pseduocode:**

XGBoost(learning\_rate, max\_depth, min\_child\_weight, n\_estimators):  
 Initialize the model with zero or a base prediction  
 For each estimator in n\_estimators:  
 Compute the gradient and hessian of the loss function  
 Create a new tree to predict the gradients  
 If max\_depth is specified, limit the depth of the tree  
 Use min\_child\_weight to control overfitting  
 Update the model with the weighted sum of the previous model and the new tree  
 Update the weights based on the learning\_rate  
 Return the final model  
To predict a new sample:  
 Sum the predictions from all trees  
 Return the final prediction

# ***Evaluation***

Performance Measures Used for Model Evaluation

* **Accuracy:** Accuracy is the most intuitive performance measure. It is simply the ratio of correctly predicted observations to the total observations. It answers the question, “Out of all the predictions, how many did we predict correctly?” This measure is particularly useful as a basic indicator of model performance.
* **Precision:** Precision is crucial in scenarios where the cost of false positives is high. As it evaluates the accuracy of the positive predictions. It is the ratio of correctly predicted positive observations to the total predicted positives. For our study, it means the proportion of correctly predicted occupancy counts compared to all instances where the model predicted that specific count.
* **Recall:** Recall, also known as Sensitivity, measures the model’s ability to identify all relevant instances. In our study, it refers to the proportion of actual occupancy counts that the model correctly identified. Recall is especially important in cases where missing the actual positive cases is undesirable. Given the imbalance in our dataset, recall helped in understanding how well our model is predicting less frequent classes.
* **F1 Score:** The F1 Score is a harmonic mean of Precision and Recall, therefore it is used when we need a single metric to balance both the precision and the recall, which is often the case in imbalanced datasets.
* **Confusion Matrix:** It is simply a matrix that shows us how many correct predictions where made and how many wrong ones, by showing us the number of the True Positives, False Positives, True Negatives, and False Negative.

Rationale for Using Specific Performance Metrics

* **Accuracy:** Despite its limitations in imbalanced scenarios, accuracy is a fundamental benchmark in classification tasks. It provides an immediate sense of how often the model is correct, which is a straightforward and intuitive measure. Accuracy offers a broad overview of model performance across all classes, serving as a primary checkpoint before delving into more advanced metrics. In our context, while accuracy tells us how often the model is right, it does not distinguish between the types of errors made. It is more reliable when class distribution is balanced, which is not the case in our study. Thus, its role was more of a preliminary indicator rather than a definitive measure of performance.
* **Precision:** Precision measures the proportion of positive identifications that were actually correct. In the context of room occupancy, this translates to how many times the model correctly predicted a specific occupancy count out of all the times it predicted that count. High precision is crucial in situations where the cost of false positives is significant. For example, overestimating room occupancy might lead to unnecessary resource allocation or safety concerns. Precision alone might not be reliable in imbalanced datasets as it does not consider the true negative cases. Therefore, its effectiveness increases when used in conjunction with other metrics like recall.
* **Recall:** Recall assesses the model’s ability to identify all actual positives. For our occupancy prediction, it reflects how well the model identifies each occupancy level, regardless of how frequently it occurs in the dataset. In practical scenarios, a high recall means the model effectively captures all varying occupancy levels, which is vital for efficient space utilization and meeting user requirements. High recall can sometimes lead to a decrease in precision, as the model aims to capture as many positives as possible, potentially increasing false positives. Therefore, recall must be balanced against precision.
* **F1 Score:** The F1 Score is the harmonic mean of precision and recall, providing a single metric that balances both. It is particularly useful when we need a balance between the importance of precision and the need for high recall. In imbalanced datasets, like ours, where minor classes are as important as the major ones, F1 offers a more balanced view than looking at precision and recall individually, therefore that is why the f1 score is our main focus in this study. The F1 Score serves as a robust overall performance indicator, especially when precision and recall are equally important.
* **Macro Averaging:** In imbalanced datasets, using a simple average (micro averaging) could be misleading as it tends to be dominated by the majority class. Macro averaging treats each class equally, emphasizing the performance on minority classes as much as on the majority class. It prevents the majority class from overshadowing the performance metrics and gives an equal voice to each class, regardless of its frequency. Our decision to use macro averages was specifically to counter the imbalance in our dataset. With ‘0’ (no occupancy) being the most frequent class, there was a risk that our model’s ability to predict less frequent classes (1, 2, 3) would be overlooked. Macro averaging ensured that the performance in these less frequent classes was equally represented. Using macro averages helped us in thoroughly evaluating the model’s performance across all classes.
* **Confusion Matrix:** The Confusion Matrix is crucial for evaluating our multiclass classification model, offering a detailed 4x4 grid that illustrates the model’s performance across these classes. Each row in this matrix represents the actual classes, while the columns show the predicted classes, showing the model’s TP, FP, TN, and FN values. For instance, it shows how often the model correctly or incorrectly predicts each occupancy level in a room. This granular view is especially important in our context for understanding the model’s strengths and weaknesses in differentiating between the occupancy levels. It’s instrumental in refining the model, ensuring it aligns with our goals, particularly in an imbalanced dataset where certain classes occur less frequently.

Model Enhancements Based on Performance Measures

In the process of developing robust machine learning models, we employed various strategies to enhance the model’s performance. While some techniques proved beneficial, others did not yield the expected improvements. This section evaluates the effectiveness of these all the different model enhancements based on our performance measures, mainly the F1 score and the accuracy.

* **Unsuccessful Strategies**

1. **Advanced Preprocessing Techniques:** Our exploration into preprocessing techniques such as Random Undersampling, Cluster Centroids Undersampling, SMOTE (Synthetic Minority Over-sampling Technique), Sequential Feature Selector, and Select K Best, coupled with hyperparameter tuning, did not result in significant performance improvements. In some cases, these techniques led to increased computational time without substantial benefits, and in other instances, they actually decreased computational time but deteriorated the model’s performance.
2. **Data Transformation Using np.log:** The application of a logarithmic transformation to our data, intended to normalize the distribution of certain features, did not contribute positively to the model’s predictive accuracy. This suggests that the nature of our data did not align well with this type of transformation.
3. **Label Encoding of Time Feature:** The attempt to encode the time feature using label encoding proved ineffective. This approach did not capture the temporal patterns effectively, which are crucial for predicting room occupancy accurately.
4. **Micro Average in Evaluation Metrics:** Our initial use of micro averaging in performance metrics did not provide a clear insight into the model’s ability to predict less frequent classes, which was critical in our imbalanced dataset.
5. **Adding the Degree Parameter to SVM:** When experimenting with various values of the degree hyperparameter, it always resulted after using the grid search in a degree of 3, and it was therefore removed as it is already the default value.

* **Successful Strategies**

1. **Formatting Time as Seconds:** Converting the time feature into a continuous numeric variable by calculating the number of seconds proved to be a significant enhancement. The temporal representation was made simpler by this transformation, which improved our models’ ability to understand and recognize useful patterns from temporal data. As a result, our models’ performance was clearly enhanced.
2. **Standard Scaling of Data:** Using the StandardScaler to normalize the feature values (to have a mean of 0 and a standard deviation of 1) was a successful strategy, as it ensured that all features contributed equally to the model’s learning process. It particularly improved the model’s accuracy and F1 score, indicating a more balanced prediction capability across different classes.
3. **Macro Averages in Evaluation Metrics:** Switching to macro averages for our performance metrics was a pivotal change. It allowed us to evaluate the model’s effectiveness across all classes equally, thereby providing a more balanced view of its predictive capability in the context of our imbalanced dataset.
4. **Experimenting with Hyperparameters:**Based on my initial resaults from my initial hyperparameters, I expereminted with other hyperparameters for all the models until I found the hyperparameters that best fit the models and the dataset to be able to do grid search between them.
5. **The Number of k-*folds:*** After experimenting with 10 folds and 5 folds, it was apparent that there was no significant difference between both of their results, but the 10 folds was taking twice as long to run, therefore, I opted to use 5 folds.

# ***Results and Discussion***

Result Reliability and Algorithm Fit

1. **Random Forest**

* **Best Parameters:** Criterion: Gini, Max Depth: 8, Max Features: log2, Number of Estimators: 20.
* **Performance Metrics:**
* **Test:** Accuracy: 99.90%, Precision: 99.61%, Recall: 99.67%, F1: 99.64%.
* **Train:** Accuracy: 99.94%, Precision: 99.78%, Recall: 99.70%, F1: 99.74%.
* **Metrics Overview:**
* **Accuracy:** Exceptionally high 99.9%, indicating almost perfect classification ability.
* **Precision, Recall, F1 Score:** All around 99.6%, suggesting high consistency in both positive prediction and actual positive class identification.
* **Comparison with Training Metrics:** The training metrics are slightly higher but very close to the test metrics. This indicates a strong model that is not overfitting significantly or might not be overfitting at all.
* **Confusion Matrix Interpretation:** The almost perfect classification with only two misclassifications suggests that the model is highly effective for this dataset. Surprisingly enough, both of these misclassifications have nothing to do with the majority class, meaning that the model was not affected with the class imbalance.
* **Reliability and Fit:** The Random Forest model appears to be highly reliable, well-tuned, and generalizable. The very slight difference between training and testing metrics suggests that the model is neither overfitting nor underfitting.

1. **XGBoost**

* **Best Parameters:** Learning Rate: 0.2, Max Depth: 6, Min Child Weight: 1, Number of Estimators: 50.
* **Performance Metrics:**
* **Test:** Accuracy: 99.70%, Precision: 99.38%, Recall: 98.98%, F1: 99.18%.
* **Train:** Accuracy: 100%, Precision: 100%, Recall: 100%, F1: 100%.
* **Metrics Overview:**
* **Accuracy:** Very high 99.7%.
* **Precision, Recall, F1 Score:** All around 99%, indicating a strong performance in classification.
* **Training Metrics:** The perfect scores on training data are a red flag for potential overfitting. The model has potentially learned the training data too well, which doesn’t generalize perfectly to unseen data.
* **Confusion Matrix Interpretation:** Minor misclassifications are present, yet the overall performance is strong. However, the perfect training scores suggest the model might be too tailored to the training set.
* **Reliability and Fit:** XGBoost shows excellent predictive power but raises concerns about overfitting. Its reliability might be compromised in the face of new, varied data.

1. **Gradient Boosting**

* **Best Parameters:** Learning Rate: 0.2, Max Depth: 3, Number of Estimators: 100.
* **Performance Metrics:**
* **Test:** Accuracy: 99.70%, Precision: 99.24%, Recall: 98.98%, F1: 99.11%.
* **Train:** Accuracy: 100%, Precision: 100%, Recall: 100%, F1: 100%.
* **Metrics Overview:**
* **Accuracy:** Similar to XGBoost 99.7%.
* **Precision, Recall, F1 Score:** Slightly lower than XGBoost but still around 99%.
* **Training Metrics:** As with XGBoost, the perfect training scores suggest overfitting.
* **Confusion Matrix Interpretation:** Very similar to XGBoost, with a few errors but generally strong performance.
* **Reliability and Fit:** Gradient Boosting, much like XGBoost, shows potential overfitting. Its predictive accuracy is high meaning that it could be reliable, but the model may not be as effective on new data sets.

1. **SVM**

* **Best Parameters:** C: 0.1, Gamma: 10, Kernel: Linear.
* **Performance Metrics:**
* **Test:** Accuracy: 99.36%, Precision: 97.91%, Recall: 98.14%, F1: 98.02%.
* **Train:** Accuracy: 99.54%, Precision: 98.39%, Recall: 98.38%, F1: 98.39%.
* **Metrics Overview:**
* **Accuracy:** Lower than ensemble methods 99.36%, but still high.
* **Precision, Recall, F1 Score:** Around 98.1%, which is lower than the ensemble methods but still indicates a good model that might be slightly underfitting when compared to the other models.
* **Training Metrics:** Higher than test metrics but not perfect, suggesting the model is well-fitted to the data without significant overfitting.
* **Confusion Matrix Interpretation:** Shows more misclassifications than ensemble methods. This indicates that while the model is accurate, it is not as finely tuned to the dataset as the ensemble models.
* **Reliability and Fit:** SVM demonstrates a balanced approach. It is less prone to overfitting compared to ensemble methods and maintains a high level of accuracy. It may not be as sensitive to the dataset’s nuances as the other models, but it offers a more generalized fit.

**Overall Evaluation**

The ensemble models (Random Forest, XGBoost, Gradient Boosting) demonstrate exceptionally high accuracy and precision, but there are varying degrees of overfitting concern, particularly with XGBoost and Gradient Boosting.

Random Forest stands out for its balance, showing only a slight indication of overfitting, making it a highly reliable model for this dataset. Its performance metrics are robust across all categories, and the minimal discrepancy between training and testing metrics implies a well-fitted model.

XGBoost and Gradient Boosting both exhibit signs of potential overfitting. This is evident from their perfect training metrics, which suggest that these models may have learned the training data too well, possibly capturing noise along with the underlying pattern. While their testing performance is outstanding, the concern is their generalizability to new, unseen data. The high accuracy and precision are commendable, but the reliability in a real-world scenario might be slightly less than what the test metrics suggest.

SVM offers a more conservative but balanced model. Its slightly lower metrics indicate it may not capture the complexity of the dataset as effectively as the ensemble models. However, the closer alignment of its training and testing metrics suggests a better generalization capability. This balance makes SVM a reliable choice for scenarios where overfitting is a major concern, even though it might slightly sacrifice accuracy and precision.

Effectiveness of Algorithm Applications

1. **Random Forest**

* **Test Metrics Effectiveness:** Random Forest’s near-perfect accuracy 99.90% is indicative of its exceptional capability to correctly classify almost every instance. This is critical in applications where misclassification costs are high. Precision 99.61% and recall 99.67% being equally high suggest that the model is excellent at both identifying the positive class and minimizing false positives. The F1 score 99.64% confirms its balanced precision and recall, which is essential in scenarios where both false positives and false negatives carry significant consequences.
* **Training vs. Test Performance:** The slight difference between training and test metrics suggests that the model has learned the patterns in the data well without memorizing it. This implies its ability to generalize to new, unseen data, a key factor in real-world applications.
* **Effectiveness in Diverse Scenarios:** Given its balanced performance across all metrics, Random Forest is likely to be highly effective in a wide range of scenarios relating to room occupancy count, whether the room was empty or occupied.

1. **XGBoost**

* **Test Metrics Effectiveness:** The high accuracy 99.70% of XGBoost demonstrates its strong classification ability. Its precision 99.38% is slightly lower than Random Forest but still indicative of a high rate of true positive predictions. The recall 98.98% and F1 score 99.18% point towards a competent balance between sensitivity and positive prediction value. This makes XGBoost suitable for applications like room occupancy count where both the identification of true positives and the overall accuracy of classification are important.
* **Training vs. Test Performance:** The perfect training scores raise concerns about its potential to overfit, which means that it might not generalize well to new unseen data. In practical terms, while XGBoost might perform exceptionally well on known data, its effectiveness might slightly diminish when applied to new data sets.
* **Use Case:** Highly effective for smart building management systems, especially in scenarios where a balance is needed between performance and computational demands, but the user has to keep an eye for overfitting.

1. **Gradient Boosting**

* **Test Metrics Effectiveness:** Similar to XGBoost, Gradient Boosting’s high accuracy 99.70% shows its robust classification ability. The precision 99.24%, recall 98.98%, and F1 score 99.11% are displays its strong performance in correctly classifying positive instances and maintaining a balance between false positives and false negatives. This model is particularly effective in scenarios where it’s critical to minimize both types of classification errors.
* **Generalization Concerns:** As with XGBoost, the perfect training metrics might suggest overfitting, potentially limiting its effectiveness in unpredictably varying or new data contexts.
* **Application Suitability:** Highly effective for applications that require high accuracy in occupancy prediction and can accommodate the computational complexity that comes with Gradient Boosting, but the user has to keep an eye for overfitting.

1. **SVM**

* **Test Metrics Effectiveness:** SVM, with its accuracy of 99.36%, precision of 97.91%, recall of 98.14%, and F1 score of 98.02%, demonstrates strong but slightly more conservative performance. This model is particularly effective in applications where the cost of overfitting is higher than the cost of occasional misclassifications. For instance, in biomedical applications or image classification, where generalizability and robustness against overfitting are critical, SVM’s performance is highly advantageous.
* **Balanced Approach:** The lower, yet high, performance metrics compared to ensemble methods indicate a model that is less prone to overfitting, making it suitable for applications where model performance is a key consideration.
* **Applicability:** Its high effectiveness, combined with generally lower computational demands than some ensemble methods, and the small size of the dataset makes SVM suitable for applications where a more straightforward model is preferred.

**Overall Effectiveness**

The effectiveness of these algorithms goes beyond their numerical metrics and into their applicability in real-world scenarios such as ours. Random Forest stands out for its balanced and robust performance, making it highly effective in a variety of applications, specifically our room occupancy count application. XGBoost and Gradient Boosting, while powerful, might face challenges in new or varied datasets due to potential overfitting issues. SVM, with its more conservative approach, offers an effective alternative, particularly in scenarios where robustness and long-term reliability are paramount.

Algorithm Strengths and Weaknesses

* 1. **Random Forest**
* **Strengths:**
* **High Accuracy and Consistency:** Exhibited near-perfect classification ability, with high scores in precision, recall, and F1 score, suggesting excellent identification of occupancy levels.
* **Minimal Overfitting:** Showed only slight indications of overfitting, indicating a robust and reliable model.
* **Versatility:** Capable of handling complex data structures and interactions between variables.
* **Weaknesses:**
* **Computational Intensity:** Can be resource-intensive, especially with a large number of trees.
* **Interpretability:** While providing feature importance insights, a Random Forest model is generally harder to interpret than simpler models such as decision trees.

1. **XGBoost**

* **Strengths:**
* **High Predictive Power:** Demonstrated strong classification ability with high accuracy.
* **Efficiency:** Known for high performance and high speed in comparison with Gradient Boosting, in addition to being suitable for large volumes of data.
* **High Accuracy and Consistency:** Exhibited near-perfect classification ability, with high scores in precision, recall, and F1 score, suggesting excellent identification of occupancy levels.
* **Weaknesses:**
* **Overfitting Concerns:** Perfect training scores suggest a potential to overfit, which might limit its effectiveness on new or varied data sets.
* **Complexity:** Requires careful tuning of parameters, and its complexity might pose challenges for practical implementation.

1. **Gradient Boosting**

* **Strengths:**
* **Effective Classification:** Similar to XGBoost in terms of high accuracy and the ability to model complex data patterns.
* **High Accuracy and Consistency:** Exhibited near-perfect classification ability, with high scores in precision, recall, and F1 score, suggesting excellent identification of occupancy levels.
* **Weaknesses:**
* **Potential Overfitting:** Like XGBoost, it exhibits signs of overfitting with perfect training scores.
* **Computational Demand:** More computationally intensive due to the sequential nature of model building.

1. **Support Vector Machine (SVM)**

* **Strengths:**
* **Generalization Capability:** Exhibited a balanced approach with less risk of overfitting, maintaining high accuracy.
* **Effectiveness in High-Dimensional Spaces:** Excels in environments with high-dimensional data, making it suitable for complex sensor data.
* **High Accuracy and Consistency:** Exhibited strong classification ability, with high scores in precision, recall, and F1 score, suggesting very good identification of occupancy levels.
* **Weaknesses:**
* **Slightly Lower Performance Metrics:** While still high, its performance metrics are slightly lower compared to ensemble methods.
* **Parameter Sensitivity:** The performance of SVM is heavily dependent on the choice of kernel and its parameters.

**Recommendations**

* **Random Forest:** Recommended for scenarios requiring high accuracy with minimal risk of overfitting. Ideal for diverse applications in smart building management where both accuracy and model robustness are crucial.
* **XGBoost and Gradient Boosting:** Suitable for applications where high precision is essential, and the data is well-understood and consistent. However, caution is advised due to their tendency to overfit.
* **SVM:** Recommended for applications where the cost of overfitting is higher than the cost of occasional misclassifications. Additionally, it is perfect for scenarios where long-term model stability and dependability are crucial, such as in building management systems where incremental accuracy gains are not as essential as predicting constancy over time.

**Comparative Analysis**

* **Robustness:** Random Forest emerges as the most robust model, striking a balance between high performance and minimal overfitting. SVM also stands out for its robustness, particularly in high-dimensional spaces and generalization capability.
* **Accuracy vs. Overfitting:** XGBoost and Gradient Boosting offer superior accuracy but at the risk of overfitting. This trade-off is important in applications where the model must adapt to new or varying data.
* **Computational Efficiency:** While Random Forest and Gradient Boosting demand significant computational resources, XGBoost, despite its complexity, is noted for its efficiency and speed, beneficial for handling large datasets.
* **Practical Applicability:** SVM, with its balanced approach, offers practical applicability in scenarios where a more conservative model is preferable. This is in contrast to the ensemble models, which, while powerful, may require careful calibration and monitoring to avoid overfitting.

Future Enhancements and Project Limitations

While the current project has achieved notable successes, continuous improvement is essential to address its limitations and adapt to evolving requirements and technological advancements.

**Future Enhancements**

* **Integration of Deep Learning Techniques:** Deep learning models, particularly recurrent neural networks (RNNs) and convolutional neural networks (CNNs), could capture temporal patterns in sensor data more effectively. It is expected to notice improved accuracy in occupancy prediction, especially in complex scenarios involving varied occupancy patterns and environmental conditions.
* **Hyperparameter Tuning and Other Classification Models:** Experimenting with other hyperparameter might yield better results in terms of performance measures, along with the potential to experiment with other classification models such as Logistic Regression, KNN, and many others to find the best model for our dataset.
* **Experimenting with Pre-processing techniques:** Exploring other preprocessing techniques might yield us with better results in the future.
* **Real-time Data Processing:** Implementing real-time data analysis can provide immediate occupancy updates, essential for dynamic energy management systems. It is expected to notice enhanced responsiveness of the HVAC and lighting systems, leading to more efficient energy use and immediate adjustments to changing occupancy levels.
* **Enhanced Feature Engineering:** Feature engineering, such as creating new features or transforming existing ones, or using dimensionality reduction algorithms could uncover more insights. It is expected to notice improved model performance by capturing more relevant patterns in the data.
* **Expanding Data Collection:** By continuously updating the models with new data and by broadening the scope of data collection to include more variables and longer time frames, it is expected to notice enhanced predictions with more data as we will have more data for training the model.
* **Scalability Enhancements:** Optimizing models for better scalability and efficiency in larger or more complex building environments.

**Limitations**

* **Data Limitations:** The current dataset might not capture all relevant factors affecting room occupancy, leading to potential biases.
* **Model Generalizability:** Models like XGBoost and Gradient Boosting showed signs of overfitting, raising concerns about their performance on unseen datasets.
* **Computational Efficiency:** Some models require substantial computational resources, limiting their scalability.
* **Real-time Adaptability:** The current models are not designed for real-time data processing, which can limit their effectiveness in dynamic environments.

# ***References***

Gandhi, R. (2018) *Support Vector Machine*, *Towards Data Science*. Available at: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

GeeksforGeeks (2022) *Gradient Boosting in ML*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/ml-gradient-boosting/

IBM Corporation (2021) *What is Random Forest?*, *IBM Corporation*. Available at: https://www.ibm.com/topics/random-forest

Kazmi, H. (2023) *Classification using XGBoost in Python*, *educative.io*. Available at: https://www.educative.io/answers/classification-using-xgboost-in-python

Nelson, D. (2020) *Gradient Boosting Classifiers in Python with Scikit-Learn*, *Stackabuse.com*. Available at: https://stackabuse.com/gradient-boosting-classifiers-in-python-with-scikit-learn/

R, S.E. (2024) *Building a Random Forest Model: A Step-by-Step Guide*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

Saini, A. (2021) *Guide on Support Vector Machine (SVM) Algorithm*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

Saini, A. (2024) *Gradient Boosting: A Step-by-Step Guide*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/

Shafi, A. (2023) *Random Forest Classification with Scikit-Learn*, *Datacamp*. Available at: https://www.datacamp.com/tutorial/random-forests-classifier-python

Singh, A.P. *et al.* (2018) ‘Machine Learning-Based Occupancy Estimation Using Multivariate Sensor Nodes’, *2018 IEEE Globecom Workshops, GC Wkshps 2018 - Proceedings*, pp. 1–6. Available at: https://doi.org/10.1109/GLOCOMW.2018.8644432.

Stecanella, B. (2017) *Support Vector Machines (SVM) Algorithm Explained*, *Monkeylearn.Com*. Available at: https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/#:~:text=How Does SVM Work%3F,-The basics of&text=A support vector machine takes,to the other as red.

T, B. (2021) *Beginner’s Guide to XGBoost for Classification Problems*, *Towards Data Science*. Available at: https://towardsdatascience.com/beginners-guide-to-xgboost-for-classification-problems-50f75aac5390

Tuychiev, B. (2023) *Learn XGBoost in Python: A Step-by-Step Tutorial*, *Data Camp*. Available at: https://www.datacamp.com/tutorial/xgboost-in-python