Khaled Nabil Subhi Alkhouli

Al Hussein technical university

Occupancy Estimation for HVAC systems Energy Optimization: A Machine Learning Approach

Contents

[Introduction 2](#_Toc163260445)

[The Problem 2](#_Toc163260446)

[The Data Set 3](#_Toc163260447)

[Source and Collection method 3](#_Toc163260448)

[Attributes and Size 3](#_Toc163260449)

[Domain 3](#_Toc163260450)

[The Learning Problem 4](#_Toc163260451)

[Preparing Training and Test Sets 4](#_Toc163260452)

[Methods 5](#_Toc163260453)

[Why the used models are appropriate to solve this problem? 5](#_Toc163260454)

[Testing the machine learning applications 6](#_Toc163260455)

[Machine Learning Algorithms Used 8](#_Toc163260456)

[Random Forest 8](#_Toc163260457)

[Support Vector Machine (SVM) 12](#_Toc163260458)

[Gradient Boosting 14](#_Toc163260459)

[XGBoost 16](#_Toc163260460)

[Evaluation 17](#_Toc163260461)

[Performance Measures Used 17](#_Toc163260462)

[Justification of Performance Measures Used 17](#_Toc163260463)

[Evaluation of Metrics and Enhancing Models 18](#_Toc163260464)

[Results and Discussions 20](#_Toc163260465)

[Reliability of Results 20](#_Toc163260466)

[Analysis of the Results, and Algorithms Effectiveness 20](#_Toc163260467)

[Strengths and Weaknesses of the Used Algorithms 24](#_Toc163260468)

[Enhancements and Limitations 25](#_Toc163260469)

[References 26](#_Toc163260470)

# Introduction

## The Problem

The problem investigates the opportunity to optimize HVAC systems’ energy usage and consumption by estimating the number of occupants in a room using machine learning. These systems can be optimized by making them demand-driven depending on the human occupancy of specific areas. In this problem, non-intrusive environmental sensors, such as CO2, temperature, light, motion, and sound, will be relied on to estimate occupancy without relying on video-based systems. Optimizing these systems is essential for various reasons as follows:

* Cost Savings: One of the most significant reasons to optimize HVAC energy consumption is saving costs, as energy-efficient systems can lead to lower utility bills which reduces operational expenses (Sankaranarayanan, et al., 2022).
* Sustainability: HVAC systems are significant contributors to energy consumption and gas emissions which affect the environment negatively. Therefore, optimizing these systems can reduce the negative environmental impact, promote sustainability, and mitigate climate change (Utilities One, 2023).
* Improved Comfort: Optimized HVAV systems provide better comfort for users, by maintaining consistent temperature and humidity levels (Saman Taheri, 2022).
* Extended Equipment Lifespan: Optimizing HVAC systems contributes to the longevity of the equipment, reducing the need for costly repairs and premature replacements (Utilities One, 2023).

## The Data Set

### Source and Collection method

The dataset was collected using IoT devices (sensors) as a part of an experiment for occupancy estimation in a room that was introduced in the paper titled “Machine Learning-based Occupancy Estimation Using Multivariate Sensor Nodes” (Singh, et al., 2018). The experiment was conducted in a 6m x 4.6m room, where 7 sensor nodes and one edge node were deployed in a star configuration. The sensor nodes transmitted data every 30 seconds to the edge node using wireless transceivers. The room had a test lab setup with four office desks, a window with blinds, and a self-closing glass door. The sensors used included temperature, light, sound, CO2, and passive infrared (PIR), and no HVAC systems were used while collecting the data. The data was collected for a period of 4 days in a controlled manner, with occupancy levels varying between 0 and 3 people. The sensors were deployed to measure environmental parameters, and the measurements were recorded at regular intervals.

### Attributes and Size

The dataset contains 10129 instances and 18 features. The features include:

* Date and Time: Representing the timestamp of the data.
* Temperature (S1\_Temp, S2\_Temp, S3\_Temp, S4\_Temp): Continuous values in degrees Celsius.
* Light (S1\_Light, S2\_Light, S3\_Light, S4\_Light): Integer values representing Lux.
* Sound (S1\_Sound, S2\_Sound, S3\_Sound, S4\_Sound): Continuous values representing amplifier output read by ADC in volts.
* CO2 (S5\_CO2): Integer values representing parts per million (PPM) of CO2.
* CO2 Slope (S5\_CO2\_Slope): Continuous values representing the slope of CO2 values calculated in a sliding window.
* PIR (S6\_PIR, S7\_PIR): Binary values representing motion detection.
* Room\_Occupancy\_Count: Integer values representing the actual value of the number of occupants in the room, which is the target variable for the problem.

### Domain

The dataset falls within the domain of energy engineering, specifically HVAC systems, with the use of computer science, specifically machine learning. The focus of the experiment is to leverage multiple non-intrusive environmental sensors to estimate the precise number of occupants in a room. The dataset is suitable for classification tasks, particularly for predicting room occupancy based on sensor readings, to optimize the performance of HVAC systems.

## The Learning Problem

The learning problem is a non-binary classification problem that aims to train models capable of accurately predicting the precise count of individuals within a room, using data emanating from a wireless sensor network.

The key aspects of the learning problem are as follows:

* Task: Non-binary classification for occupancy estimation, which involves predicting the number of people present in a room at a given time.
* Features: The dataset has features extracted from different types of sensors that give different information about the room.
* Target Variable: The target variable is the actual count of occupants in the room, labeled as "Room\_Occupancy\_Count".
* Learning Models: There will be used four supervised machine learning algorithms for the classification, which are Random Forest, Support Vector Machine, Gradient Boosting, and XGBoost.
* Evaluation Metrics: Performance is assessed using metrics such as accuracy, F1 score, and confusion matrix, providing a comprehensive evaluation of the models' ability to estimate occupancy counts, and to help choose the best possible model for this problem.

The objective is to design a model that leverages information from multiple sensors to provide an accurate and real-time estimation of the number of occupants in a room, to optimize energy consumption and performance of HVAC systems.

## Preparing Training and Test Sets

I have employed multiple techniques and principles throughout the steps of preparing the training and testing sets for the models. I have followed the following steps:

1. Exploratory Data Analysis (EDA):

* Checked for null values in the dataframe.
* Examined data types of the columns.
* Converted Date and Time columns into a single DateTime column and extracted features such as year, month, day, hour, minute, and second, each into a separate column.
* Visualized and understood the distribution of the features.
* Visualized the distribution of Room\_Occupancy\_Count using a count plot.
* Applied Synthetic Minority Over-Sampling Technique (SMOTE) to handle class imbalance, which was used only on the training set.
* Plotted the resampled class distribution.
* Used Minimax Normalization to scales the data, which was used on the training and the testing data

1. Created a dummy classifier:

* To find a baseline model performance.

1. Grid Search and Cross Validation:

* Prepared a set of parameter that I want to test the model training on for each model
* Used grid search with the set parameter and 5 folds of cross validation
* Training the models and hyper parameter tuning
* Choosing the best model

# Methods

## Why the used models are appropriate to solve this problem?

Since the learning problem is a non-binary classification, all used classifiers are capable of handling such classification problems. Also, the analysis of the dataset show that the data is non-linear, which briefly means that the models could be suitable to solve this problem, as they do not rely on the data being linearly separable, or the relationships between features and the target variable are not linear. A further justification is as follows:

1. Random Forest: Random Forest is an ensemble method that builds multiple decision trees and train each of them on a random set of the original data, with replacement. It is strong and less likely to be affected to overfitting and outliers. Random Forest provides insights into feature importance, which is valuable for understanding which features contribute more to the occupancy estimation. Since the dataset is imbalanced, Random Forests can deal with imbalanced data. The ensemble nature of Random Forests helps mitigate the impact of imbalanced classes. Each decision tree in the Random Forest is trained on a random subset of the data, and the final prediction is often based on a voting approach. Random Forests allow assigning different weights to classes. This means that more importance can be assigned to the minority class during the training process. Finally, the dataset has 18 features and 10129 instances which can be considered a mid to high dimensional data, which the Random Forest is extremely capable of handling.
2. Support Vector Machine (SVM): SVM is a powerful algorithm for multiclass classification, which suits the problem where occupants can be in four different count categories. Also, SVM is effective in high-dimensional spaces, and the dataset has 18 features. SVM allows assigning different penalties to different classes, which used to give more importance to the minority class. Also, SVM is a versatile model with the capability to adapt to various types of data by employing different kernel functions, such as linear, polynomial, and radial basis function (RBF), which allows for optimization based on the characteristics and complexities present in the dataset, providing multiple options to enhance performance and capture patterns.
3. Gradient Boosting: Gradient Boosting builds multiple weak learners sequentially, where each learner corrects the errors of its predecessor. This helps in finding complex relationships in the data, and from the distribution of the data and its size, relationships do not look simple to capture. Also, Gradient Boosting tends to generalize well, making it suitable for predicting occupancy counts based on various sensor readings, and other features. Gradient Boosting is known for its high predictive accuracy, and that it handles mixed types of data well.
4. XGBoost: XGBoost is an advanced implementation of the gradient boosting algorithm, known for its efficiency and speed. It is designed for large datasets and is suitable for real-time predictions, which makes it suitable for this problem. XGBoost includes regularization terms in its objective function, helping prevent overfitting and improving model generalization. Since the dataset seems to be imbalanced, XGBoost allows to set weights for each class, meaning that it is a flexible algorithm in handling imbalanced datasets.

## Testing the machine learning applications

Training and validating the machine learning models used will be conducted through K-Fold Cross Validation. K-fold cross-validation is a method used to assess predictive models by dividing the dataset into k subsets or folds (Pandian, 2023). The model is trained and evaluated k times, with a different fold as the validation set in each iteration. Performance metrics from each fold are averaged to estimate the model's generalization performance. This technique is valuable for model evaluation, selection, and hyper parameter tuning, providing a more dependable measure of a model's effectiveness (Pandian, 2023). During each iteration, both training and testing are carried out precisely once, helping prevent overfitting (Pandian, 2023). Instead of training a model using all the data in one go, k-fold cross-validation ensures the model is more generalized. The process involves splitting the dataset into three sets: Training, Testing, and Validation, with considerations for the data volume.

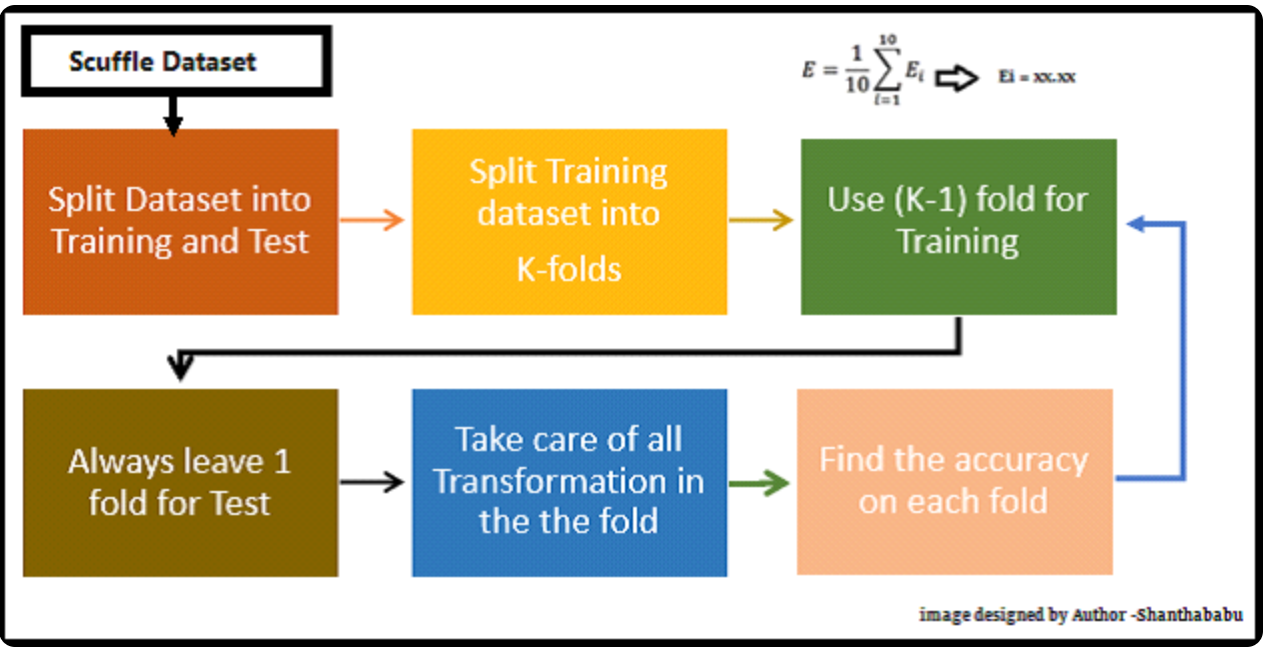
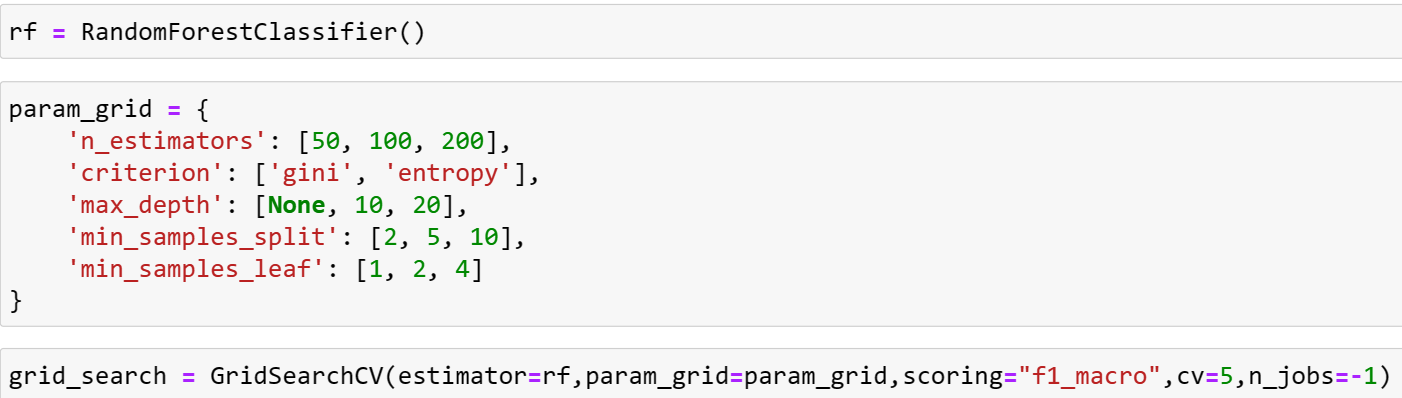


Figure 1 Life Cycle of K-Fold Cross-Validation (Pandian, 2023)

I performed a 5-fold cross-validation using GridSearch from Sklearn. GridSearch enables both cross-validation and hyper parameter tuning. It systematically cross-validates each combination of hyper parameters, and ultimately, the GridSearch returns the optimal hyper parameter combination that maximizes the chosen metrics.

For example, the following shows the GridSearch function that I have used for the Random Forest Classifier:



As shown in the picture, 5-fold cross-validation was used on the Random Forest Classifier, where the parameters shown in the pictures were optimized. The steps of the process are as follows (Pandian, 2023):

1. Splitting the data set into training and testing data.
2. Split the training data into K folds, in our case will be 5 folds.
3. Iteratively train and evaluate each combination of the model’s hyperparameters 5 times.
4. In each iteration, a different fold is used as the validation set, and the model is trained on the remaining folds
5. Performance metrics are saved for each iteration.
6. Identify the combination of hyperparameters that resulted in the highest performance.
7. Finally, the grid search returns the best hyperparameter combination and its associated performance metric.
8. The same approach was used for the remaining 3 models, with changes in the hyperparameters.

## Machine Learning Algorithms Used

### Random Forest

Random Forest is a supervised learning algorithm that falls under the ensemble learning technique, where multiple decision trees are used to create a strong model (RASTOGI, 2020). It is widely used for classification and regression tasks. Random Forest uses ensemble learning, meaning it builds multiple decision trees and combines their predictions to enhance the overall model's performance (RASTOGI, 2020). The fundamental idea is to reduce overfitting and improve generalization (RASTOGI, 2020).

To understand random forests, first I will explain decision trees. The Decision Tree is a supervised machine learning algorithm suitable for addressing classification and regression tasks with labeled data (Manav, 2020).

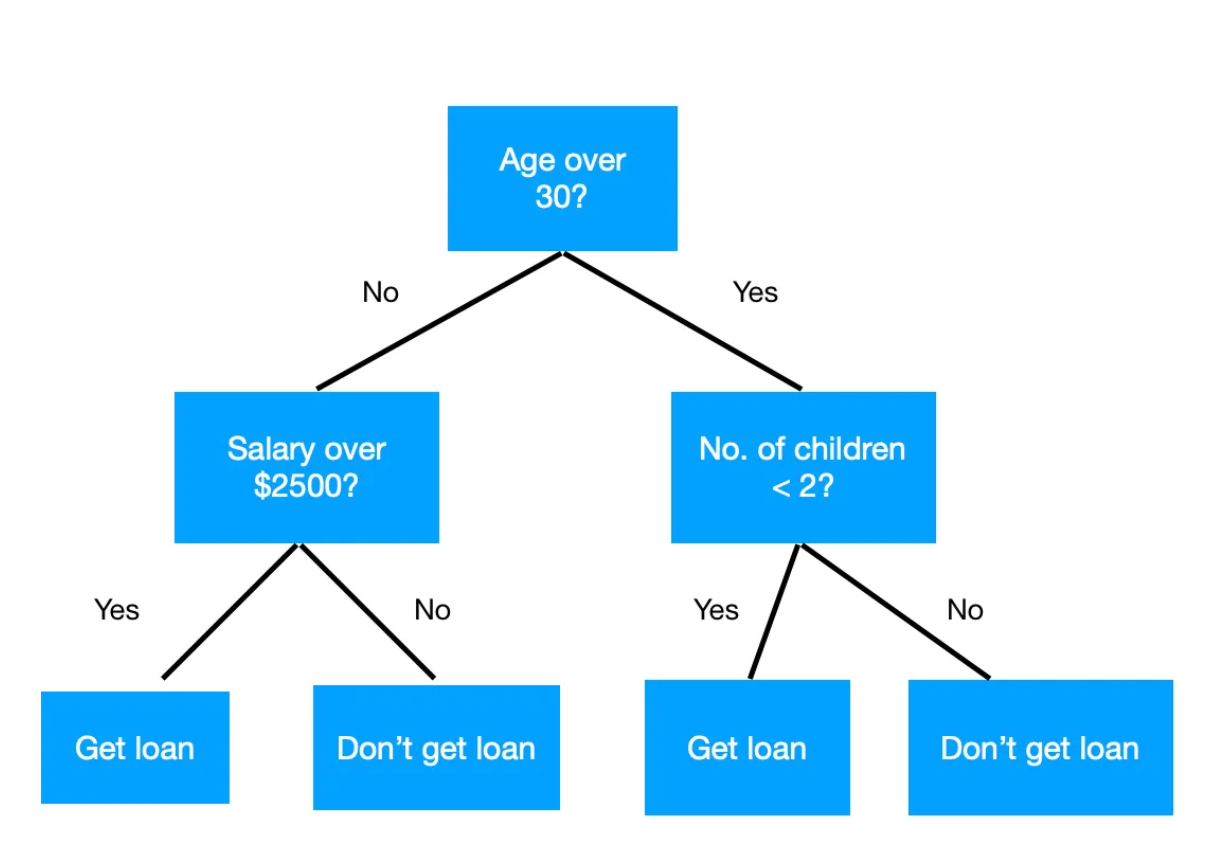


Figure 2 A simple decision tree (Manav, 2020)

The decision tree shown in Figure 2 shows that the loan approval process relies on factors such as age, salary, and the number of children a person has. At each node in the decision-making process, a specific condition is posed, leading to subsequent divisions, until a final decision is reached at the leaf node, determining whether the individual is eligible to receive a loan or not.

Terminologies in the context of decision trees and random forests: (Manav, 2020)

* Leaf Node: Nodes that don't divide are known leaf or terminal nodes.
* Splitting: It's the action of breaking a node into two or more smaller nodes.
* Parent and Child Node: A node that divides into smaller nodes is called the parent node of those smaller nodes, while the smaller nodes are the children of the parent node.
* Decision Node: If a smaller node breaks into even smaller nodes, it is referred to as a decision node.
* Depth: The depth of a decision tree indicates how many questions are posed before reaching the leaf node or final classification. The tree's depth is determined by its longest path.
* Entropy: Entropy quantifies the impurity or uncertainty present in a set of observations. In the context of evaluating the quality of a split, this metric is commonly denoted by E. Information entropy for a dataset with C classes, is calculated as follows:

Where is the probability of event , and is the number of different events or outcomes.

* Information gain: Information gain is a measure that helps in identifying the most valuable attribute within a set of training feature vectors for distinguishing between target classes. It guides us in determining the sequence of attributes in the nodes of the Decision Tree. Information gain is calculated as follows:
* Gini Index: Similar to entropy, which was used in this application. It quantifies how often a randomly chosen element in a set would be incorrectly classified based on the distribution of classes in that set.

A decision tree is built as follows: (Manav, 2020)

* Define features and the label of selected dataset.
* Calculate the Information gain and Entropy/Gini Index for each attribute in the dataset.
* Select the attribute showing the highest information gain as the decision root node.
* Calculate the information gain for the rest of the attributes in the dataset.
* Generate iterative child nodes by initiating the splitting process at the decision node, resulting in distinct child nodes for various values of the decision node.
* Iterate through this process until all attributes in the dataset are encompassed.
* Implement Tree pruning measures to mitigate overfitting risks.

Following up on random forests. Bagging, or bootstrap aggregation, serves as a method to diminish the variance in estimating a prediction function. It is effective for procedures with high variance and low bias, such as trees, where bagging is about fitting the same regression tree multiple times to bootstrap-sampled versions of the training data and averaging the outcomes for regression. In classification, a committee of trees contributes votes for the predicted class. Random Forests represent a significant modification of bagging. This approach constructs a large collection of de-correlated trees and averages their predictions. Random Forests introduce independence among trees, minimizing correlation, ensuring that strong features do not unduly influence predictions. With P features, the number of randomly selected features at each split (tree), denoted as m, is equal to √P, serving as a hyperparameter.

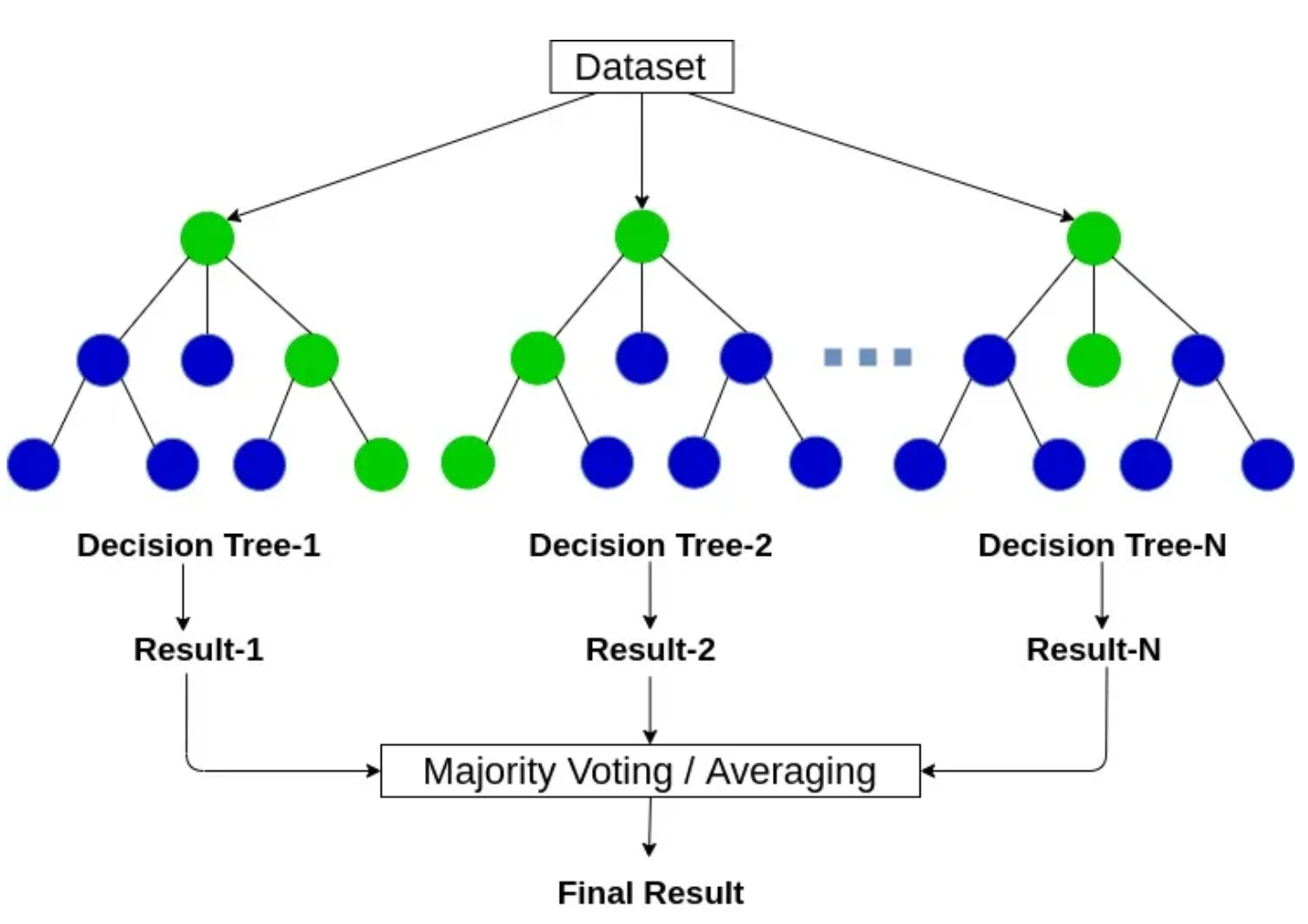


Figure 3 Random Forest Structure (RASTOGI, 2020)

The random forest algorithm works as follows: (RASTOGI, 2020)

* Select samples from the dataset.
* For each sample, construct a decision tree. Subsequently, obtain prediction results from each decision tree.
* Conduct voting for each predicted result in this stage.
* Designate the prediction result with the highest number of votes as the conclusive prediction result.
* Figure 4 provides a clear visualization of the process.

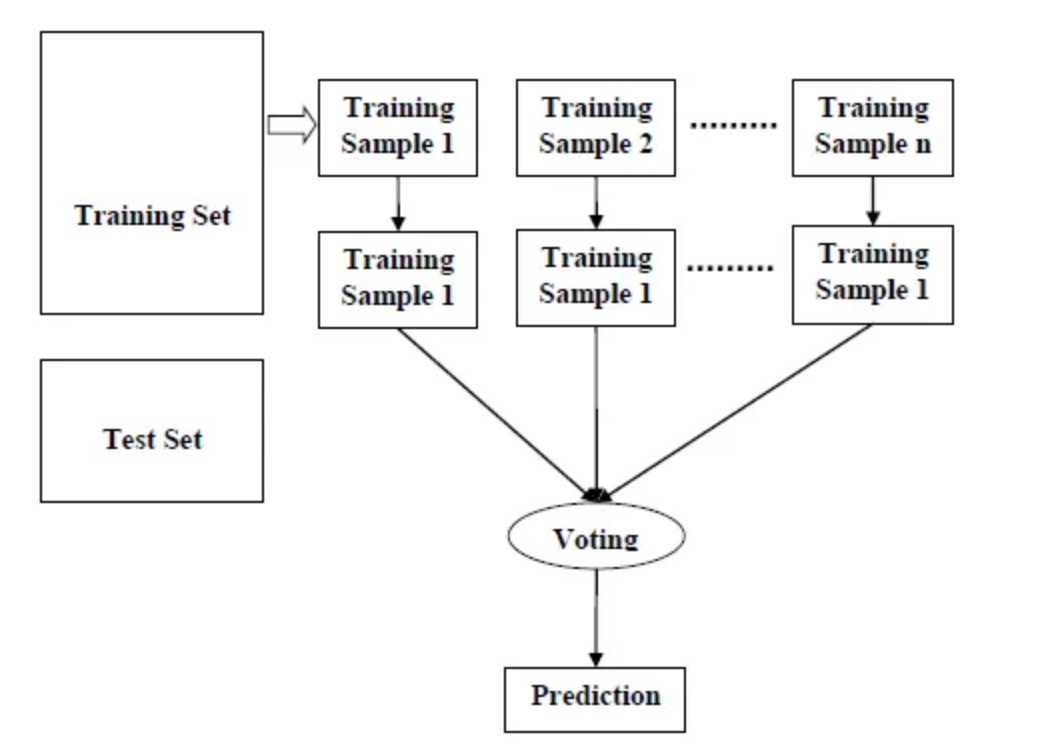


Figure 4 Random Forest Algorithm Mechanism (RASTOGI, 2020)

Random Forest algorithm pseudo code: (Hastie, 2009)

1. For each tree in the ensemble (b = 1 to B):
   1. Create a bootstrap sample (Z∗) by randomly picking N data points from the training set (to make each tree in the random forest different).
   2. Build a random-forest tree (Tb) using the bootstrap sample
      1. Randomly choose m variables from the total p variables.
      2. Determine the optimal variable/split-point among the selected variables.
      3. Recursively split nodes until the minimum node size (nmin) is reached.
   3. Get the ensemble of trees
2. Predict the outcome for a new point.
   1. If Regression:
      1. Combine the predictions from each tree in the Random Forest.
      2. Take the average of all these predictions.
   2. If Classification:
      1. Each tree provides its own class prediction for the new point.
      2. The final prediction is determined by a majority vote among all the individual tree predictions.
      3. The class with the most votes becomes the overall prediction for the Random Forest.

### Support Vector Machine (SVM)

The Support Vector Machine (SVM) is a powerful supervised machine learning algorithm utilized for both classification and regression tasks (Sharma, 2020). Its fundamental concept involves representing data points as coordinates in an n-dimensional space, where n is the number of features associated with each data point (Sharma, 2020).

SVM performs classification by determining an optimal hyperplane in the multidimensional space that effectively separates different classes (Sharma, 2020). Hyperplanes can be envisioned as decision boundaries in this space, classifying data points based on which side of the hyperplane they fall on.

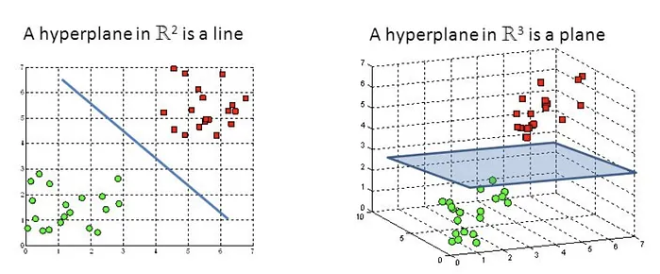


Figure 5 Hyperplane (Sharma, 2020)

The search for the best hyperplane involves considering the margins, which are the distances between the hyperplane and the closest data points from each class. The optimal hyperplane is the one that maximizes this margin (Sharma, 2020). In other words, it is the line with the largest possible gap between the two classes, ensuring robustness against slight deviations in the data. SVM is often referred to as a large margin classifier because it seeks to find the hyperplane with the maximum margin. The margin is calculated as the perpendicular distance from the hyperplane to the nearest data points. By emphasizing a large margin, SVM enhances its ability to generalize well to new, unseen data (Sharma, 2020).

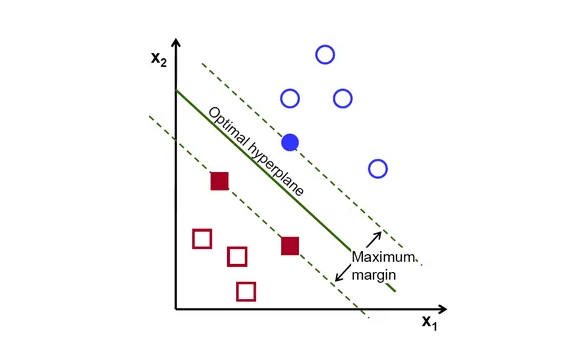


Figure 6 Optimal Hyperplane (Sharma, 2020)

SVM seeks a hyperplane that linearly separates the data points belonging to different classes (Sharma, 2020).

The training goal is to find a decision boundary that maximally separates different classes in the input space, not only classifying the data correctly but also by maximizing the margin, which is the distance between the decision boundary and the nearest data point of any class. (Medium, 2019).

Dealing with numerous dimensions, each potentially involving complex calculations, can be computationally intensive when considering every vector in the dataset (Medium, 2019). However, SVMs don't actually require the explicit vectors for their operations; they can effectively operate solely with the dot products between vectors. To leverage this in SVM classification, we use a kernel function, which essentially replaces the standard dot product (Medium, 2019).

In this application, hyper parameter tuning resulted in the polynomial kernel being the best for the problem, which is given by (Hastie, 2009):

The pseudo code for the SVM algorithm is as follows (Medium, 2019) (Hastie, 2009):

1. For i until convergence:
   1. For each training example (xi, yi):
      1. Calculate the decision function for all support vectors (j)

f(xi) = Σ (alpha\_j \* y\_j \* K(xi, x\_j)) + b

* + 1. Update alpha based on the optimization objective

if yi \* f(xi) < 1:

alpha\_i = alpha\_i + learning\_rate \* (1 - yi \* f(xi))

* + 1. Update bias term

b = b + learning\_rate \* Σ (yi - Σ (alpha\_j \* y\_j \* K(xi, x\_j))), for all support vectors (j)

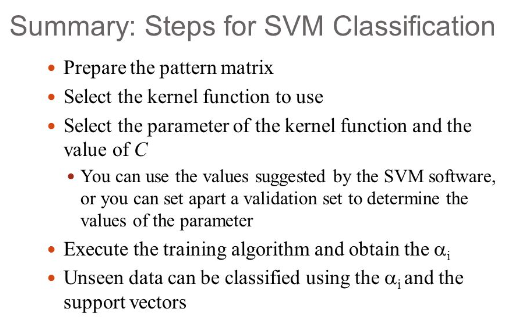
1. Make Predictions
   1. For each new example:
      1. Calculate the decision function

f(x\_new) = Σ (alpha\_j \* y\_j \* K(x\_new, x\_j)) + b

* 1. Make a prediction based on the sign of the decision function

prediction = sign(f(x\_new))

A simpler summary is as follows (Alward, 2014):



### Gradient Boosting

Gradient boosting is a widely used machine learning algorithm, especially for tabular datasets. Its robust capabilities extend to uncovering complex nonlinear connections between the target variable and features (Masui, 2022). Moreover, it demonstrates exceptional versatility, adeptly handling challenges such as missing values, outliers, and high cardinality categorical values in features without necessitating special handling (Masui, 2022). Gradient boosting is an ensemble method that involves the creation of multiple weak models, strategically combined to enhance overall performance, where every model trains with the objective of minimizing the residual error from the previous models (Masui, 2022).

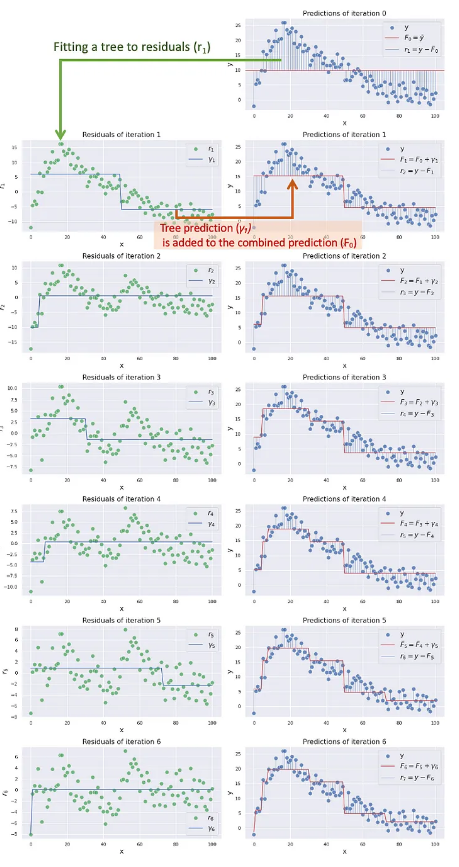


Figure 7 Fitting Decision Tree to Residuals over iterations (Manav, 2020)

A smaller value of the learning rate results in a larger number of trees, making the computation more intensive (Masui, 2022). Conversely, increasing the learning rate leads to fewer trees and quicker convergence (Masui, 2022). However, this faster convergence may result in a less robust model, as it doesn't have the opportunity to gradually approach optimal values.

The pseudo code of the used gradient boosting algorithm is as follows (optimized by gradient descent) (Masui, 2022) (Howard, 2018):

1. Initialize a model
2. For m = 1 till M:
   1. Calculate the residuals by computing the derivative of the loss function concerning the previous prediction
   2. Train a tree (weak learner) to predict r using features X and create terminal node regressions for each terminal node
   3. Find the value that minimizes the loss function on each terminal node.
   4. Update the model prediction
3. Final Model Predication

### XGBoost

eXtreme Gradient Boosting (XGBoost), is an ensemble learning machine learning algorithm that employs decision trees as fundamental models and integrates regularization methods to improve its ability to generalize (Analytics Vidhya, 2024). XGBoost is known for its computational efficiency, feature importance analysis, and handling of missing data.

XGBoost is a popular algorithm for supervised learning, particularly in regression and classification. It constructs a predictive model by aggregating the predictions of numerous individual models, predominantly decision trees, through an iterative process (Analytics Vidhya, 2024). The algorithm follows a sequential approach, gradually incorporating weak learners into the ensemble. Each new learner aims to reduce errors made by its predecessors, and the optimization process involves gradient descent to minimize a predefined loss function during the training phase (Analytics Vidhya, 2024).

Within the boosting framework, trees are constructed sequentially, with each subsequent tree focusing on minimizing the errors of the previous one. The learning process involves each tree adapting to the updated residuals left by its predecessors, ensuring that the next tree in the sequence learns from the latest version of the residuals (Analytics Vidhya, 2024).

Regularization terms are added to the objective function during the training process to prevent overfitting and improve the generalization ability of the model (Yehoshua, 2023).

The pseudo code of the XGBoost used in this application is as follows (Yehoshua, 2023):

1. Initialize the ensemble model
2. For b = 1 to B:
   1. Calculate negative gradient (residuals) for each sample
   2. Calculate residuals for each sample
   3. Fit a regression tree to the negative gradient
   4. Fit a tree to the negative gradient
   5. Calculate tree prediction for each sample
   6. Calculate the weight of each sample i in the terminal node
   7. Update the ensemble model
3. Final ensemble model

# Evaluation

## Performance Measures Used

I have used accuracy, macro averaged f1 score, and confusion matrices to assess the performance of my models. The accuracy metric provides an overall measure of how well the model predicts the number of occupants. It is the ratio of correctly predicted instances to the total instances. The F1 score is the harmonic mean of precision and recall. The macro-averaged F1 score calculates F1 scores for each class independently and then takes the average. This metric is useful for imbalanced datasets. Confusion matrices provide a detailed breakdown of the model's predictions, showing the number of true positives, false positives, true negatives, and false negatives for each class.

Also, I have calculated the time taken by each model to get trained, tuned, and cross validated, to determine differences in computation time between all four models.

## Justification of Performance Measures Used

* Accuracy: The primary purpose of the accuracy measure is to provide an overall assessment of how well the model predicts the occupancy count across different classes. It gives a general sense of the model's correctness in predicting whether the room is occupied or not. Also, Accuracy is a straightforward metric that is easy to understand and interpret, especially for non-technical users and stakeholders (Huilgol, 2019). Although I will take accuracy into consideration, I will not rely on it when assessing the models due to the imbalanced nature of the dataset.
* F1: The F1 score is a suitable evaluation metric for this problem due to its ability to handle imbalanced classes and strike a balance between precision and recall (Huilgol, 2019). With a multi-class classification problem, F1 score can be calculated for each class independently and then macro-averaged for an overall assessment. This is particularly valuable when both false positives and false negatives hold significance in practical applications, such as energy management and security in a room. The F1 score's interpretability and sensitivity to both types of errors make it a suitable metric for evaluating the model's performance in this context. Particularly speaking, the macro-averaged F1 score is more suitable than other types of F1 scores, especially in problem, due to its ability to treat each class equally and provide a balanced evaluation across different occupancy levels. The macro-averaged F1 score avoids bias towards majority classes, offering a more representative assessment, where it ensures that the evaluation considers the effectiveness of the model across all occupancy classes, irrespective of their prevalence in the dataset.
* Confusion Matrix: Using confusion matrices is crucial, especially in this multi-class classification problem, as it provides a detailed breakdown of the model's predictions for each class. Also, confusion matrices offer insights into the specific types of errors the model is making, revealing the nuances of misclassifications across different classes (H2o.ai, 2023). This will help in understanding where the model tends to over predict or under predict which can be beneficial for hyper parameter tuning and improving models’ performance. Also, confusion matrices can be used to explain the results for stakeholders, which will help them identify which classes or cases may pose more challenges.
* Time: Measuring training time is essential for this problem, as multiple models are being tested. It provides valuable insights into the computational efficiency of each algorithm. The efficiency of model training directly impacts real-world applicability, making it essential for practical deployment. Understanding the time taken for training, tuning, and cross-validation allows for informed decision-making in selecting the most time-efficient and resource-effective model. Also, measuring training time helps strike a balance between model performance and computational efficiency, helping in the selection of the most effective model for deployment in such time-sensitive applications.

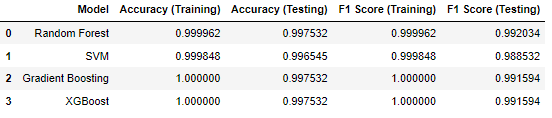
## Evaluation of Metrics and Enhancing Models

Based on the performance measures, including accuracy, F1 score, and training time, I conducted a comprehensive evaluation of four models: Random Forest, SVM, Gradient Boosting, and XGBoost, and enhanced the models based on these performance measures as follows:

* I have fitted a dummy classifier on the data set. The dummy classifier provided a baseline performance benchmark, allowing for a comparison of more complex models against a simple, naive approach. The dummy classifier's performance metrics helped gaining insights into the minimum level of predictive power achievable by a basic model. This baseline comparison served as a reference point, highlighting areas where advanced models must outperform the dummy classifier to be considered effective. Additionally, the dummy classifier helped in identifying class imbalances and potential pitfalls in model evaluation, guiding the enhancement of more complex models.

The dummy classifier resulted in an accuracy of 0.8124 suggesting that the model is correctly classifying the majority class. Also, the dummy classifier resulted in an F1 score of 0.224, indicating a struggle to balance precision and recall, likely making a significant number of false positives and false negatives, indicating a lack of discriminatory power in its predictions.

* After conducting SMOTE oversampling and K-Fold cross validation on more complex models, the results were as follows:



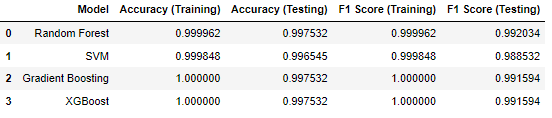
These results indicate that all four models are highly successful in predicting the target variable in the data set, due to their high predictive power, oversampling technique, cross validation, and the hyper parameter tuning.

* The SMOTE oversampling increased the performance measures of all four models, but the SVM f1 score, was extremely improved after using the SMOTE, from 0.79 to 0.988.
* Accuracy and F1 helped decide the parameters I was going to test on the models, to find a balance between model performance, and computation time.
* The confusion matrices played a crucial role in understanding how each model predicted on individual classes, allowing for a detailed analysis of misclassifications and areas for improvement. Also, it helped in understanding and interpreting the results of the models.
* The training time of each model influenced decisions regarding hyper parameter tuning and the size of cross-validation sets, helping strike a balance between model performance and computational efficiency.
* The main methodology that helped me enhance the model is the grid search. I used it with 5-fold cross validation to test a multiple combination of hyperparameter on 4 different algorithms, to find the optimal combination of these hyperparameters for each model. As each combination was tested and validated on the validation set formed on each different fold, and the combination with the highest scoring will be chosen at the end of the grid search.

# Results and Discussions

## Reliability of Results

The results suggest that the models, including Random Forest, SVM, Gradient Boosting, and XGBoost are as follows:



All four models demonstrate high F1 scores on both training and testing sets. The high training F1 scores, close to or at 1.0, indicate that the models have learned the training data well, capturing its underlying patterns, and the model successfully found a high balance between the precision and the recall.

Looking at the training F1 scores, the results are high, meaning that the model learned the data effectively, and there is no sign of under fitting.

Looking at the testing F1 scores, which remain high but slightly lower than their training counterparts, suggests good generalization, which means the results are reliable and there are no signs of overfitting. This means that all four models actually learned the relationships and pattern, as evident when comparing the results of the training sets and testing sets.

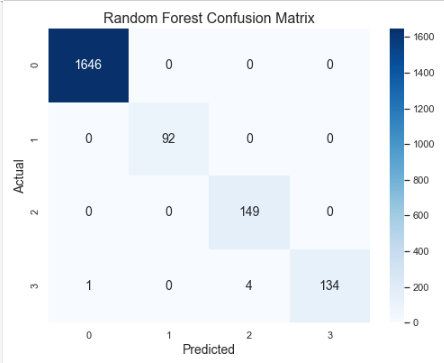
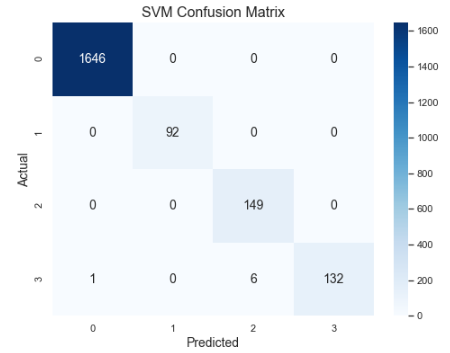
Considering bias and variance, the models seem to have found a balance between bias and variance.

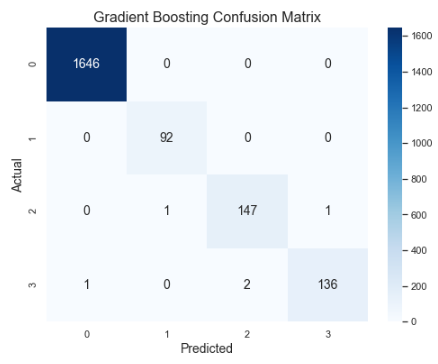
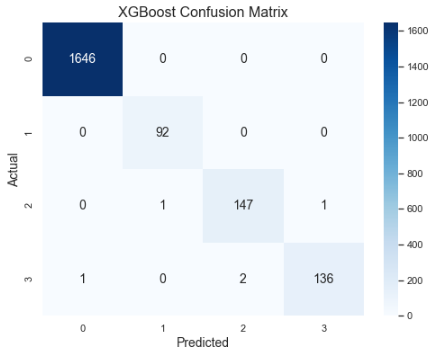
In conclusion, all four models seem to generalize well to unseen data, as indicated by high testing scores, which suggests that the models are well-trained and seem to be balanced.

## Analysis of the Results, and Algorithms Effectiveness

The analysis of the models’ performance measures reveals several key insights. All four models scored high scores on both training and testing sets, indicating that the models effectively learned the underlying patterns in the training data. The drop in performance on the testing set suggests good generalization and balanced models that avoid overfitting, as evidenced by the slight reduction in F1 scores

The confusion matrices further illustrate the models' ability to make accurate predictions across different classes.

****

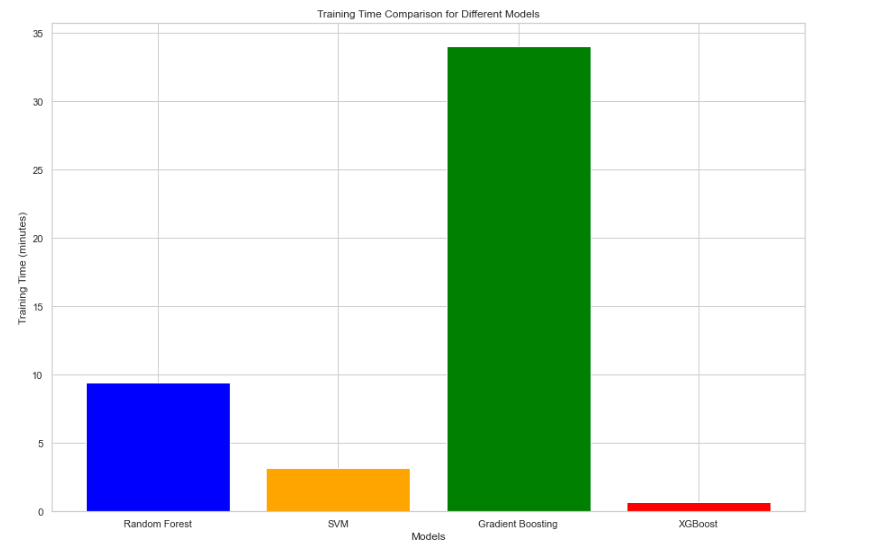
**** ****

Random forest exhibits strong predictive performance, with most values on the diagonal, indicating correct predictions. The slight off-diagonal values suggest a small number of misclassifications, specifically in classes 3 and 4.

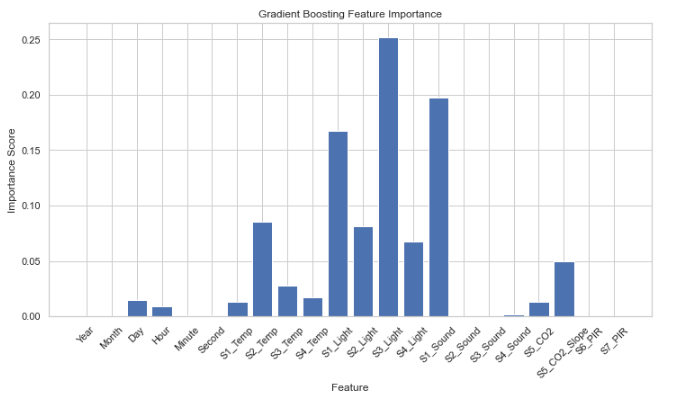
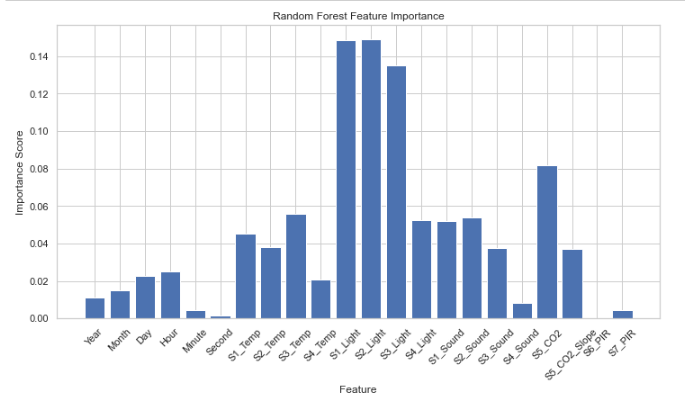
The SVM confusion matrix shows high diagonal values, signifying accurate predictions. However, there are a few misclassifications, particularly in predicting instances of Class 4.

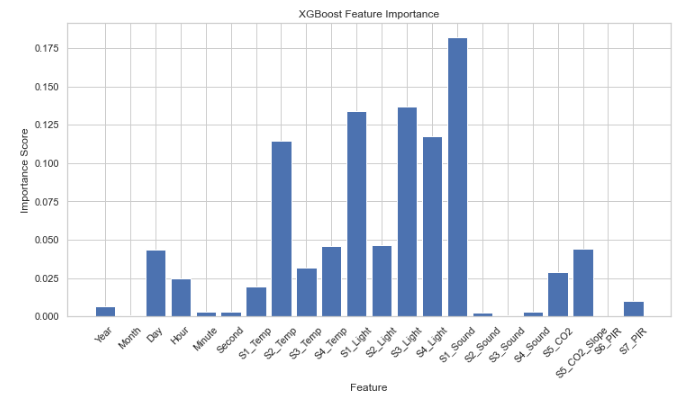
The confusion matrix for Gradient Boosting also indicates accurate predictions, with most values on the diagonal. However, there are a few instances where misclassifications occurred specifically in classes 3 and 4.

The training times vary, with Random Forest taking 9 minutes, SVM 3 minutes, Gradient Boosting 34 minutes, and XGBoost 2 minutes. This indicates that Gradient Boosting takes the most computation time, dramatically varying from other models. The XGBoost was the most effective in terms of training time.



Feature importance analysis from Random Forest, Gradient Boosting, and XGBoost highlight key contributors. These insights suggest that while the models generalize well and performed in a balanced way, careful consideration is needed to what features to use while training the models.





In the Random Forest model, the three most crucial features are S1\_Light, S2\_Light, and S3\_Light, emphasizing the significance of light sensors in accurately estimating occupancy. The least important features include Seconds, S6\_PIR, and S7\_PIR, suggesting that these variables have limited impact on the model's predictions. For the Gradient Boosting model, the most influential features are also related to light sensors (S1\_Light, S3\_Light), with an additional emphasis on S1\_Sound. In contrast, the least important features involve sound sensors (S2\_Sound, S3\_Sound) and temporal variables (Year, Month, Minute, Second) as well as PIR sensors, indicating their relatively lower impact on the model's predictive performance. The XGBoost model emphasizes the low influence of sound sensors (S3\_Sound), PIR sensors (S6\_PIR), and temporal information (Month).

## Strengths and Weaknesses of the Used Algorithms

Table 1 Algorithms: Strengths & Weaknesses

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Strengths** | **Weaknesses** |
| Random Forest | * High accuracy and F1 scores on training and testing sets. * Feature Importance analysis capabilities * Robust to noise and outliers | * Slightly longer training time compared to SVM and XGBoost. * Many hyper parameters that need to be tuned. * Requires careful hyper parameter tuning. * Could be computationally expensive during training |
| SVM | * High accuracy and F1 scores on training and testing sets. * Efficient training time | * Some misclassifications, particularly in predicting Class 4 instances. * Highly sensitive to the distribution of the data, especially imbalanced data * Requires careful hyper parameter tuning. |
| Gradient Boosting | * High accuracy and F1 scores on training and testing sets. * Feature Importance analysis * Builds a strong predictive model by combining weak learners. | * The need for high computation power, and the longest training time. * The model might be too complex for the data. * Requires careful hyper parameter tuning. |
| XGBoost | * High accuracy and F1 scores on training and testing sets. * Feature Importance analysis. * Efficient training time * Improved version of Gradient Boosting * Implements L1 and L2 regularization to reduce model complexity. | * Requires careful hyper parameter tuning. |

In conclusion, all four models demonstrate compatibility with the dataset, showcasing strong predictive capabilities with high accuracy and F1 scores. Random Forest, SVM, and XGBoost exhibit efficient training times, but Gradient Boosting requires more computational resources. Feature importance analyses provide valuable insights for tuning models.

## Enhancements and Limitations

The project demonstrates notable achievements in predicting room occupancy based on sensor data, yet it also presents several limitations and opportunities for future enhancements.

One significant challenge lies in the varying patterns and unequal distribution of data points across different labels. This issue can impact the reliability of the models, and addressing it requires strategies for handling imbalanced datasets. In this project, I have employed the SMOTE oversampling technique that looks to be very effective, however, it is recommended to test the model against other oversampling techniques.

Furthermore, using features like light raises concerns about their suitability for occupancy prediction, given scenarios where the light may remain on even in the absence of occupants. To overcome this, conducting feature selection or engineering techniques may be beneficial. Also, consulting with domain experts can also help draw insightful conclusion about the features.

The size of the dataset might be another challenge, as a larger dataset could provide a more comprehensive foundation for training strong accurate models. In other words, the data was collected on a time period of 6 days, and 6 days might not be enough to include most possible patterns in the context of the problem. Real-time experiments could offer insights into the models' adaptability.

To address these limitations, several enhancements and future improvements can be considered. Firstly, utilizing a larger dataset is recommended to train the model on more diverse patterns. A larger dataset provides a more representative sample of real-world scenarios, enabling the models to learn and generalize better. Exploring a variety of machine learning models, including less complex and simpler ones, is advisable. This helps in assessing the trade-off between model complexity and performance, avoiding unnecessary intricacies.

Regularization techniques such as L1 and L2 regularization can be employed to simplify complex models like XGBoost, and control the use of features.

Dimensionality reduction techniques are another suggestion for exploration, aiming to reduce the computational complexity of machine learning algorithms and enhance training efficiency. A further EDA could help identify the features that significantly impact model performance, guiding further feature engineering and refinement. Leveraging insights gained from feature analysis by Random Forest, Gradient Boosting, and XGBoost, testing models by dropping the least influential features can provide a streamlined set of features for optimal model performance.

These considerations aim to enhance model accuracy, reliability, and real-world applicability.

# References

Alward, E., 2014. *SlidePlayer.* [Online]   
Available at: https://slideplayer.com/slide/4044039/

Analytics Vidhya, 2024. *Analytics Vidhya.* [Online]   
Available at: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/

H2o.ai, 2023. *H2o.ai.* [Online]   
Available at: https://h2o.ai/wiki/confusion-matrix/#:~:text=Benefits%20of%20Confusion%20Matrix&text=It%20shows%20how%20predictions%20are,relying%20solely%20on%20classification%20accuracy.

Hastie, T. T. R. F. J., 2009. *Random Forests. In: The Elements of Statistical Learning. Springer Series in Statistics..* 2nd ed. New York: Springer.

Hastie, T. T. R. F. J., 2009. *Support Vector Machines and Flexible Discriminants. In: The Elements of Statistical Learning.* 2nd ed. New York: Springer .

Howard, T. P. a. J., 2018. *explained.ai.* [Online]   
Available at: https://explained.ai/gradient-boosting/faq.html

Huilgol, P., 2019. *Medium.* [Online]   
Available at: https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2#:~:text=Accuracy%20is%20used%20when%20the,as%20in%20the%20above%20case.

Leventis, D., 2018. *Medium.* [Online]   
Available at: https://dimleve.medium.com/xgboost-mathematics-explained-58262530904a

Manav, 2020. *Medium.* [Online]   
Available at: https://medium.com/analytics-vidhya/decision-trees-explained-in-simple-steps-39ee1a6b00a2

Masui, T., 2022. *Towards Data Science.* [Online]   
Available at: https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502

Medium, 2019. *Medium.* [Online]   
Available at: https://ankitnitjsr13.medium.com/math-behind-support-vector-machine-svm-5e7376d0ee4d

Pandian, S., 2023. *Analytics Vidhya.* [Online]   
Available at: https://www.analyticsvidhya.com/blog/2022/02/k-fold-cross-validation-technique-and-its-essentials/#:~:text=K%2Dfold%20cross%2Dvalidation%20helps%20to%20mitigate%20the%20risk%20of,to%20perform%20on%20unseen%20data.

RASTOGI, R., 2020. *Medium.* [Online]   
Available at: https://medium.com/analytics-vidhya/random-forest-classification-and-its-mathematical-implementation-1895a7bb743e

Saman Taheri, P. H. A. R., 2022. *Model predictive control of heating, ventilation, and air conditioning (HVAC) systems: A state-of-the-art review,* Indianapolis: ScienceDirect.

Sankaranarayanan, C., Shaju, S. & Sukhwani, M., 2022. *Actor-Critic based Adaptive Control Strategy for Effective Energy Management,* Hammamet: IEEE.

Sharma, N., 2020. *heartbeat.* [Online]   
Available at: https://heartbeat.comet.ml/understanding-the-mathematics-behind-support-vector-machines-5e20243d64d5

Singh, A. P. et al., 2018. *Machine Learning-based Occupancy Estimation Using Multivariate Sensor Nodes,* Hyderabad: IEEE.

Utilities One, 2023. *Utilities One.* [Online]   
Available at: https://utilitiesone.com/optimizing-hvac-systems-for-balanced-energy-efficiency-and-acoustic-comfort

Yehoshua, D. R., 2023. *Towards Data Science.* [Online]   
Available at: https://towardsdatascience.com/xgboost-the-definitive-guide-part-1-cc24d2dcd87a