Sara Aljamal

Room Occupancy Estimation project

Machine

learning

Table of Contents

[**Introduction** 2](#_Toc157537519)

[**Methods** 6](#_Toc157537520)

[**Evaluation** 9](#_Toc157537521)

[**Results and Discussion** 12](#_Toc157537522)

[**References** 17](#_Toc157537523)

# **Introduction**

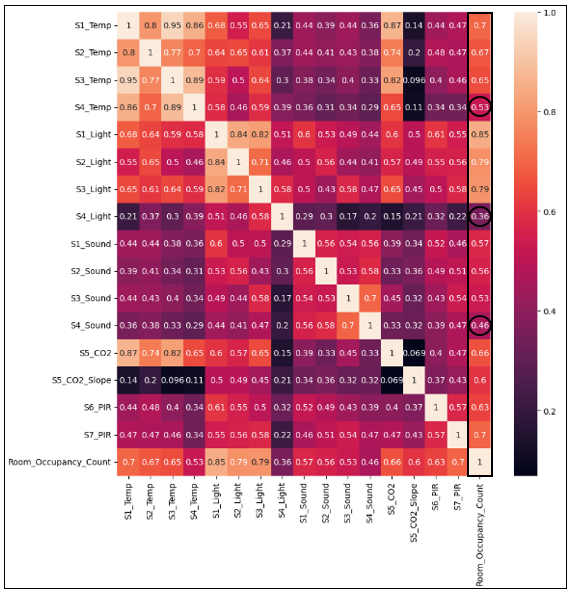
* **Describe the problem you are addressing and why is it important?**
* The energy consumed in heating, and ventilating rooms might sometimes be unnecessary where the occupant in the room differs or when the room is empty, estimating the number of occupants in a room through analysing data from a star configurated nodes that contains 7 slave nodes and 1 master node that all collected data is transmitted to it, each slave node had at least one sensor.
* Knowing the number of occupants in a room might optimize energy consumption rates and reduce cost. By controlling the energy that is used to heat and ventilate the room, we can control the temperature used to heat or cool the room, or the ventilation rate based on the demand for them which differs according to the number of occupants.
* **Describe the dataset's source, collection method, attributes, size, and domain.**
* **Source:** The dataset was collected for a study that aims to estimate the number of occupants in a room, created by Adarsh Pal Singh and Sachin Chaudhari. The dataset was published on UCI machine learning repository website which contains a large number of datasets, the link for the dataset can be found at the end of the refencing section.
* **Collection method:** They collected data through placing a network of 7 slave nodes and 1 master node in a 6m\*4.6m room that contained 4 desks, each desk on the four had a node that contained three sensors (Temperature, light, and sound), while node 5 which was placed in the middle of the room had a CO2 sensor, and nodes 6 and 7 that were placed on the door and the window respectively contained PIR sensors. Each node was connected to an Arduino Uno microcontroller that transmitted data to the master node every 30 seconds using Zigbee module. The duration for collecting the data was 4 full days (almost 96 hours), but they were distributed on 7 days.
* **Attributes:** The dataset contains 18 attribute, the date, time, and the other 16 attributes are represents the readings for each sensor of the 16 sensors, there are 4 temperature, 4 light, and 4 sound sensors (from nodes 1-4), in addition to the CO2 sensor on node 5, and the improved version of It using linear regression which is CO2 slope attribute, lastly, 2 PIR sensors on nodes 6 and 7. The label is the room occupancy count column which has values from (0-3).
* **Size:** The data contains 19 column and 10129 row (instances), where the duration between each instance and the one after in the same day is 30 seconds.
* **Domain:** Energy consumption and monitoring environments fields.
* **Describe the learning problem you are trying to solve.**

I will try to predict an estimate for the number of occupants in a room through applying 4 classification algorithms, Random Forest, SVM, gradient boosting, and XG Boost to sensors data.

The 4 algorithms will be trained on sensor data that comes from 7 nodes to a master node, these 7 nodes contain different sensors and are placed in different places around the room, such as temperature, sound, light, CO2, and PIR sensors. Each of the 4 algorithms will be applied twice, the first one using a baseline model, and the second time is through applying grid search with cross validation. The results for each model will be compared with each other, then the results from all models will be compared as well.

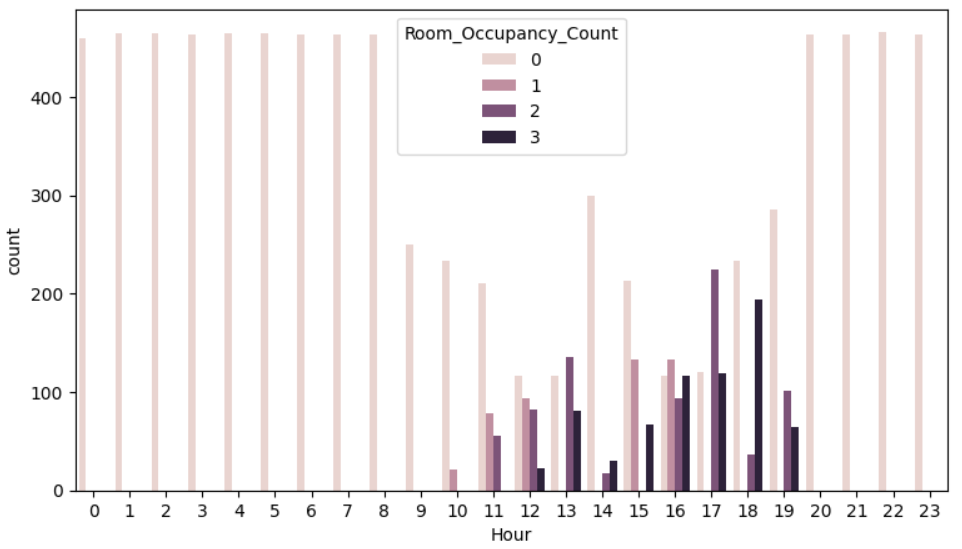
* **How did you prepare training and test data before implementing machine learning models?**

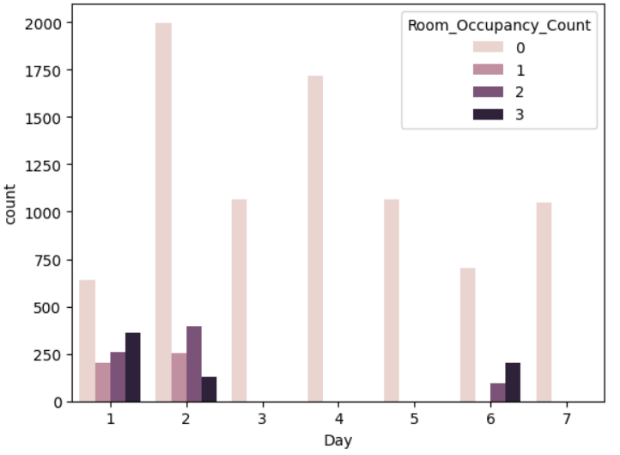
1. **EDA:**

* I started by overall exploration for the data, knowing the data types, checking null and duplicated values, and reviewing some statistics such as knowing that the data contains outliers in the columns for the ‘Light’, ‘sound’, and ‘CO2’ sensors.
* To explore the relationships between columns, I used the heatmap, and I noticed that the temperature and light sensors on nodes 1,2, and 3, in addition to PIR sensor on both nodes 6 and 7, and the CO2 sensor on node 5 have stronger relationships with the label (Room occupancy counts) that other sensors such as the sound sensors on nodes 1-4. Furthermore, the temperature, light, and sound sensors on desk 4, have the least correlation compared to other sensors.

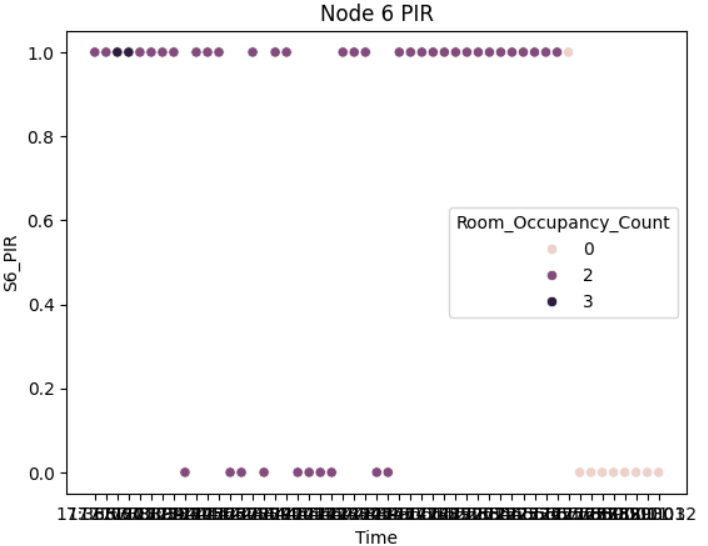
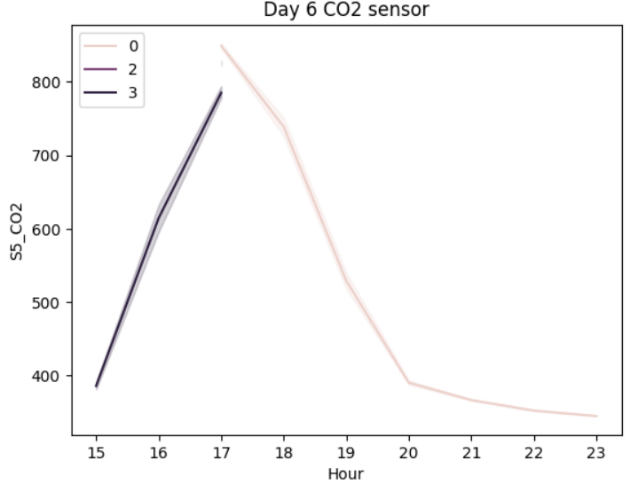
A graph of a number of objects

Description automatically generated

* Bar plot to see the counts of the values in the label from 0 to 3, almost 8200 out of 10129 were 0, while the remaining instances varied from 1,2, or 3. So the imbalance in the data is clear in the figure.
* I extracted a feature called ‘Hour’ from the time by taking only the hour part, I did that to explore if there are some hours that has more occupants that others, and I noticed that from 8 at night till 9 in the morning there is no occupants in the room in all days, but from 10-17 they range between no occupants, 1,2, or 3 occupants as shown in the figure below.
* Using the ‘Date’ column which contained only 7 dates, I extracted a new feature called ‘Day’ which has values from 1-7, each day represents a day. Using the ‘Day’ feature, I created a new data frame for each day, and after reviewing the room occupancy values counts in each data frame, I noticed that days 3,4,5, and 7 had no occupants at all, while days 1,2, and 6 varied for the number of occupants as shown in the figure below.



* A diagram of a number of dots

  Description automatically generatedFor further understanding for the PIR sensors placed on nodes 6 and 7, and to decide which one gives better results, I took a sample of 51 rows from day 6, which represents 25 minutes. I visualized the values of the PIR sensor on node 6 and on node 7 as scatter plots, if the PIR sensor returned 0 then it means it didn't detect any changes, while 1 means there are some changes in the infrared radiation which means there are people or occupants in the room. The colour of the points in the scatter plot indicates the room occupancy count for the observation. The PIR sensor on node 6 detected more correct points than the one on node 7, the location can be the reason for that, because node 6 was placed on the door while node 7 was placed on the window at the end of the room. The figures below show the difference in detecting the same points in node 6 and node 7.
* **A graph showing the temperature of a co2 sensor

  Description automatically generatedIn order to understand the effect of the CO2 sensor which is placed on node 5, first I reviewed the maximum and minimum values for it from the original data frame, after that I visualized the values of it using a line plot and I used the hour instead of time over 2 days, day 6 where the number of occupants varied, and day 4 where there was no one occupants all day long. We can clearly notice from the figure on the left that on day 6 where there were occupants in the room the CO2 levels was increasing, while when all occupants left, the CO2 levels began to decrease until it reached the normal values when no one is around which is less than 400. While on day 4 (the figure on the right), all values for the CO2 sensor were between 350 and 355 because no one entered the room. This shows that the level of the CO2 increases when there are occupants in the room, and it takes some time to get back to its normal levels after the room becomes empty.**

**2. Data preprocessing:**

* Removing unnecessary columns: I dropped the ‘Date’ and ‘Time’ columns, along with the ‘Hour’ and ‘Day’, The ‘Date’ and ‘Day’ are quite the same, and I think the effect of the ‘Hour’ column will be the same as the ‘Time’ column, so when building the models I tried both keeping the ‘Day’ and ‘Hour’ and removing them, and the results when removing them improved a lot and the models’ performance became better in most of the algorithms used.
* Normalizing data: I normalized all remaining columns except the PIR columns because their values are discrete and only 0 or 1. Normalization is important to ensure that no attribute gets more weight just because its scale is bigger, I used Minmax scaler as I am not sure if the data is normally distributed to use the Z score scaler.

# **Methods**

* **Explain why the provided models are appropriate to solve this problem.**

1. **Random Forest:**

* A ML algorithm that is suitable for classification and regression algorithms, so it works on the classification problem I have.
* Knowing that the data contains outliers, it’s important to choose algorithms that are not sensitive for outliers such as Random Forest.
* Random Forest is suitable for the dataset I have as the dataset is not large and Random Forest does not need a large dataset to give good results.
* Handles linear and non-linear relationships that exists in the data.

1. **Support Vector Machine:**

* Suitable for small but complex dataset, which aligns with the small dataset I have.
* Works on both classification and regression problems, so it can be used in our classification problem.
* The ‘kernel’ hyperparameter allow us to specify the type of the decision boundaries, which could be linear or not, and this is appropriate for the dataset as there exist non-linear relationships.
* Its performance gets better with scaled features which could be applied easily to our dataset as most of the features are continuous features.

1. **Gradient Boosting:**

* A boosting algorithm that combines weak learners sequentially where each weak learner corrects the errors made by the previous one, providing accurate results and more generalized model that could be used with another dataset for the same problem and performs better on unseen data.
* Has a lot of hyperparameter options allowing us to optimize the performance according to our dataset.
* Manage the bias and variance that could happen, especially the overfitting as the dataset is considered an easy dataset that can cause overfitting easily.

1. **Extreme Gradient Boosting (XG Boost):**

* Improved and faster version of Gradient Boosting, which may give the same or better results on our dataset but with reduced time and computational power.
* Has a built-in cross validation.
* **Demonstrate how you will test the machine learning application using a range of test data and explain each stage of this activity (Apply k-fold cross-validation).**
* After the preprocessing, I split the data to X and y, where X represent the predictors, and y is the response or the target variable.
* As I don’t have test dataset, I used a part of the training data to test the machine learning algorithms by using the ‘train test split’, I choose the splitting percentage to be 20% for testing that I’ll keep to test the machine learning algorithms and evaluate their performance, and 80% for training, to train the algorithms. The data will be split once before the modelling stage and will remain the same so all algorithms train on the same data for better and equal evaluation and comparison.
* For further testing and training, I will apply k-fold cross-validation along with the grid search for the 4 algorithms, this will be done by specifying the ‘cv’ in the grid search to 5, which means the grid search will apply 5-folds cross-validation. This means that for each combination in the grid search, cross validation will be applied.
* The full process will follow these steps in all algorithms:

1. Defining the model
2. Training the model
3. **Testing the model by predicting the values of the test data**
4. Evaluating the model performance of the train and test data using multiple evaluation measures.
5. Evaluating and comparing all algorithms performance.

* For each algorithm grid search will be applied to tune hyperparameters and choose the best one among the list of values for each hyperparameter, this process may improve the performance of the model on test data or unseen data.
* **Explain in detail the machine learning algorithms you are using to address this problem.**

1. **Random Forest:**

* An ensemble learning algorithm that combines weak learners (a collection of decision trees) to get improved, and more stable results, in addition to reducing the risk of overfitting.
* Random forest builds decision trees in parallel (Bagging algorithm) where each tree is trained on a different subset of features and observations that is selected randomly with replacement.
* Around 2/3 of the observations are used in training and building the decision trees, the remaining 1/3 is called Out-Of-Bag observations, OOB are observations that were not chosen in the training, thus, they could be used as a validation set to test the performance by calculating the Out-Of-Bag score.
* Works with both classification and regression problems, in classification, the final prediction is decided by the majority votes of all trees, while in regression the prediction is the average between all values that are predicted by all trees.
* Random forest has a lot of hyperparameters such as ‘n\_estimators’ which represents the numbers of trees in the forest, ‘criterion’ which is the function that evaluates the splits quality in the trees, and a lot more.

1. **Support Vector Machine:**

* An algorithm that could be used in classification or regression.
* Effective when the number of features > the number of observations, so it’s suitable for small, medium, but complex datasets.
* In classification problems, the objective of SVM is to find the hyperplane (decision boundary) that maximizes the margin, which is the difference between the two classes or more, support vectors are the closest datapoints to the hyperplane in which they determine the optimal position for it and maximize the margin.
* The hyperplane dimension differs according to the number of features, for example, if we have 3 features, then the hyperplane will be in 2 dimensions.
* ‘C’ is a hyperparameter that controls the margin (the street width), a small C value results in a wider street, but higher chances for margin violations (misclassifications).
* SVM can handle non-linear classifications, through adding more polynomial features the dataset becomes linearly separable, but it increases the runtime and the computational power needed.
* The kernel trick gives the same results as adding polynomial features but without adding them, which allow us to deal with complex datasets but with reduced the computational power and runtime.
* SVM could also be used in regression where we try to include as many possible datapoint in the street (inside the margin).
* Sensitive to feature scaling, so we need to scale the features before training the model.

1. **Gradient Boosting:**

* An ensemble boosting algorithm that minimizes both the variance and bias resulting in a more generalized model with reduced chances for overfitting or underfitting.
* Boosting means combining multiple weak learners such as decision trees sequentially, which means we build the trees according to the errors or misclassifications that are made by the previous one in order to correct them, by doing so, the model performance improve and the chance for misclassifications decrease.
* We start with initial prediction that is equal for all observations, and it might be calculated through the probability of log(odds) in classification.
* In gradient boosting, we calculate the residuals after each tree is build, and we feed the next tree by these residuals.
* Gradient boosting uses gradient descent to minimize the loss function.
* The prediction is done through combining the results of all trees after multiplying each output from each tree with the learning rate.
* Increasing the learning rate hyperparameter values results in a faster and fewer tree, but at the same time increased chance for misclassifications.

1. **Extreme Gradient Boosting (XG Boost):**

* An improved variation of gradient boosting that takes less time and improve performance.
* XG Boost parallelize the tree building process instead of building them sequentially.
* Reduces the runtime through using the CPU’s cache memory. Another efficient choice is running XG boost on GPU, as GPU's have massive number of small cores, which allow for parallel tree building on each core simultaneously.
* XG Boost have an automated cross validation that reduces the overfitting chances even more.

# **Evaluation**

Evaluate the effectiveness of the learning algorithms used by answering the following questions:

* **What performance measures did you use to evaluate the effectiveness of your models?**
* Accuracy.
* Marco Recall.
* Macro Precision.
* Macro F1 score.
* Confusion matrix.
* In our case, we have multi class prediction, so the Recall, Precision, and F1 score are calculated for each of the 4 classes alone, after that, it takes the average for all classes. There are many ways of calculating the average, the appropriate way is selected according to the problem and the dataset. Here are the most popular 3 ways:

1. Micro averaging: Indicates the overall performance regardless of the classes through calculating the metric globally by taking the total average of TP, FP, FN for the whole dataset. In micro averaging, the performance of the major class will have more significant impact on the average. Micro average calculation is similar to the standard calculation for each metric.

2. Macro averaging: It calculates the metric for each class and takes the unweighted mean for them, so it does not take the class weights or balancing into account, it treats all classes equally. Macro averaging is calculated through taking the sum of all metrics coming from each class and dividing it on the number of classes.

3. Weighted averaging: It calculates the metric for each class but while taking the mean it considers the class weight (number of actual observations for that class or frequency). It’s appropriate when we care about the major class and want to give it more weight when we want to evaluate the model’s performance.

* **Why did you use these metrics?**
* Because these measures are usually used for classification problems, and the reasons for using each one of them in the context of estimating room occupancy are explained below:

1. **Accuracy:**

* I used it as it measures the percentage of correct predictions made by the model regardless of the class.
* Accuracy is not suitable for imbalanced data such as our data because even if the model fails in predicting the minority classes, it will still give high values if it predicts the majority class which is class 0 correctly. As the accuracy is not enough in my case, I used other measures to ensure the model’s performance is measured correctly and appropriately.
* Even though accuracy is not a good enough measure in case of imbalanced data, but it helped me in detecting overfitting or underfitting, as I used it to see the train accuracy and test accuracy for each model, I could tell if the models are generalizing well and if the results are balanced, overfitted, or underfitted.

1. **Recall:**

* It measures the percentage of the how many positive actual values were predicted positive, in other words, out of all actual positive points, how many points the model predicted positive. So, I used it to give me information about the model’s performance with respect to false negatives (the actual positive values the model missed).
* I used Marco averaging in calculating the metrics, this is because the dataset have 4 classes and it’s imbalanced, so I didn’t want class 0 which is the majority class to get more importance in the calculation and results because in the case of estimating the room occupancy, it’s important to get accurate estimation for the number of occupants rather than predicting if there are occupants in the room (no matter of their number) or not, so all classes have the same importance and this should be taken into consideration when calculating the results.

1. **Precision:**

* It measures the percentage of correct positive predictions done by the model, so it calculates the percentage of the number of observations predicted positive that are truly positive. In our case, Precision for multiclass classification is measured through taking the average, referring to the averaging ways mentioned above, the Precision average might be micro, macro, or weighted.
* In calculating the Precision, I used macro-Precision as the data is imbalanced and I want a representative value for the whole performance on all classes that is not affected by the majority class.
* I used Precision as it shows if the model is being precise or not, high values indicate better preciseness for the model indicating better performance and reduced false positives in estimating the room occupancy count.

1. **F1 score:**

* I used F1 score because the accuracy might not be a good representative of the model’s performance in case of imbalanced data and because I am not interested in only one of the precision or recall, I want a measure which indicates both precision and recall at the same time.
* F1 score is the harmonic mean of both precision and recall.
* I used Marco F1 score specifically as it calculates the F1 score for each class independently, and it treats all classes equally which is suitable for the imbalanced data we have.

1. **Confusion matrix:**

* I used it after each model to show were the misclassifications happened.
* Through the confusion matrix, I can know the exact number of correctly classified and misclassified number of instances, which enables me to compare models when all of them have high accuracy and F1 measures.
* Another reason why I used confusion matrix is because I can see the exact performance of the model among classes, so I can see on which classes the model makes more mistakes or performs poorly, and on which classes the model performs great.
* And because I used Precision and Recall, after reviewing their values, I can use the confusion matrix to see were did the False positives and False negatives happens which enables me to understand and interpret the Precision and Recall values easily.
* **Evaluate how, based on the performance measures, you were able to enhance the model.**
* Depending on the values of the performance measures, I was able to decide whether I want to scale the data or not and why, first, I trained the models without scaling the data, then, I scaled the data and trained the models again, from the performance measures I used, I deciding to scale the data as it effects the model’s performance positively especially when it comes to SVM, because it’s sensitive to scaling.
* I also relied on the performance measures to decide whether I should drop the unnecessary columns or not, and after reviewing the results I decided to drop them because they affected the performance negatively, this results in reducing the model’s complexity and improving its performance.
* I tried balancing the data using class weights, but after recognizing that the metrics are not getting better, I removed the class weights as I thought that there is no need for additional unnecessary computation and complexity. So, the performance measures I used assists me in reducing the complexity, thus, enhancing the model.
* Another benefit of performance measures is that I could compare between the baseline for each model and the model using grid search do decide which algorithms work better with grid search and which don’t.

# **Results and Discussion**

* **Discuss the reliability of your results and whether they are balanced, overfitting, or underfitting.**
* The results of all algorithms I implemented could be described as balanced and great results.
* Overfitting happens when the model performs well on the training data but performs poorly on unseen data, while in underfitting the model performs poorly on training and testing data. In the algorithms I implemented I didn’t notice any overfitting or underfitting, when evaluating each model’s performance I used many performance measures, Accuracy was one of them, for each model I made sure to get the accuracy for the training and testing to detect overfitting or underfitting, but in all models I implemented, the Accuracy for both training and testing was great. In addition, I didn’t only rely on the Accuracy as the data is not balanced, so I used other performance measures such as Recall, F1 score, Precision, and Confusion matrix to detect overfitting or underfitting, because in both cases the model will perform poorly on the unseen data, but from the performance measures values for all models I ensured that there is no underfitting or overfitting in all models.
* **Analyse the result of the applications to determine the effectiveness of the algorithms.**

In my project, I implemented 4 algorithms, Random Forest (RF), Support Vector Machine (SVM), Gradient Boosting Machine (GBM), and Extreme Gradient boosting (XG boost), for each algorithm, I tried a simple model with the default hyperparameters from Sklearn, and a grid search to tune the hyperparameters and find the best values. The overall results were great, and all algorithms seemed to be effective solution for the problem, each point below analyse the results and effectiveness of each algorithm independently and compares the simple model with the grid search done for each. Lastly, a comparison between all models will be shown using the F1 score metric.

* **Random Forest:** At first, I tried the random forest classifier without any tuning for the hyperparameters, keeping the default hypermeter values such as 100 for the ‘n\_estimators’, ‘gini’ for the ‘criterion’, and ‘sqrt’ for the ‘max\_features’, the results were very good, it only misclassified 4 instances (2 instances with actual class 2, and the other 2 were from class 3). When I applied grid search with the RF, the parameter grid contained values for the ‘n\_estimators’, ‘criterion’, and ‘max\_features’ hyperparameters, the best values for them were 150, ‘entropy’, and ‘log2’ respectively, 3 instances were misclassified from the confusion matrix for the grid search (2 from actual class 2, and 1 from class 3). The table below shows the performance metrics for the RF and RF with grid search, the results are really close despite the difference of the hyperparameters values, this may indicate that the RF performs well on the dataset we have regardless of the hyperparameters values, both models’ misclassifications happened on classes 2,3 (where the number of occupants is 2 or 3.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Train accuracy | Test accuracy | Macro F1 score | Marco recall | Marco precision |
| RF | 1 | 0.998 | 0.993 | 0.993 | 0.994 |
| RF with grid search | 1 | 0.998 | 0.994 | 0.995 | 0.994 |

* **SVM:** In SVM, the model also performed great with the default hyperparameters which are ‘rbf’ for ‘kernel’, and 1 for ‘C’, the misclassifications also happened on classes 2 and 3, but their number was bigger than RF with total 11 misclassified instances. When performing grid search along with the SVM, I specified the ‘kernel’ and ‘C’ to choose the best value for each, 100 was the best value for ‘C’, and ‘linear’ was the best value for the ‘kernel’, the misclassifications in the grid search where 13, which is more than the normal model, for further comparison, I used the performance measures shown in the table below, although the differences between the 2 models are small, it’s clear the SVM performed better with the default hyperparameter without grid search, the value for ‘C’ hyperparameter might be the reason, in the first model, the ‘C’ value was 1, which indicates a wider and soft margin, but in grid search, it was 100, which can be considered as hard margin compared to the first model, and hard margin is sensitive to outliers which exist in our data, in addition it performs better when the data is linearly separable. Lastly, the Precision metric had the biggest difference in values, so it reflected on the F1 values, making the value for the SVM model better.

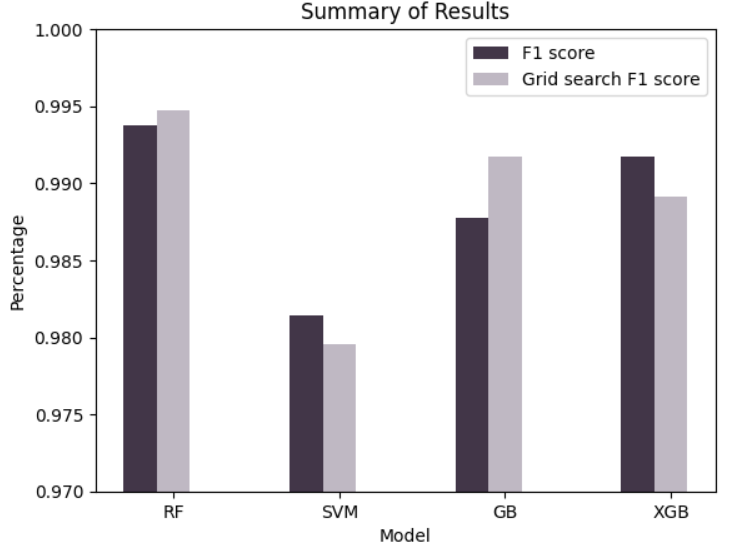
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Train accuracy | Test accuracy | Macro F1 score | Marco recall | Marco precision |
| SVM | 0.995 | 0.994 | 0.981 | 0.981 | 0.980 |
| SVM with grid search | 0.996 | 0.993 | 0.979 | 0.981 | 0.977 |

* **Gradient boosting:** There was a noticeable difference in the performance for the Gradient boosting with the default hyperparameters and the gradient boosting with the grid search where I specified 3 hyperparameters to be tuned, first, ‘n\_estomators’, ‘learning\_rate’, and ‘max\_depth’. The best ‘n\_estomators’ was 200, 0.1 for the ‘learning\_rate’ which is the same as the default value, and 8 for the ‘max\_depth’. The difference between the models’ performance is clear by the number of misclassifications made by each model, for the normal Gradient boosting, there was 10 misclassifications, while there was 6 in the grid search, also must of them were for classes 2 and 3. The values for other metrics displayed in the table also shows that the gradient boosting with grid search is better than without because all values increased in the grid search indicating better performance.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Train accuracy | Test accuracy | Macro F1 score | Marco recall | Marco precision |
| GB | 1 | 0.995 | 0.987 | 0.984 | 0.991 |
| GB with grid search | 1 | 0.997 | 0.991 | 0.989 | 0.993 |

* **XG Boost:** At first, I implemented a simple XG boost classifier without any hyperparameter tuning, and it showed very good results with only 6 misclassifications all between classes 2 and 3. In the grid search, I specified the same hyperparameters to be tuned that I specified in the gradient boosting as XG boost is an improved version of gradient boosting, the results for the best hyperparameters values were the same as the results in the grid search for gradient boosting except the ‘max\_depth’ which was 16 in XG boost. From the confusion matrix of the XG boost with grid search model, it’s clear that XG boost without grid search and with the default hyperparameters was greater than the XG boost with grid search as it made 8 misclassifications. Normal XG boost also showed better performance through the Precision and F1 scores.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Train accuracy | Test accuracy | Macro F1 score | Marco recall | Marco precision |
| XG Boost | 1 | 0.997 | 0.991 | 0.989 | 0.993 |
| XG Boost with grid search | 1 | 0.996 | 0.989 | 0.988 | 0.990 |

* **Comparing all models:** After analysing each model independently, it’s time to compare and evaluate their performance with each other, the Random Forest showed the best results among all other algorithms, with the least number of misclassifications in both normal and grid search models, XG boost comes after the Random Forest, it’s performance without grid search was better than with grid search with greater values for all performance measures. The gradient boosting comes after the XG Boost, with an improved performance using the grid search. Lastly, the SVM, it had the lowest values for the performance measures and the biggest number of misclassifications made by all models. It’s important to note that most misclassifications that happened in the models were in classes2 and 3, which may be an indication of the complex patterns between them which made it difficult for the models to differentiate between them sometimes. The figure shows the F1 scores for all 8 models implemented, it reconfirms that RF especially with grid search is the best one, followed by XG boost, gradient boosting, and lastly, SVM. It’s important to note that some algorithms perform better with grid search while others don’t such as the SVM and XG Boost.
* **Draw conclusions regarding the strengths and weaknesses of the different algorithms?**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Random forest | SVM | Gradient boosting | XG Boost |
| Strengths | 1. Fast, don’t need a lot of time.  2. Give accurate and great results.  3. Doesn’t need so much preprocessing. | 1. Don’t need a lot of hyperparameter tuning. | 1. Have good performance.  2. Have a lot of hyperparameter options that might improve performance. | 1. Faster than Gradient boosting.  2. Have a great performance.  3. Doesn’t need grid search for better results. |
| Weaknesses | 1. May need grid search (hyperparameter tuning) to give better performance.  2. Might be hard to interpret. | 1. Takes more time especially in the grid search.  2. Require scaling for the data.  3. Weak performance compared to other algorithms.  4. Doesn’t work on large datasets. | 1. May need grid search (hyperparameter tuning) to give better performance.  2. Takes a long time (The algorithms with longest run time) | 1. Might be complex to interpret, as it has built in cross validation, and because it’s a boosting method but build trees in parallel. |

* **Identify further enhancements which can be done in the future? ﻿Discuss any limitations and future improvements of your project.**
* To achieve better results, I could try manual hyperparameter tuning especially for the algorithms where grid search performance was worse than the normal model such as SVM, and XG Boost. Another way to improve the tuning results would be adding more hyperparameters in the grid search for each model.
* Other classification algorithms might be used to examine their performance on the data such as KNN and naïve Bais.
* We could also try applying neural networks for deeper analysis for the data, thus, better results, but in the current situation where ML algorithms are performing well and giving great results, I think there is no need to increase the complexity by adding neural networks, they could be added if the data in the future contain more sensors, and more observations.
* Because the data is imbalanced, techniques for balancing data might enhance the models, I tried class weights, but it was not effective, under sampling will reduce the data size so much, so applying oversampling might be the solution for the imbalance which might enhance the model’s performance and results.
* As there was no overfitting detected, adding more data to the dataset may improve some algorithms performance, but not for SVM.
* One of the limitations is that most misclassified datapoints among all models belong to actual class 2 or 3, which indicates that after combining all sensors data, we still have weak differentiation between these two classes, adding more sensors that their reading might enable us of differentiating between these 2 classes specifically might be an efficient solution.
* Another limitation that might have affected the algorithms’ performance is the data itself, it contains outliers, which effected some algorithms such as SVM.
* Finally, although there exist some further enhancements and some limitations, the project was successful, the performance measures and results were very high, and the algorithms performed well without any overfitting or underfitting detected.

# **References**

Singh, J. (2020) *Random Forest: Pros and cons*, *Medium*. Available at: https://medium.datadriveninvestor.com/random-forest-pros-and-cons-c1c42fb64f04 (Accessed: 21 January 2024).

K, D. (2023) *Top 4 advantages and disadvantages of support vector machine or SVM*, *Medium*. Available at: https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107 (Accessed: 21 January 2024).

*What is the advantage of gradient boosting?* (2022) *Deepchecks*. Available at: https://deepchecks.com/question/what-is-the-advantage-of-gradient-boosting/ (Accessed: 21 January 2024).

*What is Random Forest?* (no date) *IBM*. Available at: https://www.ibm.com/topics/random-forest (Accessed: 26 January 2024).

*Out of bag evaluation(oob) and OOB score or error in random forest* (2022) *YouTube*. Available at: https://www.youtube.com/watch?v=fXfs\_6o\_hOs (Accessed: 26 January 2024).

Gandhi, R. (2018) *Support Vector Machine - introduction to machine learning algorithms*, *Medium*. Available at: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (Accessed: 26 January 2024).

Saini, A. (2024) *Gradient boosting algorithm: A complete guide for beginners*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/ (Accessed: 26 January 2024).

*Gradient boost part 3 (of 4): Classification* (2019) *YouTube*. Available at: https://www.youtube.com/watch?v=jxuNLH5dXCs (Accessed: 26 January 2024).

N, Y. (2023) *F1 score*, *LinkedIn*. Available at: https://www.linkedin.com/pulse/f1-score-yeshwanth-n/ (Accessed: 21 January 2024).

*Precision vs. recall: Differences, use cases & evaluation* (no date) *V7*. Available at: https://www.v7labs.com/blog/precision-vs-recall-guide#:~:text=The%20macro%20precision%20and%20recall,a%20different%20number%20of%20samples. (Accessed: 26 January 2024).

*Sklearn.metrics.recall\_score* (no date) *scikit*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall\_score.html (Accessed: 26 January 2024).

**Dataset link:** https://archive.ics.uci.edu/dataset/864/room+occupancy+estimation.