## Introduction:

* **Describe the problem you are addressing and why is it important:**

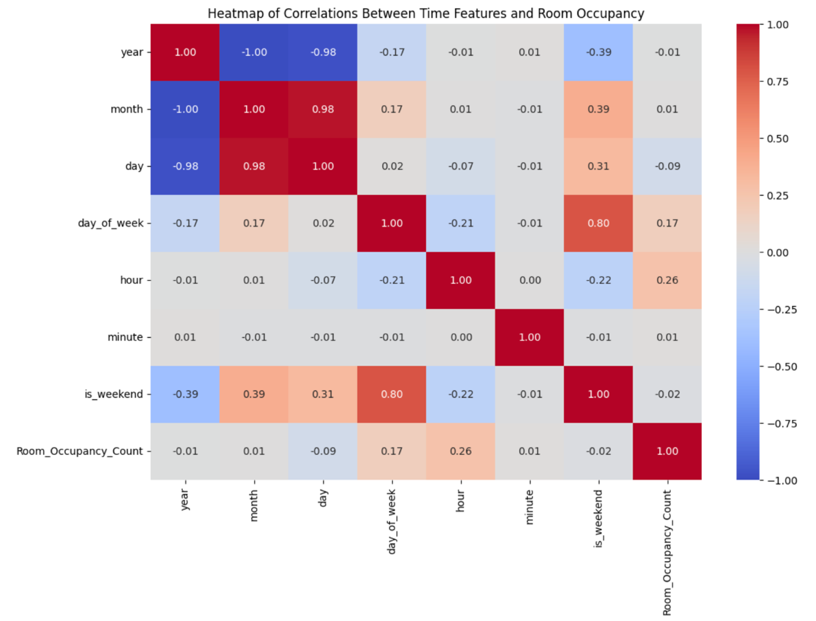
**The problem:** Based on what was addressed in the paper, the problem is the estimation of human occupancy in buildings, particularly in a room, using sensors, this issue is critical to the optimization of energy consumption, particularly for building HVAC systems (heating, ventilation, and air conditioning).

**The importance:** This issue is important since it can save energy. Reducing needless energy usage may lead to more effective operations of HVAC and lighting systems, so energy consumption can be reduced through accurate occupancy estimation. This will be useful for the economic and environmental sides since it can potentially lower buildings' carbon footprints.

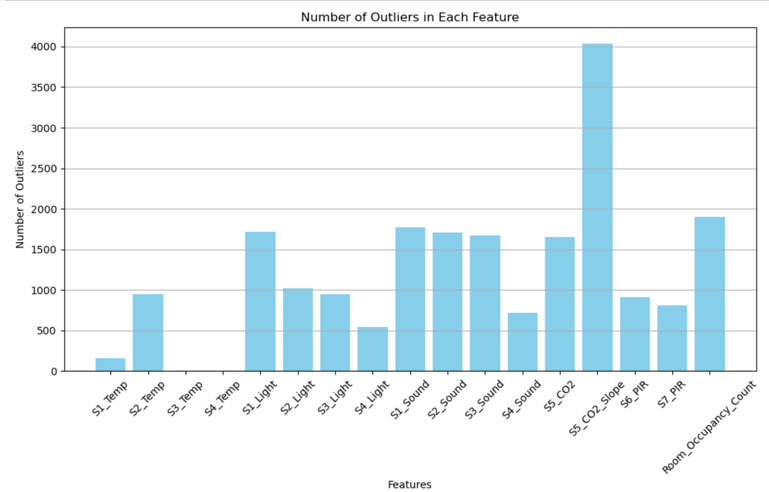
* **Describe the dataset's source, collection method, attributes, size, and domain:**
* **Source:** The first appearance of this Dataset was because of research and this research paper called “Machine Learning-based Occupancy Estimation Using Multivariate Sensor Nodes.”. This dataset was collected in a room using sensors, this room was located in The International Institute of Information Technology (IIIT) in Hyderabad, India. The resource that I used to get the data set was from a website “UC Irvine Machine Learning Repository”
* **Collection method:** This Dataset was collected using a Zigbee-based star network with sensor nodes that measured and recorded the environmental parameters such as CO2 levels, temperature, illumination, sound, and motion.
* **Attributes:** The dataset has records of multiple sensor nodes in a room, collecting various environmental parameters such as
  + **Data and time:** This attribute shows the time series when the data was recorded.
  + **Sound level (S1\_Sound to S4\_Sound):** This attribute records the sound level at different sensors that are located in the room so this will indicate an activity in the room.
  + **Temperature Readings (S1\_Temp to S4\_Temp):** This attribute shows the temperature recorded values from sensors S1 through S4, which are probably spread across the room.
  + **CO2 Concentration Levels (S5\_CO2):** CO2 levels are an important indicator of occupancy since they rise with the number of people in the space is high.
  + **Room Occupancy Count:** This shows the exact amount of people that are in the room at all times, and also it is the target feature.
  + **Lighting Levels (S1\_Light to S4\_Light):**
  + **CO2 Concentration Rate of Change (S5\_CO2\_Slope):**
  + **Motion Detection Readings (S6\_PIR and S7\_PIR):**
* **Size:** The dataset has 10129 rows and 19 columns.
* **Domain:** This dataset's domain is room-based environmental sensing, with a focus on factors related to human occupancy for estimation.
* **Describe the learning problem you are trying to solve:**

The learning problem which is in the project is a multi-classification problem. So, what is needed is to estimate the number of occupants that are in the room (range from 0 to 3) using the data that was recorded by the sensors, by using different types of classification models such as Random Forest, SVM, Gradient Boosting and XGBoost and then choosing which model fits and deals better with this data by evaluating the model prediction.

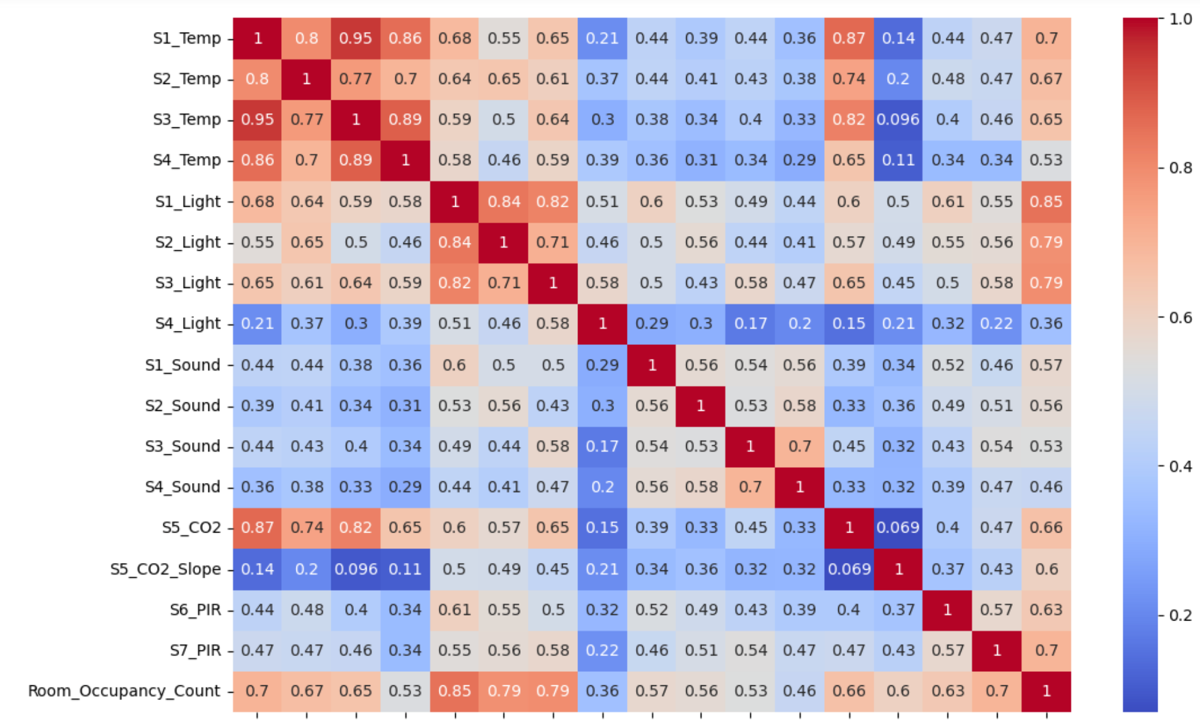
* **How did you prepare training and test data before implementing machine learning models?** 
  + **Dropping Date and Time columns:** Because these columns have a very small correlation with the target feature, which is Room Occupancy Count, I considered that after I changed the Date and Time features to numeric data type (**Encoding)** and used the correlation function.

****

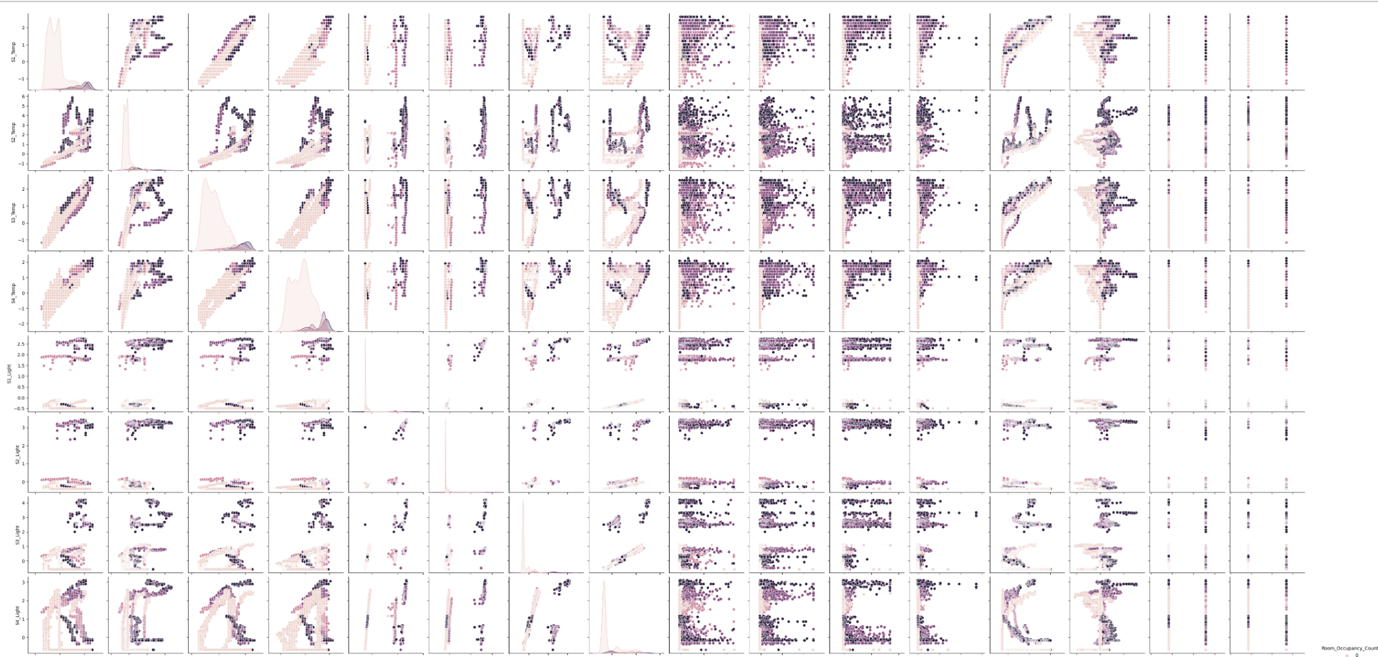
* + **Feature Scaling:** I used StandardScaler from sklearn. preprocessing to scale your features and make the values closer to each other (Normalized values), this step is so important because one of the models needed to be implemented is SVM, and this model is so sensitive.
  + **Excluding The Target Feature:** Putting all the data set in X variable except the target feature, and putting the target feature in Y variable
  + **Splitting The Data into Train And Test**: I used train\_test\_split from sklearn.model\_selection to divide your dataset into training and test sets so I can evaluate the performance of the model.
  + **Under Sampling:** I used Cluster Centroids which is provided by imbalanced-learn which is an extension of scikit-learn but it does not affect the overall performance of the models, so I removed it.
  + **Over Sampling:** I used SMOTE which is provided by the imbalanced-learn which is an extension of scikit-learn but it raised the running time so much and it did not affect the overall performance of the models, so I removed it.
  + **Feature selection:** I used Select KBest feature selection which is provided by scikit-learn but it does the overall performance of the models even when the Over Sampling and Under Sampling was applied, so I removed it.
* **EDA:**
  + I started by making boxplots for all the features so I could see the distribution of the data of each feature and the distribution of the outliers of each feature, I found that all the features contain outliers.
  + I created a bar plot that contains all the features, and this bar plot describes and shows the frequency of outliers of each feature. This step was made so I could judge which type of scaler I would use, the RobustScaler or the StanderdScaler because the number of outliers was not that much, I decided to use the StanderdScaler.

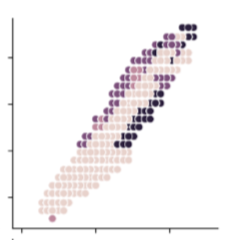
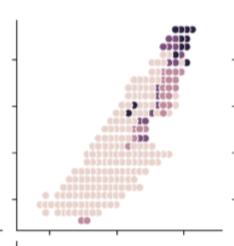
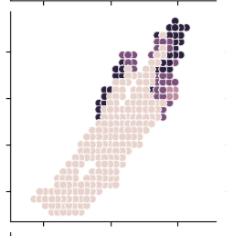


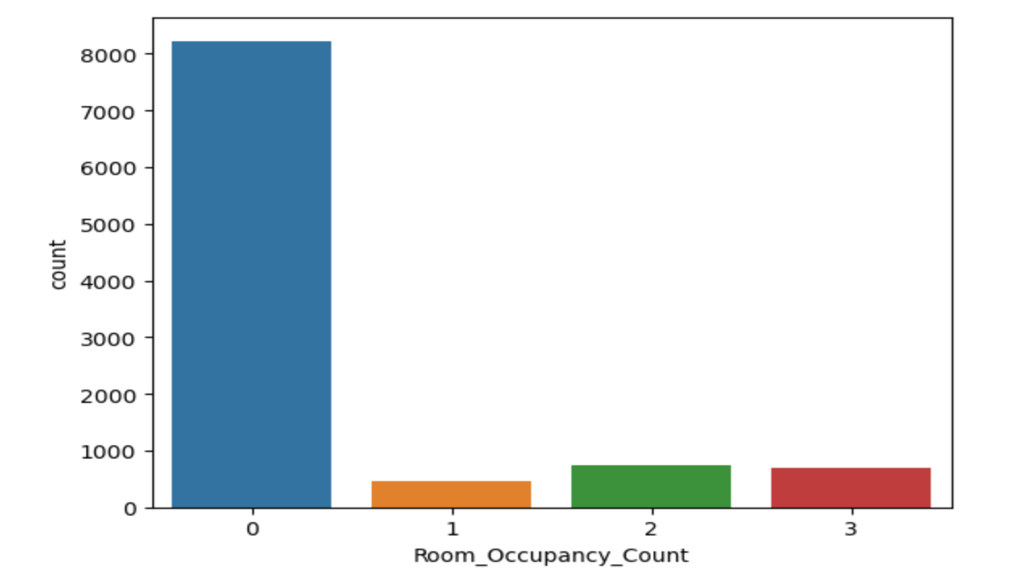
* + Then I made a Correlation Matrix so I could have a general overview of the relations between the columns. I found that most of the sensors have a relation with the target feature.



* + Then I created a pair plot for all the features with the target feature.



* Insights I came out with from the pair plot:
  1. I found that the temperature readings of sensor-1 and sensor-3 increase when the occupation count increases.
  2. I found that the temperature readings of sensor-1 and sensor-4 increase when the occupation count increases.
  3. I found that the temperature readings of sensor-3 and sensor-4 increase when the occupation count increases.
  + I created a bar plot that shows the frequency of each class in the target feature, and I found that the target feature is not balanced.



* + Then I created a lot of boxplots for all the features with the target feature, each boxplot shows each class value with its reading of each sensor, I found that when the Occupancy\_Count increases the temperature readings of all sensors increase and the sound readings also increase, and the light readings increase and the Co2 readings also increase when the Occupancy\_Count inceases.

### Methods*:*

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

1. **Random Forest Classifier:**
   1. **Ensemble Approach:** improves accuracy and reduces overfitting in classification by combining results from many decision trees, which is especially useful for complicated sensor data.
   2. **Noise and Overfitting Resilience:** Because it is ensemble-based, it is less vulnerable to noise and overfitting, which are commonly seen in sensor data.
   3. **Complex Data Handling:** This model has the ability to understand complex correlations between variables without requiring a lot of analysis of features.
   4. **Data Type Robustness:** Performs well when dealing with non-linear, outlier data, which is characteristic of sensor readings.
   5. **Feature Importance Analysis:** The feature importance that Random Forest offers may be quite helpful in determining which sensors are most likely to predict a room's occupancy.
   6. **Versatility and Flexibility:** This model can handle multiple types of relations such as linear and non-linear data.

* **Conclusion:**
* Because of its ensemble methodology, the Random Forest Classifier performs very well when predicting room occupancy using sensor data. It can handle complicated, noisy, and non-linear sensor data easily.

1. **Support Vector Classifier (SVC):**
   1. **Kernel Method:** It can handle non-linear interactions since it can employ kernel functions, which is useful for complicated sensor data that might not be linearly separable.
   2. **Customizable with Different Kernels:** SVMs offer flexibility to model complicated and unique data relationships since they may employ customized kernels in addition to the basic linear, polynomial, and RBF options.
   3. **Margin Maximization:** SVM has the ability to maximize the margin between classes, which can improve generalization on unseen data, which is a key aspect of occupancy prediction.
   4. **Robustness to Overfitting:** Compared to other algorithms, SVM may be more resistant to overfitting, especially in situations with a high number of dimensions as in our case, which will improve the performance.

* **Conclusion:**
* The SVC is special in occupancy estimation because of its kernel approach, which can handle non-linear sensor data with ease. Even with high-dimensional sensor data, its ability to modify kernels and avoid overfitting makes it an excellent choice for correctly classifying occupancy in rooms.

1. **Gradient Boosting:**
   1. **Sequential Learning (Iterative Refinement):** This model can be useful for complicated datasets such as sensor readings, as it generates trees one at a time, with each new tree helping to fix the miss classified by the previous trees which can lead to a high accurate model.
   2. **Handling of Non-Linear Data**: Gradient Boosting is strong at handling non-linear relationships, which are most commonly seen in this dataset.
   3. **Good Performance with Unbalanced Data:** It works effectively with datasets that have a significantly higher frequency of one class than the other (unbalanced data), which is often the case in the occupancy detection feature.
   4. **Flexibility in Depth of Trees:** Gradient Boosting allows you to balance the complexity of the model with the need for generalization by adjusting the depth of the trees.

* **Conclusion:**
* Gradient Boosting provides great accuracy in occupancy detection because of its iterative refinement and capacity to handle non-linear and imbalanced sensor data. It is an effective tool for accurate occupancy estimates because of its adjustable tree depth, which provides a customized modelling technique for complex sensor data.

1. **XGBoost:**
   1. **Built-in Regularization:** This model includes regularization from L1 (Lasso Regression) and L2 (Ridge Regression), which enhances model generalization and helps avoid overfitting.
   2. **Speed and Performance:** XGBoost is famous for its speed of execution and high model performance, both are essential when working with big datasets such as sensor data.
   3. **Cross-Validation Capability:** When creating models, this model uses cross-validation, helping to identify the most accurate prediction structure.

* **Conclusion:**
* It is a great choice for processing complicated, large-scale sensor information in occupancy estimation.  its included cross-validation and regularization features guarantee reliable and accurate occupancy estimates.
* **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):**
  + **Separating the data into Train and Test (split train–test):** I will split our dataset which is Occupancy\_Estimation into 80% for the training and 20% for the testing, the training set will be used you train the data on it and apply cross-validation on it, but the testing set I will leave it to the end so I can evaluate the overall performance of the model. So, this splitting was made to ensure that the model can generalize on unseen data.
  + **Applying GridSearchCV on the Training Set:**
    - **Identifying Parameters:** first, I identified the hyperparameters that strongly affect all the chosen models performances such as SVM hyperparameters (C, Kernel, Gamma). Random forest hyperparameters (n\_estimators, max\_features, max\_depth, criterion). GradientBoosting hyperparameters (n\_estimators, learning\_rate, max\_depth). XGBoost hyperparameters (n\_estimators, learning\_rate, max\_depth)
    - **Range Specification:** I give a set of values for each model hyperparameter to try:

**Random forest:**

'n\_estimators': [10, 50, 100, 200]

'max\_features': [None, 'sqrt', 'log2']

'max\_depth' : [4, 5, 6, 7, 8]

'criterion' :['gini', 'entropy']

**SVM:**

'C': [0.1, 1, 10, 100]

'kernel': ['linear', 'rbf', 'poly', 'sigmod']

'gamma': ['1', 'auto']

**GradientBoosting:**

'n\_estimators': [100, 200, 300]

'learning\_rate': [0.01, 0.1, 0.2

'max\_depth': [3, 4, 5]

**XGBoost:**

'n\_estimators': [100, 200, 300],

'learning\_rate': [0.01, 0.1, 0.2],

'max\_depth': [3, 4, 5],

* + - **Setting Up GridSearchCV:** Instantiate GridSearchCV to each model with the given parameters.
    - **Cross-Validation Configuration:** I specify the number of K-folds to generate which is 5 folds, which means the training set will be split into subsets so the model will train on k-1 and the remaining one will be for testing and this will be iterated until all the K folds will be as a train validation at least on time.
    - **Performance Evaluation Across Folds:** Each value of the hyperparameter of each model will be tested by the grid search which will train the model on each K-1 fold and the last fold will be the validation for testing so the model can evaluate the performance, this process will be repeated a lot of time tell all the folds they were a train set at least for one time and on all the hyperparameters.
    - **Compare the results:** the grid search will calculate the average accuracy of the folds iteration for each hyperparameter set, which means will have the performance of each hyperparameter set and these steps will be used for all the models that I’m required to do.
    - **Best Parameter Selection:** The set of hyperparameters that have the best K-fold average performance will be chosen.
  + **Final Model Training Using Optimal Parameters:** Retrain each model on their best hyperparameters using the training data
  + **Evaluating the Model on the Test Set:** I will use the testing set that was put on the side from the beginning to evaluate the model’s ability to generalize on unseen data, and this will be evaluated using precision, recall, f1-score, accuracy, and confusion matrix since it the a classification problem.
* **Explain in detail the machine learning algorithms you are using to address this problem.** 
  + **Random Forest:** 
    - **Overview:** This model is an ensemble learning method that works well for problems involving regression and classification. this model provides predictions that are more reliable and accurate because this model creates many decision trees during the training phase and mixes their results. This approach is very resistant and stable to overfitting, which is commonly seen in machine learning models. The reason why this model is robust against overfitting is mainly because this model creates many trees and mixes their output. This model not only can deal with overfitting but also can deal with and handle non-linear data effectively and easily all these strengths are because of the ensemble learning approach which allows the random forest model to deal with problems easily. Also, this model has the ability to make a decision and gives insights into which features are most important in making predictions. The Random Forest model is used in many real-life problems such as finance, fraud detection, healthcare, and disease prediction all because this model can deal with complex datasets. (IBM, 2017) (Wikipedia, 2023)
    - **How the Random Forest Classifier Model Works:** (IBM, 2017) (Wikipedia, 2023)
* **Starts with bagging:** Start creating the trees by using a random subset of the training set (Bootstrap Sample), this way of working allows the trees to be different from each other.
* **Feature Randomness:** while the trees are under construction, a random subset of features will be split to the location of nodes.
* **Each Single Tree Prediction:** Every single tree from the trees that were made will start making its own prediction by giving labels because our case is a classification problem.
* **Majority Voting:** After each tree finishes there prediction, the random forest starts doing aggregates to the results of each tree prediction. These aggregates will be done using something called majority voting, and then the Random Forest model's final prediction is given to the class that gets the majority of votes from all the trees.

**Strengths:** (IBM, 2017) (Wikipedia, 2023)

* Handles both categorical and numerical data well.
* Robust to overfitting, especially in cases where the dataset is large.
* Can be used to determine feature importance.

**Weakness:** (IBM, 2017) (Wikipedia, 2023)

* Performance can be impacted if there are too many trees in the forest.

**Random Forest Hyperparameters:**

* **n\_estimators:** Number of trees in the forest.
* **max\_features:** The maximum number of features that are considered for splitting a node.
* **max\_depth:** The maximum depth of each tree
* **criterion:** The function used to evaluate the quality of a split
  + - **Random Forest Pseudo Code:**

RandomForestClassifier(n\_estimators, max\_features, max\_depth, criterion):

Create an empty list for the forest

For each tree in n\_estimators:

Randomly select samples from the training data (Bootstrap)

If max\_features is given, randomly select features

Create a decision tree with the selected data and features

If max\_depth is given, limit the tree depth

Use the criterion for splitting (like 'gini' or 'entropy')

Add the tree to the forest

Return the forest of trees

Function predict(X, forest):

Initialize an empty list for predictions

For each tree in the forest:

Make a prediction for X using the tree

Add the prediction to the list of predictions

Combine predictions

Return the most common prediction from all trees (majority vote)

* + **Support Vector Machine (SVM):**
* **Overview:** Support Vector Machines (SVM) is a strong supervised learning algorithms that work well in high-dimensional places, the way SVM operates is by identifying the best possible hyperplane for classifying the data. The thing that gives the SVM strength is that it uses kernel functions (“Kernel Trick”) which give the SVM the ability to handle and deal with non-linear relations efficiently just by mapping input features into higher-dimensional spaces. Also, SVM usually focuses on maximizing the margin that it is between a lot of classes that are different from each other, by maximizing the margin between classes, will increase the performance because the model will better generalization on unseen data. Also, one of the SVM model features is that SVM has the ability to regularization features, which will prevent the model from overfitting. The SVM regularization is managed by the 'C' parameter, which offers a trade-off between having a low testing error and a low training error. Also, the SVM model works greatly in situations where the number of features is higher than the number of samples. (Wikipedia, 2024) (scikit, 2016)
* **How the SVM Classifier Model Works:** (Wikipedia, 2024) (scikit, 2016)
  + **The objective of the model:** Is to find which is the best hyperplane that divides data points belonging to distinct classes in the feature space
  + **Hyperplane and Margin:** The method first finds a hyperplane, or decision boundary that maximizes the margin, This measures the distance between the closest data points from each class and the hyperplane.
  + **Support Vectors:** These are the crucial information points that affect the hyperplane's placement that are closest to it.
  + **Linear SVM:** SVM locates a hyperplane that linearly divides the classes with the largest margin in a linear situation.
  + **Non-linear SVM (Kernel Trick):** SVM uses the kernel method to project data into a higher-dimensional space where a linear separation can be achieved in cases when the data are not linearly separable.
  + **Optimization Problem:** To identify the hyperplane with the largest margin, SVM formulates and resolves a quadratic optimization problem.
  + **Prediction:** After training, SVM classifies new data points by locating them in relation to the hyperplane.
* **Strength:** (Wikipedia, 2024) (scikit, 2016)
  + Perform very well in high-dimensional spaces.
  + Perform very well in cases where the dimension is larger than the samples.
  + **Weakness:** (Wikipedia, 2024) (scikit, 2016)
  + It is not suitable for large datasets because the training time will be so high.
  + It is sensitive, so it requires careful parameter tuning and selection of the suitable kernel.
* **SVM classifier Hyperparameters:** (scikit, 2016)
  + **'C':** Manages the trade-off between accurately classifying training points and a smooth decision boundary (Regularization Parameter).
  + **'kernel':** Decides what kind of transformation is used on the input data, which has an impact on whether the algorithm can handle linear or non-linear connections.
  + **'gamma':** Indicates the range of effect that a single training example can have; low values indicate "far," while high values indicate "close."
* **SVM Pseudo Code:**

Function SVM\_SVC(C, gamma, kernel):

Choose the kernel function (e.g., 'linear', 'RBF', 'polynomial') based on the kernel parameter

If the kernel is 'RBF', use gamma to define the influence of a single training example

Initialize the SVM model

Use the C parameter to set the regularization strength

Transform the input data into the chosen kernel space (if necessary)

Train the model to find the optimal hyperplane that separates the classes:

- Optimize the model to maximize the margin between classes

- Minimize the classification errors (misclassifications)

- Identify support vectors as the critical data points nearest to the hyperplane

Return the trained SVM model

Function predict(X, model):

For each sample in X:

- Transform the sample into the kernel space (if necessary)

- Use the trained model to predict the class of the sample

Return the predicted classes for X

* + **Gradient Boosting:** (Saini, 2024) (scikit, 2017)
    - **Overview**: It is an ensemble method that creates trees in sequential order, with each tree trying to correct the mistakes made by the ones before it, focusing on improving the areas where the models before it did not perform well. The fundamental idea of gradient boosting is to build a powerful overall model by combining several weak learners, which we mean decision trees. Every tree in the series corrects the errors of the one before it, which improves the model's performance. This model also includes controls for overfitting, which is an essential component of creating an accurate prediction model.
    - **How the Gradient Boosting model works:** (Saini, 2024) (scikit, 2017)
      * **Initialization:** Begins with a base model that produces initial predictions for each observation in the dataset, usually a simple decision tree.
      * **Sequential Tree Building:** A new tree is added in each step that follows. Every new tree makes up for solving an error of the ensemble of the already-made trees.
      * **Error Correction:** The algorithm determines where the predictions made by the current ensemble are wrong after every round. After this, each new tree focuses on specific areas that they misclassified, and these new trees learn from the existing ensemble trees’ mistakes.
      * **Gradient Descent:** The method uses gradient descent to minimize the loss, or the difference between the expected and actual values by modifying the hyperparameters.
      * **Addition of trees:** The new tree predictions will be added to the previous tree predictions. Usually, the result is a weighted total in which accurate trees have a greater impact.
      * **Stopping Criteria:** The process will continue until it reaches the predefined number of trees in the hyperparameter or there's nothing much that can be improved.
    - **Strengths:** (Saini, 2024) (scikit, 2017)
      * Has the ability to offer excellent prediction accuracy.
      * offering several hyperparameter tuning choices and enabling optimization for different loss functions using Gradient Descent.
    - **Weakness:** (Saini, 2024) (scikit, 2017)
      * Easily Overfitted if data is noisy.
      * It takes a long time to train and is very sensitive to hyperparameter tuning.
      * Hard to interpretable, which means it is somehow complex.
    - **Gradient Boosting Hyperparameters:** (Saini, 2024) (scikit, 2017)
      * **'n\_estimators':** This parameter is the number of sequential trees to be created, something to consider, is that more trees can lead to higher performance, but the model will be at risk of overfitting.
      * **'learning\_rate':** This parameter is to assesses each tree's effect on the result (lower rates need more trees but can result in greater generalization).
      * **'max\_depth':** This parameter is to tell the maximum depth needed for each tree, also this parameter controls the complexity of the mode.

**Gradient Boosting Pseudo Code:**

GradientBoostingClassifier(n\_estimators, learning\_rate, max\_depth):

Initialize boosting parameters

For each tree in range(n\_estimators):

Compute the negative gradient of the loss function

Fit a decision tree to the negative gradient of the loss function

If max\_depth is specified:

Limit the depth of the decision tree to max\_depth

Update the model by adding the scaled output of the current tree to the previous model's output

Scale the tree's output by the learning\_rate

Return the final results (ensemble of trees)

Function predict(X, model):

Initialize a variable to hold the sum of predictions

For each tree in the model:

Add the tree's prediction for X to the sum

Return the sum of predictions as the final prediction

* + **XGBoost:**
    - **Overview:** XGBoost stands for eXtreme Gradient Boosting, it is a more advanced and efficient version of the simple Gradient Boosting method. It is made to be very effective, scalable, and portable. Also, this model has regularization parameters L1 and L2, that help in lowering overfitting. This model has a built-in procedure that enables it to handle missing values which will result in low problems and errors that might happen because of null values. This model also has a built-in cross-validation method for each iteration which also improves the model performance and gives more accurate results. (Simplilearn, 2023) (GeeksforGeeks, 2023)
    - **How XGBoost Classifier Model Works:** (Simplilearn, 2023) (GeeksforGeeks, 2023)
      * **Creates Base Learner:** Begins with a basic model (most often it is a decision tree) that creates initial predictions for every instance in the dataset.
      * **Calculate Loss Function and Gradient descent:** Applying a different loss function to calculate the loss which is the difference between the expected and actual values). Then calculates the gradients of the loss in relation to the predictions, indicating the best course of action (where to move) for minimizing mistakes.
      * **Adding new trees to correct the errors for the previous trees:** creates a new decision tree using the gradient information, with each leaf trying to lower the loss. The mistakes or residuals from the earlier trees are the main goals of this new tree.
      * **Update the model using iterations:** This includes calculating the output values for each leaf in the tree, which then get added to the previous model's predictions.
      * **Apply Regularization:** Regularization is used both in the building of the trees and in the calculation of the leaf outputs.
      * **Repeating steps 2 (Calculate Loss Function and Gradient descent) and 5 (Apply Regularization):** This step stops when the model archives the minimum loss value.
      * **Cross-Validation and Hyperparameter Tuning:** Applying the built-in methods such as cross-validation so we can have the best performance, and then tune some parameters such as (n\_estimators, learning\_rate, and max\_depth)
      * **Optimization:** Simplifies the model by removing low-value tree parts, and speeds computations using data-handling built-in techniques.
    - **Strengths:** (Simplilearn, 2023) (GeeksforGeeks, 2023)
      * High speed in execution, and it’s excellent performance.
      * This model has the ability to handle missing data by itself.
      * Have built-in regularization and cross-validation techniques.
    - **Weakness:** (Simplilearn, 2023) (GeeksforGeeks, 2023)
      * It might overfit if the parameters were not tuned correctly.
      * Have a higher complexity than any simple model and also it is hard to tune this model
    - **XGBoost Hyperparameters:** (Simplilearn, 2023) (GeeksforGeeks, 2023)
      * **n\_estimators':** This parameter is the number of sequential trees to be created, something to consider, is that more trees can lead to higher performance but the model will be at risk to overfit.
      * **learning\_rate':** This parameter is to assesses each tree's effect on the result (lower rates need more trees but can result in greater generalization).
      * **max\_depth':** This parameter is to tell the maximum depth needed for each tree, also this parameter controls the complexity of the mode.
    - **XGBoost Pseudo Code:**

XGBoostClassifier(n\_estimators, learning\_rate, max\_depth):

Initialize model

For each tree to n\_estimators:

Compute the gradient of the loss function

Fit a new tree to predict these gradients

If max\_depth is specified:

Limit the depth of the tree to max\_depth

Update the model by adding the weighted output of the new tree to the previous model

Scale the update by the learning\_rate

Return the final result (ensemble of trees)

Function predict(X, model):

Initialize a variable for the sum of predictions

For each tree in the model:

- Add the tree's prediction for X to the sum

Return the sum as the final prediction

### Evaluation*:*

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

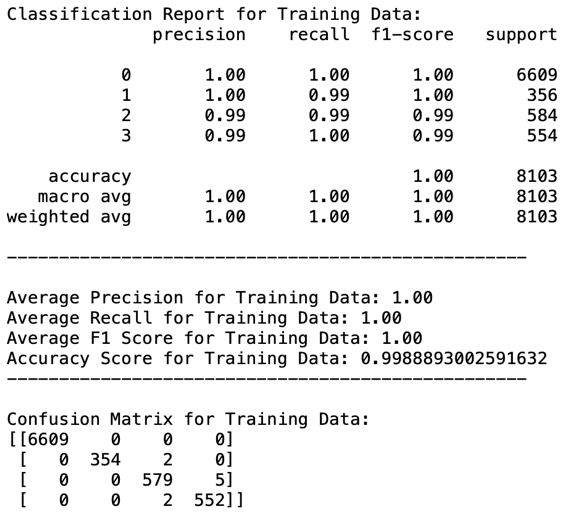
* **What performance measures did you use to evaluate the effectiveness of your models?**
* **Accuracy Score:** It is a simple method for evaluating a model's performance. It is computed by dividing the number of true predictions by the model's total number of predictions, in a simple way, it shows the percentage of the model's predictions that come true.

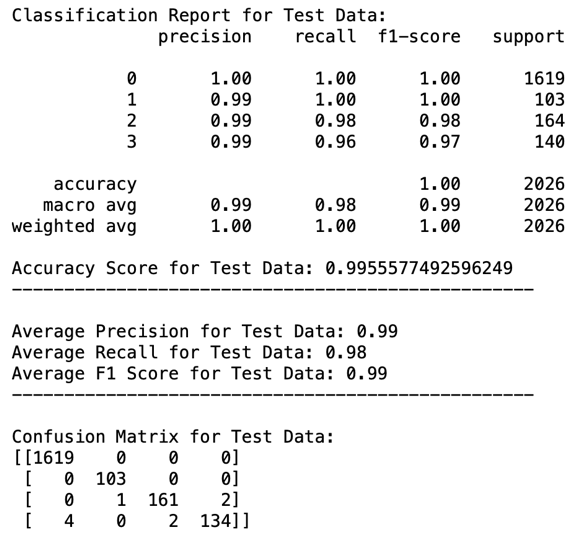
I calculated the accuracy score for all models that I have used and also the accuracy was calculated for the training data and the testing data so I could see if the model performed well on both seen and unseen data so I can identify issues such as overfitting, underfitting, or if it balanced, which means that this measure can give me a quick overview of how the model is dealing this with the training and testing data. But watch out because this measure is not reliable with imbalanced datasets so the data need to be prepared strongly.

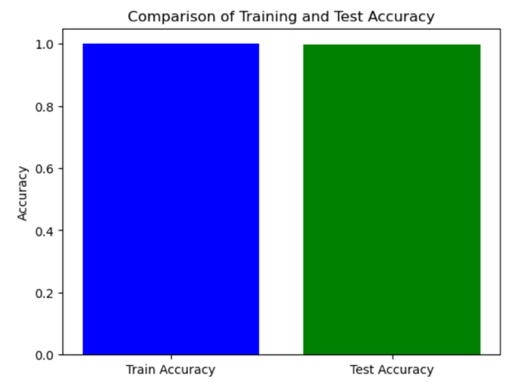
* **Classification Report:** The classification report gives a full summary of the model performance, and this summary is shown by Precision, Recall, and F1-Score. I have used this Classification Report for all models that I have used and also I used the Classification Report for the training data and the testing data so I could see if the model performed well and classified the classes correctly on both seen and unseen data
  + **Precision:** This measure shows the ratio of correctly predicted positive observations to the total predicted positives. The high value of precision means a low false positive rate.
  + **Recall:** This measure shows the ratio of correctly predicted positive observations to all observations in actual class. This shows how many actual positives were correctly predicted.
  + **F1-Score:** This measure takes the weighted average (Harmonic Mean) of both recall and precision to get a single value. This score takes both false positives and false negatives. It is used to show a balanced assessment of precision and recall how much a class has good precision and recall. This measure is very important in cases where a class is imbalanced as the one we have in our dataset
* **Confusion Matrix:** The confusion matrix is a table that shows the performance of any classification model, this is considered as one of the ways to visualize the classification model performance. Each column in this matrix shows the predicted instance of the class, and each row in the matrix shows the actual instance class. I have used it in my project for both the training set and testing set so I can see how many instances each model is miss-classified in the training phase and in the testing phase.
* **Why did you use these metrics?** 
  + **Accuracy Score:** This measure is usually used as the first measure to evaluate any model performance and as I mentioned before this measure is computed by dividing the number of true predictions by the model's total number of predictions which is necessary as a first step to know the percentage of correct predictions over the total number of cases that were predicted. However, I know that the target class in my dataset is imbalanced, and the accuracy measure value could be very high even if the target is imbalanced which means that the accuracy measure could be not accurate because the model always predicts the majority class which is (0) so that’s mean that the model will still achieve high accuracy. Even though this measure has a limitation that I have mentioned, but it is useful to have a quick view of the overall model performance.
  + **Conclusion why I used this measure:**
    - To get a general overview of how each model performs and provide a quick assessment of the performance.
  + **Classification Report:**
* **Precision:** As I have mentioned before this measure shows the ratio of correctly predicted positive observations to the total predicted positives that were actually correct, so one of the reasons that I used it, is to measure the accuracy of positive predictions. Also, in my case which the data is not balanced it is important to calculate the precision because it helps in determining the reliability of the model, but it is not very reliable in the case of imbalanced data because it does not contain false negatives. Also, I used this measure because it enables us to avoid overestimation or underestimation in occupancy predictions. But it has a not good thing which is that this measure ignores the cases (false negatives) that the model failed to detect. But to get useful information I have to use precision with recall which means balancing it with recall is necessary.
* **Conclusion why I used this measure:**
  + To evaluate the accuracy of positive predictions.
  + To avoid overestimation or underestimation in occupancy predictions.
* **Recall:** This measure was used to measure and evaluate how well the model is identifying less frequent classes that are in the occupancy column in our data set and evaluates the model's capacity to recognize every actual positive. The high value of recall shows that the model effectively identifies and captures all classes that are in the occupancy column. However, since the column is imbalanced the recall measure cannot be so accurate to judge because false positives are missing in the recall measure. But to get useful information I have to use precision with recall which means balancing it with precision is necessary. As a result it helps me to assess how well the model is identifying the less frequent classes (0,1, 2, 3)
* **Conclusion why I used this measure:**
  + To assess how well the model identifies less frequent classes.
  + To evaluate the model's ability to recognize every actual positive instance.
* **F1-Score:** I used this measure because this measure provides a balance between precision and recall. This measure was so important to apply since the data that I’m dealing with is imbalanced and both minor and major classes are important, so it will give an accurate measurement than using precision and recall individually. This measure is the only way to make sure that both false positives and false negatives are minimized.
* **Conclusion why I used this measure:**
  + To balance precision and recall
  + To provide a more helpful measure than precision or recall
  + To minimize both false positives and false negatives.
* **Note:** The precision, recall, and F1-Score all use the Macro Average because these measures were generated by the Classification\_Report which is a function from scikit-learn which will automatically calculate and include the macro average in its output without needing to specify the average parameter.
* **Macro Average:** It is an approach that determines performance measures (such as precision, recall, or F1-score) independently for each class regardless matter how frequent a class is in the dataset it will give it the same weight for each class, which is so helpful in my dataset which is imbalanced data, and then computes the arithmetic mean of those metrics. This method is used in classification tasks to find the average of these metrics which is suitable for my data set because it has multiple values to classify which means it is not binary classification. So I used the macro average to make sure that I would have a balanced evaluation for the model performance across multi-classes.
  + **Confusion Matrix:** I used the confusion matrix because it enables a detailed visualization showing the prediction in relation to their actual values and also it shows how many predictions were correctly predicted in each class and how much miss classification of each class was done by the model and this is so helpful in my situation which is dealing with imbalanced data because it shows how much good the model is identifying each class which helped me to make sure that the model is not biased to the majority class which is (0).
* **Note:** I used these measures in the training phase and in the testing phase so I could judge if there was overfitting or underfitting or if it balanced.
* **Evaluate how, based on the performance measures, you were able to enhance the model.**
* I have used a lot of approaches and techniques so I can enhance the model performance. I will start talking in detail about each technique that I have used and evaluate how I accept or reject this technique based on the performance measures:
  + **Over-Sampling:** Because I have found that my data set is imbalanced, I decided to balance class distribution by increasing the number of instances in the minority classes, so I applied an Over-Sampling Technique the first one was SMOTE (Synthetic Minority Over-sampling Technique) and the second one was the KMeans SMOTE Technique. After applying the first one which is SMOTE, I found that the performance measure values stayed as they were before applying the over-sampling technique, then I tried another over-sampling technique which is KMeans SMOTE, and also the performance measure values stayed as they were before applying the over-sampling technique, but when using both techniques the running time increased so much, so after this I consider that my imbalanced dataset does not have an impact of the model ability to classify correctly. So, I removed the Over Sampling techniques since the running time increased so much without having any improvements on the values of the performance measures of each model.
  + **Under Sampling:** Because the Over-Sampling techniques do not improve the models performance, I decided to try the Under-Sample Technique which is Cluster Centroids to reduce the size of your dataset by decreasing the number of instances in the majority class. The results were a big decrease in the running time which is good but also it led to a reduction in the performance of the models and this was shown from the performance measures that I’m applying to each model, which means that the removed data was necessary for the model to learn and perform well, and this also means that the data is not enough to capture the patterns. So I removed the Under-Sampling technique that was applied because it does not improve the model performance
  + **Feature Selection:** I applied a Feature Selection Technique, which works by removing irrelevant features from the dataset, but unfortunately it did not improve the performance measures of the models that are used. This means that there are no irrelevant features and the dataset and most of the features are relevant and help the models to catch the pattern which is in the dataset.
  + **Standard Scaling:** I applied Standard Scaling to normalize all the values of the dataset because one of my models which is SVM is so sensitive to the scale of data, and this step improves the performance measures of the models because the models now are dealing with the same scale of data.
  + **Macro Average:** I applied the Classification\_Report which is a function from scikit-learn which will automatically calculate and include the macro average in its output without needing to specify the average parameter. This feature helps ensure that the model's performance is not biased towards the majority class, which means it offers more balanced performance measure values of all the models that are applied even if it does not improve the performance measures directly, but it is so useful which means more efficient models prediction.
  + Hyperparameters: I used GridSearch to I can use many hyperparameters so for each model and then the models use the parameters that give the highest performance measures.

### Results and Discussion*:*

* Discuss the reliability of your results and whether they are balanced, overfitting, or underfitting.
* **Results of Random Forest:**
  + Best Parameters found by Grid Search: {'criterion': entropy, 'max\_depth': 8, 'max\_features': 'log2', 'n\_estimators': 10}

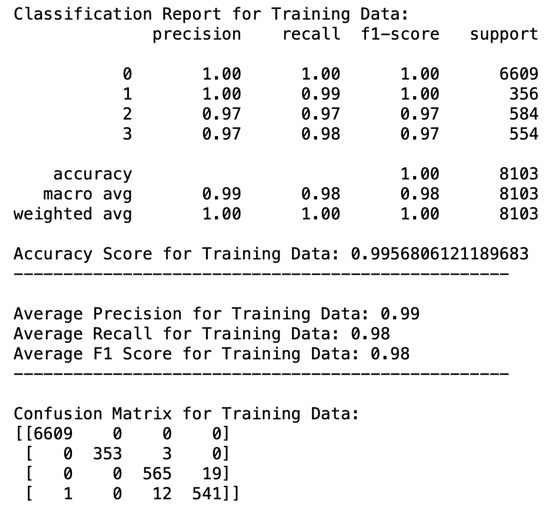
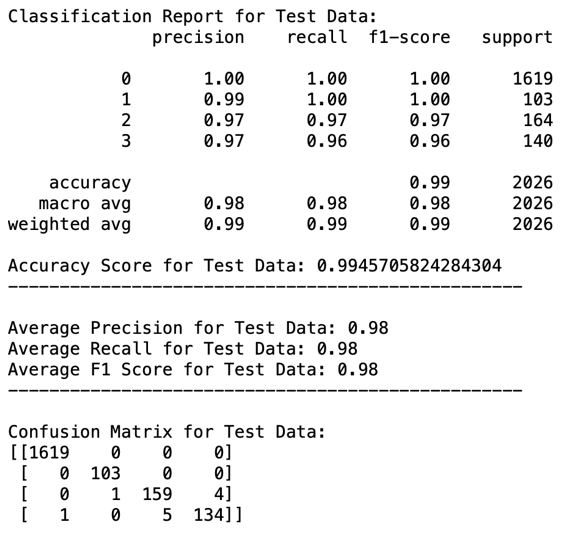


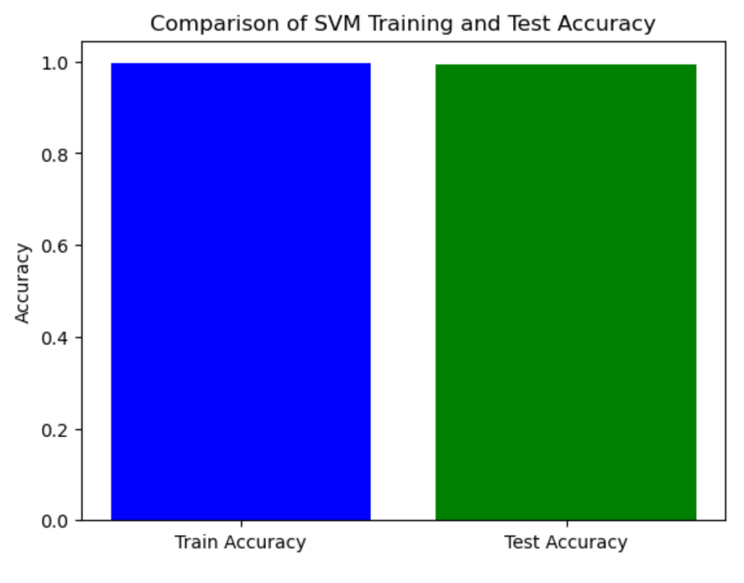
* + 



* + **Discussing the results of the Random Forest model:**
  + **For the Training Data:** 
    - **Precision, Recall, and F1-Score:** I have almost a perfect score, which was the average of each value of these measures was 1.00 in the training data. This indicates that the model is very good at identifying the relevant instances for each class.
    - **Accuracy:** The accuracy value on the training data was so high (0.9989), which shows that my model is correctly predicting the correct class.
    - **Confusion Matrix:** The confusion matrix shows that the true positive is very high and just a very few misclassifications, which also this measure shows that the model is performing very well using the training data.
  + **For the Testing Data:**
    - **Precision, Recall, and F1-Score:** The values of these measures were around 0.99 which means it is a bit lower than the values of these measures in the training set, but it is still so close and high, and this means that the model is performing well and have the ability to generalize.
    - **Accuracy:** The accuracy value in the testing data set was 0.9956 which is a bit lower than the value of the accuracy on the training data set, but it is still a high value, and it is so close to the value of the accuracy on the training data set, which this means that the model is performing very well on unseen data sets.
    - **Confusion Matrix:** The confusion matrix shows that the true positive is very high and just a very few misclassifications, but the misclassification frequency is a bit higher than its frequency in training data but that was expected. Also, this measure shows that the model is performing well, and it can generalize.
  + **Is my model being Overfitting, Underfitting, or Balanced:**

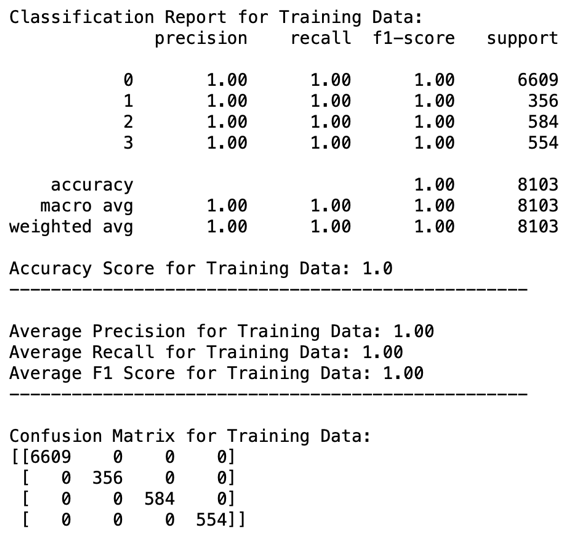
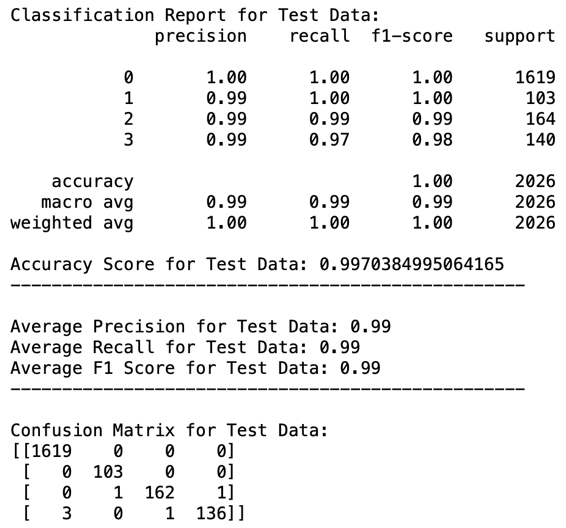
As you can see, the Random Forest model has performed excellently on both the training data set and the testing data set even if there was a very small difference between the accuracy of the model between the training and testing, but the difference is very small, so it is not enough to say it is overfitting. Also, as I mentioned before the accuracy of both the training and testing was too high, which means that the model is not underfitting. The Random Forest model was able to generalize and performed very well and was very suitable with my data set, so we can say it is a balanced and reliable model.

* **Results of the SVM Model:**
  + Best Parameters found by Grid Search: {'C': 1, 'gamma': 'auto', 'kernel': 'linear'}
  + 
  + 



* + **Discussing the results of the SVM Model:**
  + **For the Training Data:** 
    - **Precision, Recall, and F1-Score:** I have almost a perfect score, which was the average of each value of precision and recall was close to 1.00 in the training data. This indicates that the model is very good at identifying the relevant instances for each class.
    - **Accuracy:** The accuracy value on the training data was so high (0.9957), which shows that my model is correctly predicting the correct class.
    - **Confusion Matrix:** The confusion matrix shows that the true positive is very high and just a very few misclassifications, which also this measure shows that the model is performing very well using the training data.
  + **For the Testing Data:**
    - **Precision, Recall, and F1-Score:** The average values of these measures were around 0.98 which means it is a bit lower than the values of these measures in the training set, but it is still so close and high, and this means that the model is performing well and have the ability to generalize and reliable.
    - **Accuracy:** The accuracy value in the testing data set was 0.9946 which is a bit lower than the value of the accuracy on the training data set, but it is still a high value, and it is so close to the value of the accuracy on the training data set, which this means that the model is performing very well on unseen data sets, so this model is reliable.
    - **Confusion Matrix:** The confusion matrix shows that the true positive is very high and just a very few misclassifications, but the misclassification frequency is a bit higher than its frequency in training data but that was expected. Also, this measure shows that the model is performing well, and it can generalize.
  + **Is my model being Overfitting, Underfitting, or Balanced:**

As you can see, the SVM model has performed excellently on both the training data set and the testing data set even if there was a very small difference between the accuracy of the model between the training and testing, but the difference is very small, so it is not enough to say it is overfitting. Also, as I mentioned before the accuracy of both the training and testing was too high, which means that the model is not underfitting. The SVM model was able to generalize and performed very well and was very suitable with my data set, so we can say it is a balanced and reliable model.

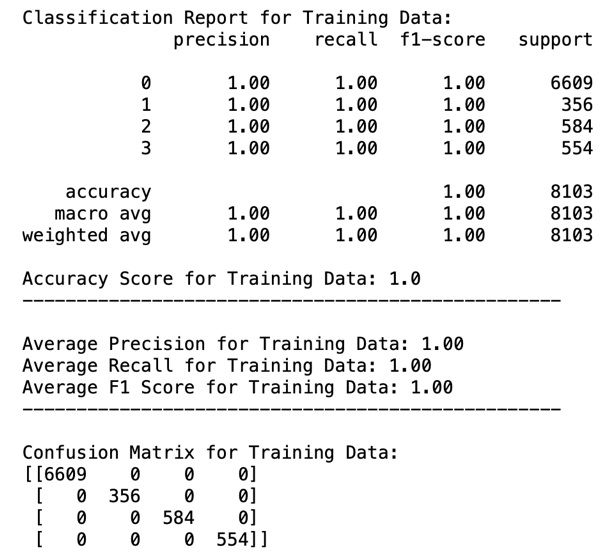
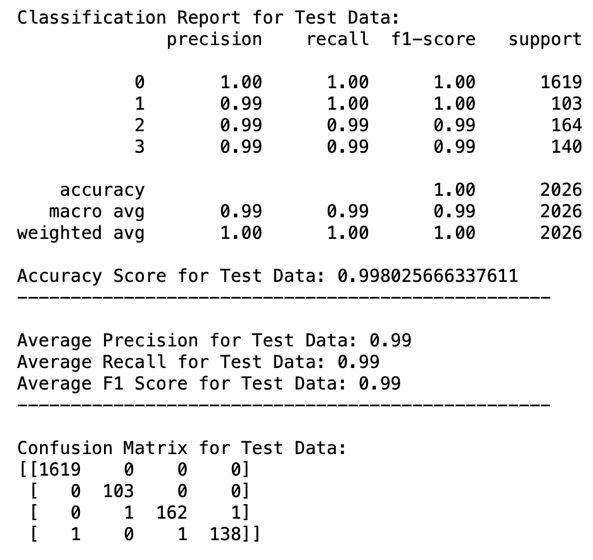
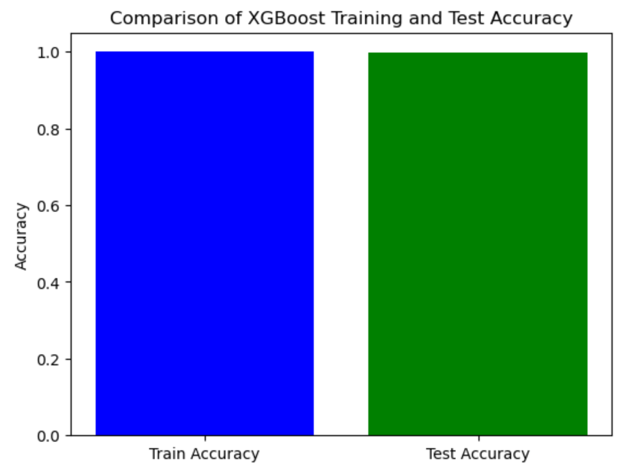
* **Results of the Gradient Boosting Model:**
  + Best Parameters found by Grid Search: {'learning\_rate': 0.1, 'max\_depth': 4, 'n\_estimators': 100}
  + 
  + 
  + A blue and green rectangular bars

    Description automatically generated
  + **Discussing the results of the Gradient Boosting Model:**
  + **For the Training Data:** 
    - **Precision, Recall, and F1-Score:** I have almost a perfect score, which was the average of each value of precision and recall was 1.00 in the training data. This indicates that the model is very good at identifying the relevant instances for each class.
    - **Accuracy:** The accuracy value on the training data was so high (1.0), which shows that my model is correctly predicting the correct class without any error.
    - **Confusion Matrix:** The confusion matrix shows that all the predictions of all classes were true positive without any misclassifications, which also this measure shows that the model is performing very well using the training data and is reliable.
  + **For the Testing Data:**
    - **Precision, Recall, and F1-Score:** The average values of these measures were around 0.99 which means it is a bit lower than the values of these measures in the training set, but it is still so close and high, also this means that the model is performing well on the training data and did not misclassify any class but in the testing data the model did misclassify, which means the model in overfitting.
    - **Accuracy:** The accuracy value in the testing data set was 0.997 which is a bit lower than the value of the accuracy on the training data set, but it is still a high value, and it is so close to the value of the accuracy on the training data set, which this means that the model is overfitting, because the accuracy is high but the model is not doing well with new data.
    - **Confusion Matrix:** The confusion matrix shows that the true positive is very high and just a very few misclassifications but comparing it with the training matrix it shows that there is overfitting.
  + **Is my model being Overfitting, Underfitting, or Balanced:**

As you can see, the Gradient Boosting model has performed excellently on both the training data set and the testing data set even if there was a very small difference between the accuracy of the model between the training and testing, the difference is very small, but because this model on the training set did not misclassify any class but the model did misclassify on the testing set that means that the model is overfitting which means that the model knows all the patterns in the training set but did not generalize on unseen data.

* **Results of the XGBoost Model:**

Best Parameters found by Grid Search: {'learning\_rate': 0.1, 'max\_depth': 3, 'n\_estimators': 300}

* 
* 
* 
  + **Discussing the results of the XGBoost Model:**
  + **For the Training Data:** 
    - **Precision, Recall, and F1-Score:** I have almost a perfect score, which was the average of each value of precision and recall was 1.00 in the training data. This indicates that the model is very good at identifying the relevant instances for each class.
    - **Accuracy:** The accuracy value on the training data was so high (1.0), which shows that my model is correctly predicting the correct class without any error.
    - **Confusion Matrix:** The confusion matrix shows that all the predictions of all classes were true positive without any misclassifications, which also this measure shows that the model is performing very well using the training data and is reliable.
  + **For the Testing Data:**
    - **Precision, Recall, and F1-Score:** The average values of these measures were around 0.99 which means it is a bit lower than the values of these measures in the training set, but it is still so close and high, also this means that the model is performing well on the training data and did not misclassify any class but in the testing data the model did misclassify, which means the model in overfitting.
    - **Accuracy:** The accuracy value in the testing data set was 0.997 which is a bit lower than the value of the accuracy on the training data set, but it is still a high value, and it is so close to the value of the accuracy on the training data set, which this means that the model is overfitting, because the accuracy is high but the model is not doing well with new data.
    - **Confusion Matrix:** The confusion matrix shows that the true positive is very high and just a very few misclassifications but comparing it with the training matrix it shows that there is overfitting.
  + **Is my model being Overfitting, Underfitting, or Balanced:**

As you can see, the XGBoost model has performed excellently on both the training data set and the testing data set even if there was a very small difference between the accuracy of the model between the training and testing, the difference is very small, but because this model on the training set did not misclassify any class but the model did misclassify on the testing set that means that the model is overfitting which means that the model knows all the patterns in the training set but did not generalize on unseen data.

* **Conclusion:** starting with the first two models which are the Random Forest and SVM models, these models performed very well on both the training data set and the testing data set, this was proved by the values of the performance measures that were used for both the training set and the testing set and all the measures values were so high while using train data and also the values of the measures were high while using the test data but they were a bit lower, this shows that these models are able to generalize without overfitting or underfitting, this means that these models are well-balanced and reliable for our data. Moving on to the two other models that I have used which were Gradient Boosting and XGBoost, these models give perfect measure values (1.0) while dealing with the train data, but when dealing with the testing data the measure values were a bit lower which this means that the model have learned every single pattern in the training set and the model was not able to generalize it to the testing set so these two models where overfitting and not reliable.

Bottom of Form

* **Analyse the result of the applications to determine the effectiveness of the algorithms.**
* **Random Forest Model:**
  + **Training Data:** The results were perfect, with the precision, recall, and F1-scores (averaging 1.00), and an accuracy of 0.9989. this shows that the model can correctly identify and classify instances across different classes.
  + **Testing Data:** The results were also perfect even when the performance values decreased a little bit, with the precision, recall, and F1-scores values averaging around 0.99 also the accuracy was so high (0.9956), this shows the ability of the model to generalize on unseen data.
  + **Generalization Ability:** The small difference in the performance between when applying the model on the training set and when applying the model on the testing set shows that the model can generalize and the performance measure of the training set and the testing set show that there is no overfitting which means that the model has learned the patterns without memorizing it.
  + **Conclusion:** The Random Forest model has high effectiveness, it has a strong predictive ability and that is proved through the accuracy, F1-Score, and reliability in both the training and testing sets. This model is also effective because it can manage between a high accuracy and a perfect generalization which makes it suitable for a lot of applications.
* **SVM Model:**
  + **Training Data:** The results were almost perfect, with the precision average value (0.99), recall average value (0.98), and F1-scores average value (0.98 ) and an accuracy of 0.9957. This shows that the model can correctly identify and classify instances across different classes in the training phase.
  + **Testing Data:** The results were also perfect even when the performance values decreased a little bit, with the precision average value (0.98), recall average value (0.98), and F1-scores average value (0.98) also the accuracy was so high (0.9946), this shows the ability of the model to generalize on unseen data.
  + **Generalization Ability:** The small difference in the performance between when applying the model on the training set and when applying the model on the testing set shows that the model can generalize and the performance measure of the training set and the testing set show that there is no overfitting which means that the model has learned the patterns without memorizing it
  + **Conclusion:** The SVM model is effective and reliable, it has a strong predictive ability and that is proved through the accuracy and reliability in both the training and testing sets which means that this model has strong predictive capabilities on both seen and unseen data. This model is also effective because it can manage between a high accuracy and a perfect generalization which makes it suitable for a lot of applications where generalizing is important.
* **Gradient Boosting Model:**
  + **Training Data:** The results were almost perfect, with the precision average value (1.00), recall average value (1.00), and F1-scores average value (1.00) and an accuracy of 1.00. This shows that the model can correctly identify and classify instances across different classes in the training phase.
  + **Testing Data:** The results were also perfect even when the performance values decreased a little bit, with the precision average value (0.99), recall average value (0.99), and F1-scores average value (0.99) also the accuracy was so high (0.9970
  + **Generalization Ability:** The small difference in the performance between when applying the model on the training set and when applying the model on the testing set shows that the model has learned too much from the training data which results in the model memorizing all patterns of the training data but this model cant generalize on the testing data. Which causes an overfitting.
  + **Conclusion:** while this model performs effectively on the training and testing set, the Gradient Boosting model is overfitted because the model learns too much on the training set and does not generalize it on the testing set. But this model is still effective but not for our data set.
* **XGBoost Model:**
  + **Training Data:**

The results were almost perfect, with the precision average value (1.00), recall average value (1.00), and F1-scores average value (1.00) and an accuracy of 1.00. This shows that the model can correctly identify and classify instances across different classes in the training phase.

* + **Testing Data:** The results were also perfect even when the performance values decreased a little bit, with the precision average value (0.99), recall average value (0.99), and F1-scores average value (0.99) also the accuracy was so high (0.998)
  + **Generalization Ability:** The small difference in the performance between when applying the model on the training set and when applying the model on the testing set shows that the model has learned too much from the training data which results in the model memorizing all patterns of the training data but this model cant generalize on the testing data. Which causes an overfitting.
  + **Conclusion:** while this model performs effectively on the training and testing set, the Gradient Boosting model is overfitted because the model learns too much on the training set and does not generalize it on the testing set. But this model is still effective but not for our data set.
* **Overall Evaluation:**
  + All the models have high effectiveness with high performance on training data, but the Random Forest and SVM models have a stronger ability to generalize more than the Gradient Boosting and XGBoost models.
  + While the Gradient Boosting and XGBoost models are effective on training data, but they overfit on the testing data, which will affect the performance of many real-world scenarios.
  + For scenarios where generalization is important, Random Forest and SVM will be the best choice. For scenarios where the future data would be represented by the training data, Gradient Boosting and XGBoost will be the best choice, but these models need to be tracked to avoid overfitting.
* **Draw conclusions regarding the strengths and weaknesses of the different algorithms:**
* **Random Forest:**
  + **Strengths:** (IBM, 2017) (Wikipedia, 2023)
    - **Have a very high ability to Generalize:** which means have the ability to generalize the pattern that the model learns from the training data and apply it to the testing data. In another way, this model can make the same performance on the training and testing data or with a very small drop in the performance from training to testing data.
    - **High-Performance Measure Values and Consistency:** This model has the ability to make classifications perfectly, which leads to high values in precision, recall, and F1 score and accuracy, which makes it reliable for a wide range of applications.
    - **Balanced:** Provides high performance in both training and testing without outfitting or underfitting, because this model can manage itself to stay balanced.
    - **Strong:** Can handle complex data sets
    - **Handles both categorical and numerical data well**.
    - **Can be used to determine feature importance**.
  + **Weaknesses:** (IBM, 2017) (Wikipedia, 2023)
    - **Model Complexity:** In situations where a lot of trees are needed, the model will be highly computational and will use a lot of resources, especially when dealing with large data sets that need a lot of trees.
    - **Hard to Understand:** The high complexity of this model, makes it hard to understand the exact decision path in a prediction and makes it hard to understand the relations between the inputs, not unlike any simple model.
* **SVM:**
  + **Strengths:** (scikit, 2016) (Wikipedia, 2024)
    - **Strong with complex and large data sets:** This means it achieves high performance even when data is complex and large.
    - **Have a very high ability to Generalize:** which means have the ability to generalize the pattern that the model learns from the training data and apply it to the testing data. In another way, this model can make the same performance on the training and testing data or with a very small drop in the performance from training to testing data.
    - **Kernel trick:** This means that the SVM model can adapt different data types and relationships like the linear kernel.
  + **Weaknesses:** (scikit, 2016) (Wikipedia, 2024)
    - **Scaling Problems:** This model is very sensitive to the scale of the data, that’s why I applied the StandardScaler to my data set without scaling the data the performance of the model will decrease.
    - **Need careful Hyperparameter Tuning:** This model is also very Sensitive to the hyperparameter tuning and kernel parameter.
    - **It is not suitable for large datasets:** because the training time will be so high.
* **Gradient Boosting:**
  + **Strengths:** (Saini, 2024)
    - This model has the ability to offer excellent prediction performance accuracy because this model deals very well with the training data which results in strong predictions.
    - Offering several hyperparameter tuning choices and enabling optimization for different loss functions using Gradient Descent.
  + **Weaknesses:** (Saini, 2024)
    - Easily Overfitted if data is noisy.
    - It takes a long time to train because this model will be highly computational and will use a lot of resources, especially when dealing with large data sets and is very sensitive to hyperparameter tuning.
    - Hard to interpretable, which means it is somehow complex.
* **XGBoost**
  + **Strengths:**
    - High speed in execution, and it provides excellent performance.
    - This model has the ability to handle missing data by itself.
    - Have built-in regularization and cross-validation techniques
  + **Weaknesses:**
    - It might overfit if the parameters were not tuned correctly similar to Gradient Boosting Model.
    - Have a higher complexity than any simple model and also it is hard to tune this model
* **Identify further enhancements which can be done in the future? ﻿Discuss any limitations and future improvements of your project.**
* **Further Enhancements for the Future:**
  + **Use more models that deal with imbalanced data better:** Try another model that deals better with imbalanced data rather than the Gradient Boosting model and XGBoost, because these two models made overfitting.
  + **Deep Learning Techniques:** Use deep learning models that can uncover non-linear relationships in my data, and also because it can deal better with large data sets as mine
  + **Incremental Learning:** Use models that can learn from new data without needing to be retrained from the beginning which will decrease the running time.
  + **Outliers handling:** Apply techniques to handle the outliers that are in my data set.
  + **Handling imbalanced data:** use more under-sampling and over-sampling techniques that enhance the model performance, not like the ones that I have used did not enhance the performance.
  + **Data is not big enough:** When I applied the under-sampling technique the data became very small, and the performance decreased.
  + **Apply regularization techniques:** such as ridge and lasso to prevent models from overfitting.
  + **Feature extraction:** So, I can get new features from the current dataset features that would improve the performance of the models.
* **Limitations and Future Improvements:**
  + **Handling Imbalanced Data:** As you saw in the analyses of the results, the Gradient Boosting and XGBoost models were overfitted, one of the reasons that the data is balanced. Even when I used under-sampling and over-sampling techniques.
  + **Data is not big enough:** When I applied the under-sampling technique the data became very small, and the performance decreased.
  + **Outliers:** I have a lot of outliers that would impact the performance.
  + **Overfitting Risk:** As I said before the Gradient Boosting and XGBoost showed signs of potential overfitting.
  + **Model Complexity:** Gradient Boosting and XGBoost were too complex and not transparent to track the predictions.
  + **Computational Resources:** As I said before Gradient Boosting and XGBoost can be computationally intensive, which increases the running time and needs a lot of resources.

## References

You need to write the references here using the Harvard style of referencing.

IBM (2017) *What is Random Forest?*, *IBM*. Available at: https://www.ibm.com/topics/random-forest (Accessed: 26 January 2024).

Wikipedia (2023) *Random Forest*, *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Random\_forest (Accessed: 26 January 2024).

scikit (2016) *1.4. Support Vector Machines*, *scikit*. Available at: https://scikit-learn.org/stable/modules/svm.html (Accessed: 26 January 2024).

Wikipedia (2024) *Support Vector Machine*, *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Support\_vector\_machine (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).

scikit (2017) *Sklearn.ensemble.gradientboostingclassifier*, *scikit*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html (Accessed: 26 January 2024).

GeeksforGeeks (2023) *XGBoost*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/xgboost/ (Accessed: 26 January 2024).

Simplilearn (2023) *What is xgboost? an introduction to XGBoost algorithm in Machine Learning: Simplilearn*, *Simplilearn.com*. Available at: https://www.simplilearn.com/what-is-xgboost-algorithm-in-machine-learning-article#:~:text=XGBoost%20is%20a%20robust%20machine,optimize%20their%20machine%2Dlearning%20models. (Accessed: 26 January 2024).