**Machine Learning**

**Assignment 2 ML Project**

Dr. Rami Al-Ouran

**Done by: Sama Al-Qaisi**

**Date: 30/1/2024**

**Se**

***Table of Contents***

[**1-** **Introduction:** 2](#_Toc157538626)

[**2-** **Methods:** 4](#_Toc157538627)

[**3-** **Evaluation:** 6](#_Toc157538628)

[**4-** **Results and Discussions:** 8](#_Toc157538629)

[**References:** 12](#_Toc157538630)

**Technical Report**

# **Introduction:**

* Describe the problem you are addressing and why it is important.
* **The Problem:** The problem is determining how many humans are in the room, essential for increasing energy savings and improving the system's adaptability and efficiency. This could result in intelligent heating, air conditioning (HVAC), and lighting and ventilation systems in buildings, improving occupant comfort.
* Describe the dataset's source, collection method, attributes, size, and domain.
* **Data source:** From the UC Irvine Machine Learning Repository site.
* **Collection Method:** The data was collected using the Zigbee-based WSN (Wireless Sensor Network) in a test lab by deploying 5 non-intrusive sensors as nodes distributed around the room then the Arduino Uno that is supplied with the nodes will take a sample of data every 30 seconds at the same time eliminating the PIR and sound sensors, which allows the master node to get the data and records it as a timestamp.
* **Attributes:** (Date, Time, S1\_Temp, S2\_Temp, S3\_Temp, S4\_Temp, S1\_Light, S2\_Light, S3\_Light, S4\_Light, S1\_Sound, S2\_Sound, S3\_Sound, S4\_Sound, S5\_CO2, S5\_CO2\_Slope, S6\_PIR, S7\_PIR). These features show the 5 sensors used: two PIR (passive infrared) sensors, a CO2 sensor, and four sensors for each of the three types of sensors temperature, light, and sound as well as the target column room occupancy count.
* **Size:** (rows:10129, columns:18), and the target column.
* **Domain:** The domain focuses on the energy savings field.
* Describe the learning problem you are trying to solve.
* **Problem Statement:** The problem I am working to address is to predict the estimation of the number of human occupancies in a specific room using specific algorithms Random Forest, SVM, gradient boosting, and XGBoost.
* How did you prepare training and test data before implementing machine learning models?

I have gone through five stages before implementing the machine learning model:

* **Read the data:**

|  |  |
| --- | --- |
| **Technique** | **Justification** |
| dtypes attribute | To discover the data type of each column in the dataset to decide how to deal with them. |
| shape attribute | To discover how many rows and columns “features” are in the dataset. |
| isnull.sum | To check for null values in each column to decide how to handle them. |

* **EDA:**

|  |  |
| --- | --- |
| **Plot** | **Justification + Insights** |
| Heatmap. | To discover if there are any correlations between the columns in my dataset, I have conducted that the columns are correlated in a balanced way but sensor 1 and sensor 3 are correlated by 95% which is a high correlation percentage that may cause multicollinearity, which gives me the choice of deleting it or not based on the results. |
| Bar chart for the target column. | To discover if the data is imbalanced by viewing the distribution of each class in the target column. I have concluded that the data is imbalanced, and I must handle it. |
| Line plot between the targeted column and the time column. | To discover the relation between the average room occupancy throughout the day, I have concluded that:   1. The peak hours of room occupancy are 1 p.m. and 5 p.m. whereas the low occupancy hour is 2 p.m. 2. During the daytime from 10 a.m. to 1 p.m., the room occupancy will be high whereas at lunchtime from 1 p.m. to 3 p.m., the room occupancy will decrease because usually people get out of the room during lunch. 3. At night from 8 p.m. to 9 a.m. the next morning there will be no room occupancy as no one is in the room. |
| 2 Bar plots for each PIR sensor 6 and 7. | I have compared the two PIR sensors to observe if the PIR can affect room occupancy throughout the day because a high correlation indicates that the PIR sensor is effective in detecting occupancy, and which hour of the day has both the PIR and room occupancy is high. I have concluded for sensor 6 and sensor 7 that at 5:00 PM, the average room occupancy is high when there is PIR (detect motion). |
| Distribution plot for all the sensors. | To check for outliers in each of the columns. I have concluded that the S1-Temp, S3-Temp, S4-Temp, and S1-Light have 0 outliers and the rest of the sensors have which gives me the choice of leaving them or removing them, based on the nature of the data I have decided not to remove them. |

* **Preprocessing:**

|  |  |
| --- | --- |
| **Columns** | **Technique** |
| hour, Date, Time, outlier\_zscore, z\_score | Drop these columns. |
| S1\_Temp, S2\_Temp, S3\_Temp, S4\_Temp, S1\_Light, S2\_Light, S3\_Light, S4\_Light, S1\_Sound, S2\_Sound, S3\_Sound, S4\_Sound, S5\_CO2, S5\_CO2\_Slope. | Standardize the values in these columns, meaning the values will have a mean of zero and a standard deviation of 1. |

* **Split the data into x, the features, and y, the target column.**
* **Split the data into training and testing, which is 30% of the data will be used as test data and 70% will be as training data using the train\_test\_split function.**

# **Methods:**

* Explain why the provided models are appropriate to solve this problem.
* The learning problem I am trying to solve is a multiclass Classification problem, which supports choosing these 4 models Random Forest, SVM, Gradient Boosting, and XGBoost.
* Since the data I'm working with is unbalanced, the random forest and SVM models were selected because they offer a class weights parameter that allows you to determine the weights of each class as you see fit. If you choose to set the parameter to balanced, the weights will be adjusted inversely based on the frequencies of the classes. Gradient boosting and XGBoost will also be useful in determining the class weights.
* Random Forest, Gradient Boosting, and XGBoost are Ensemble Methods that are commonly used to suit imbalanced datasets.
* 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).

I have tested the machine-learning models using the test data in two ways: testing it using normal testing and tuning hyperparameters using grid search. Here is a step-by-step explanation:

* **Random Forest/ Support Vector Machine (**normal**):**
* Define the **Random Forest model/ Support Vector** Machine and keep it with its default hyperparameters, except the class weight hyperparameter set it to balance.
* Train the model on the training set after splitting the data into training and testing sets.
* Evaluating the model’s performance on the test set using classification performance metrics.
* **Random Forest/ Support Vector Machine (**hyperparameter tuning using grid search**):**
* Define the **Random Forest / Support Vector** model and keep it with its default hyperparameters.
* Define the hyperparameter grid with the hyperparameter I want, for this model, I have defined for the **RF** the (**n\_estimators, max\_depth, criterion, and class\_weight)**, and for the **SVM** the (**kernel, C, and class\_weight**) and assigned values for each hyperparameter to go through during the grid search.
* Perform Grid Search using cross-validation by going through the hyperparameters and evaluating their performance based on the performed cross-validation.
* Train the grid search on the training set.
* Evaluating the grid search’s performance on the test set using classification performance metrics.
* **Gradient Boosting/ Extreme Gradient Boosting (**normal**):**
* Calculate class weights using a balanced method.
* Define the **Gradient Boosting/ the Extreme Gradient Boosting** model and keep it with its default hyperparameters.
* Train the model on the training set using the computed class weights after splitting the data into training and testing sets.
* Evaluating the model’s performance on the test set using classification performance metrics.
* **Gradient Boosting/ Extreme Gradient Boosting (**hyperparameter tuning using grid search**):**
* Calculate class weights using a balanced method.
* Define the **Gradient Boosting/ the Extreme Gradient Boosting** model and keep it with its default hyperparameters.
* Define the hyperparameter grid with the hyperparameter I want, for this model, I have defined the **GB** (**n\_estimators, max\_depth, criterion, and learning\_rate),** and for the **XGB** (**n\_estimators, max\_depth, criterion, reg\_lambda, and learning\_rate**), and assigned values for each hyperparameter to go through during the grid search.
* Perform Grid Search using cross-validation by going through the hyperparameters and evaluating their performance based on the performed cross-validation.
* Train the grid search on the training set using the computed class weights.
* Evaluating the grid search’s performance on the test set using classification performance metrics.
* Explain in detail the machine learning algorithms you are using to address this problem.
* **Random Forest:**

1. Random forest is a machine learning algorithm that works on combining multiple decision trees to make predictions for both classification and regression problems but in my case, it will work on a classification problem.
2. It reduces the risk of overfitting and increases the accuracy by employing bagging and feature randomness.
3. As an ensemble method for classification, it conducts multiple decision trees on random subsets of the training data which are selected with replacements to ensure variety.
4. Feature randomness is introduced by considering only a subset of features at each split reducing the relation between the individual trees.
5. The algorithm during classification aggregates predictions through the majority votes which minimizes the overfitting associated with individual trees.
6. The inclusion of out-of-the-bag samples not used in the training of specific that was not utilized in the training of decision trees, are included in the Random Forest method and act as an independent validation set for each tree. This makes it possible for the algorithm to assess generalization performance automatically, without the requirement for a different validation set, improving its capacity to manage new, untested material (IBM, 2020).

* **Gradient Boosting:**

1. Gradient Boosting is an ensemble method that works on combining weak learners, usually decision trees to build a strong predictive model.
2. It contains three main concepts:

* Loss function: To evaluate the model predictions.
* Weak learners: Simple models with limited predictive capabilities, commonly used decision trees with high error rates.
* Additive model: To sequentially correct the errors.

1. Each weak learner minimizes the residual errors of its predecessors through gradient descent.
2. The model trains each tree to reduce errors from previous models to improve the performance. The process involves aggregating predictions by adding the outputs of the first tree and the measured versions of subsequent trees.
3. This iterative approach allows the algorithm to improve its predictions and produce an accurate predictive model (paperspace, 2023).

* **Extreme Gradient Boosting:**

1. XGBoost is an ensemble method that works on combining weak learners, usually, decision trees to build a strong predictive model.
2. It contains three main concepts:

* Objective Function: It works on minimizing the cost function by iteratively adjusting model parameters based on the gradients of errors.
* Regularization: It contributes to implementing regularization terms to enhance generalization on unseen data.
* Learning rate: A default parameter that comes with this model that controls the contribution of each tree to the overall prediction.

1. It works by building trees parallelly, adding nodes for each feature at a certain depth before moving to the next level. It evaluates the best splits for each feature at each level and prioritizes them to maximize the objective function. Going level by level reduces the overhead of re-going through the same feature and evaluating multiple times during tree building.
2. It incorporates improvements like regularisation and tree building with the basic ideas of gradient boosting (geeksforgeeks, 2023).

* **Support Vector Machine:**

1. Support Vector Machine is an unsupervised learning model that works on finding the best hyperplane in an N-dimensional space that separates the data points in different classes in the feature space.
2. The hyperplane focuses on separating the data points with dimensions that depend on the number of features.

* Hard margin: This is when the data is separable, so the hyperplane focus separates the data points without misclassification by making the margin between the closest points in different classes as maximum as possible.
* Soft margin: This is when the data is not perfectly separable. It works on acknowledging the outliers or overlapping in classes which works on finding a balance between having a large margin and allowing a limited number of violations.
* Kernel: This is when the data is not linearly separable, this function works on finding the perfect hyperplane in this situation which works on transforming the input data points into a higher-dimensional feature space to make accurate classifications (geeksforgeeks, 2023).

# **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?

I have used the classification performance measures, which are:

* **Accuracy Score metric.**
* **Precision Score metric.**
* **F1-Score metric.**
* **Recall Score metric.**
* Why did you use these metrics?

When I must deal with a multiclass classification task, it is reasonable to use classification measures related to my problem. These metrics include information on how well the models classify instances, capture true positives, keep false positives and false negatives to a minimum, and strike a balance between recall and accuracy.

The following justifies the use of these classifying metrics:

* **Accuracy:** It helps in measuring the overall accuracy of the model in predicting the correct human occupancies. Although this metric measures the correctness however, I can’t just depend on it because my dataset is considered an imbalanced dataset, the accuracy will not be enough to just take results from because it may be misleading, that is why I did more than one measure.
* **Precision:** It helps in measuring the model accurately identifying positive instances among the instances it predicted as positive, so a high value will indicate that the model is likely to be accurate. I will need to minimize the false positive when in scenarios it will predict incorrectly the human occupancy when there is no one in the room.
* **F1-Score:** It is important because it provides a balance between precision and recall. Which is a balance between false positives and false negatives.
* **Recall:** The importance of calculating it is to ensure that the model can correctly identify individuals present in the room out of all the actual cases of human occupancy. A higher recall rate indicates a better ability to correctly predict positive instances, which is crucial for the model's effectiveness in scenarios where accurately identifying human presence is essential.

By using these performance measures in room occupancy prediction, we can assess different aspects of the model's performance, including overall correctness, correctly identifying human room occupancy cases, minimizing false positives, and achieving a balance between precision and recall.

* Evaluate how, based on the performance measures, you were able to enhance the model.

I have tested the models using the default hyperparameters and without balancing the data in the beginning stages of implementation, the performance measures displayed high values, which is expected given the nature of the data is imbalanced, I added balanced class weights to the model after observing that this imbalanced data issue needs to be addressed, which has caused a lower scores of the performance measures which accurately reflected the model’s performance on balanced data. To improve the performance of the model I have implemented a grid search to change the hyperparameters effectively, so I started exploring combinations of hyperparameter values to reach the set with the best and optimized results during these changes I have referred to the performance measures continuously to evaluate how each resulted set is impacting the scores. In addition, while tunning the hyperparameters the measures guided and formed my decision, for example, I started to remove and add hyperparameters and changed their values based on the observed results in the performance metrics.

The ability to change the model repeatedly to match the features of my data led to a better performance, as reflected in the final set of performance measures.

# **Results and Discussions:**

* Discuss the reliability of your results and whether they are balanced, overfitting, or underfitting.

The results I've obtained from using the grid search approach and the normal approach with the default parameters on each model show that they are balanced and neither over- nor underfitted. This conclusion is based on the following insights:

The data's simplicity helps to deliver balanced results. Additionally, there was no evidence of overfitting because, in an overfitting scenario, the testing data should perform poorly, and the training data should perform very well. If there was underfitting, however, both the testing and training data should perform poorly. In my case, the models performed well on both the training and test sets, which is why my findings' reliability is balanced. Additionally, the validity of my findings is further supported by performance metrics that I employed in the process of evaluating my models using the two approaches. I consistently observed high values on both my training and testing sets, and I concluded that the models are well-balanced and that there is no evidence of overfitting or underfitting because they perform consistently and effectively on both seen and unseen data.

* Analyze the results of the applications to determine the effectiveness of the algorithms.

**Random Forest: Gradient Boosting:**

A graph of a bar chart

Description automatically generated with medium confidenceA graph of results with green and blue bars

Description automatically generated

A bar graph with different colored bars

Description automatically generated**Extreme Gradient Boosting: Support Vector Machine:**A graph of a bar chart

Description automatically generated with medium confidence

After the machine learning models stage followed by the visualization stage, they must be followed with the result analysis step because it will help in getting insights into how the performance of the model was whether it performed well or the areas that underperformed.

As you can see above from the charts, the random forest held the higher results of the four measures (accuracy, precision, recall, and f1-score) out of the four models but on the other hand, the GBoost and XGBoost the readings between them were different to varying degrees, and they were closer in results, lastly the   
SVM held the lowest results out of the four models.

Now going deeper into the model's performances, I will be comparing using the accuracy and F1-score measures. The Random Forest model shows higher performance using the grid search approach for both the accuracy and f1-score by **0.998, 0.998** compared to the normal approach which was **0.997, 0.997** as well as the SVM model by **0.9934, 0.9934** compared to the normal approach which was **0.9930, 0.9930**, which indicates that in these 2 models, it is best to tune the hypermeters and train on the best hypermeters, Whereas the GBoost model shows higher performance using the normal approach for both the accuracy and f1-score by **0.997, 0.997** compared to the grid search approach which was **0.996, 0.996** and the XGBoost model by **0.9976, 0.9976** compared to the grid search approach which was **0.9970, 0.9970**, which indicates that using the default parameters for those two models is better than tunning the hyperparameters which will save more computational power and time.

Finally, we concluded that the Random Forest has the best performance overall the other four models Gradient Boosting, Extreme Gradient Boosting, and Support Vector Machine Hyperparameter tuning, specifically through grid search, improved its accuracy and F1 score to 0.998. Also, although they all differ in performance, they all have in common that they performed very well on my data and the results were reasonable.

* Conclude the strengths and weaknesses of the different algorithms.

I have applied 4 different machine learning models, and each of them solved the same problem I have at hand, then compared the results from each model in the analysis part, and now I will go deeper into each model and evaluate it depending on the effect they had on the problem. I will go through each one of the models and present the areas they considered strong choices and the areas they limited their performance.

* **The Random Forest:**

The ensemble method of random forest was approved effective and performed very well in handling our numeric dataset despite the small number of outliers that I didn’t remove and the unbalanced nature of the data, by working on splitting multiple decision trees and collecting their predictions which showed the model ability to capture patterns in the dataset. The fact that the model was flexible with outliers proved to be beneficial in proving that the model performed well despite the presence of abnormal data points. However, Random Forest has its limitations which in my case highlighted the complex interpretability of the model and that was due to the ensemble method nature which made it hard to interpret. Moreover, if I want to experiment with many trees may result in a longer time in training which can be computationally intensive.

* **Gradient Boosting & Extreme Gradient Boosting:**

The strength of this model lies in the ability to showcase feature importance like the random forest model which is good for gaining insights about the impact of features on predictions, also the nature of this model is that it is built sequentially by building the trees iteratively allowing the model to adapt and increasing the accuracy.

In the case of the XGBoost, the key strength lies in the regularization techniques including the lasso(L1) and ridge(L2) techniques to prevent any overfitting, Also a key strength is the tree pruning step which contributes to efficiency and accuracy by removing any additional, non-contributing branches. However, one of its limitations is that it is not suitable for a non-large dataset which in our case the dataset is not large enough.

One of the two limitations of the two models that have been shown clearly in the conducted results is that both are sensitive in hyperparameters tunning which requires a lot of time experimenting to find the best set of parameters which takes a long time in training, that is why in our results they both excel in the normal approach with the default parameters than the grid search approach.

* **Support Vector Machine:**

The key strength of SVM is that it has proved beneficial in handling our non-linearly separable data by implementing kernel functions.

This model was the least well-performed model out of the other models because of some of the SVM’s limitations like the sensitivity to outliers, as I have mentioned before our dataset contains some outliers, I didn’t remove which has affected the decision boundary of the SVM. In addition, SVM was originally built to handle binary classification tasks, although it can use specific techniques to handle multi-class problems this fundamental nature of SVM makes it a limitation in handling our multi-class classification problem.

* Identify further enhancements which can be done in the future. ﻿Discuss any limitations and future improvements of your project.

After being done with implementing the work, we need to provide our work with some recommendations on how to improve our work which will improve performance and help with identifying the places that will need to be improved to enhance performance.

As for my project, I will be providing you with some recommendations that if I had done earlier will change the result that I already got for the better way.

* **First,** the outliers limit the model's performance so I could remove the small number of outliers that I have discovered to improve the performance of the models, especially the SVM because it is sensitive to outliers.
* **Second,** I could experiment more in tunning the hyperparameters in the grid search approach to improve the performance of the models especially the GBoost and the XGBoost models as they are sensitive to hyperparameters tuning.
* **Third,** I have handled the imbalanced data by adding class weights, I could experiment more with the class weights I have assigned to reach the optimal result, or I could use advanced methods to handle my imbalanced data such as performing oversampling and under-sampling.
* **Fourth,** I have implemented the cross-validation in the grid search and assigned the number of folds as 7, I could experiment more with the number of folds by assigning the number of folds higher as equal to 10 or smaller as equal to 5, which allow me to observe how the performance of the model will change and allowing me to choose the optimal result.

# **References:**

geeksforgeeks, 2023. *ML | XGBoost (eXtreme Gradient Boosting).* [Online]   
Available at: https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/  
[Accessed 26 1 2024].

geeksforgeeks, 2023. *Support Vector Machine (SVM) Algorithm.* [Online]   
Available at: https://www.geeksforgeeks.org/support-vector-machine-algorithm/  
[Accessed 26 1 2024].

IBM, 2020. *What is a random forest?* [Online]   
Available at: https://www.ibm.com/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems.  
[Accessed 25 1 2024].

paperspace, 2023. *Gradient Boosting In Classification: Not a Black Box Anymore!* [Online]   
Available at: https://blog.paperspace.com/gradient-boosting-for-classification/  
[Accessed 25 1 2024].