|  |  |
| --- | --- |
| A picture of a winding road and trees  Technical Report  Assignment 2 ML Project | Abstract  This technical report explores the ability to create a machine learning model that would be able to detect the number of occupancies in a room, in order to reduce the amount of money spent on heating, ventilation, and air conditioning systems by making them demand driven.  Professor: Rami Al-Ouran  Student: YOUSEF ABU ALI |

1. Introduction:
   1. The problem and its importance:

* The problem is that in any building a huge amount of energy is spent on heating, ventilation, and the air conditioning systems. And this is important because there are no current methods that control how energy is used on these systems, therefore energy could be wasted without being used efficiently. This shows the importance of having a smart solution to reduce unnecessary power consumption.
  1. Information on the data set:
* This dataset was collected from various sensors that were placed in a lab that is (6 m x 4.6 m) room, which contained four desks, a window, and a door. four sensors gathered information on temperature, light and sound (desks sensors). A sensor for computing CO2. And two sensors that gather motion detection data (window and door sensors). All these data were gathered every couple of seconds, and the final dataset had over 10,000 rows and 16 features. In a four days period.
* Exploratory Data Analysis (EDA) comes before the training, it helps in understanding distribution and the relationship of the features within the dataset. Creating visualizations, such as correlation matrix, and other plots are involved:
* I Visualized the data using violin plots, as they provide information on the distribution of the data, as the width represents the frequency for data point in a specific region. The median in the violin plots also provide information about data distribution and wither there is skewness (left or right). So overall it provides information about the spread of the data. I also used box plots and count plots. And here I will explain my plots:

1. Relationship between room Occupancy and CO2 levels:

This violin plot suggests that when the number of people in the room increases, the level of CO2 in the room will also increase. And it decreases when the number of people decreases. Therefore, it is a good indicator for finding occupancy.

A diagram of different shapes and sizes of objects

Description automatically generated

A diagram of a diagram of a diagram

Description automatically generated with medium confidence

This box plot represents how the slope changes, it is important as it indicates people entering or leaving the room based on co2 changes, it helps in better predicting for Occupancy, as the co2 level could change for multiple reasons, such as a door or a window opening which would reduce Co2 level, However it doesn’t mean that people has left the room.

1. Relationship between room Occupancy and Temperature:

A group of graphs showing different types of light

Description automatically generated with medium confidence

These four violin plots show the temperature in different parts of the room. it clearly shows that the temperature changes between different parts of the room. the first, third, and fourth plots clearly delivers that as the number of people increases the temperature of the room increases, and this makes sense because human body generates heat so the more people their are the higher the temperature will get. however, the plots are not all the same and this is logical as it depends on the location of the sensor and the position of the people in the room.

1. Relationship between room Occupancy and light:

A group of graphs showing different types of graphs

Description automatically generated with medium confidence

These four violin plots show the amount of light at different sensors in the room. it shows that as the number of people increases, the amount of light also increases. however, it is not the same amount across all the sensors.

1. Relationship between room Occupancy and sound:

These plots indicates that as the number of people increases, the amount of sound would increase. another thing that these plots indicates is that the amount of sound never reaches zero, that could be duo to the heating, ventilation and air conditioning systems. as the number of people increases there will be more talk and the conditioning systems will produce higher sounds.

A group of graphs showing different types of numbers

Description automatically generated with medium confidence

1. visualizing the frequency of the target variable:

A graph of a number of people

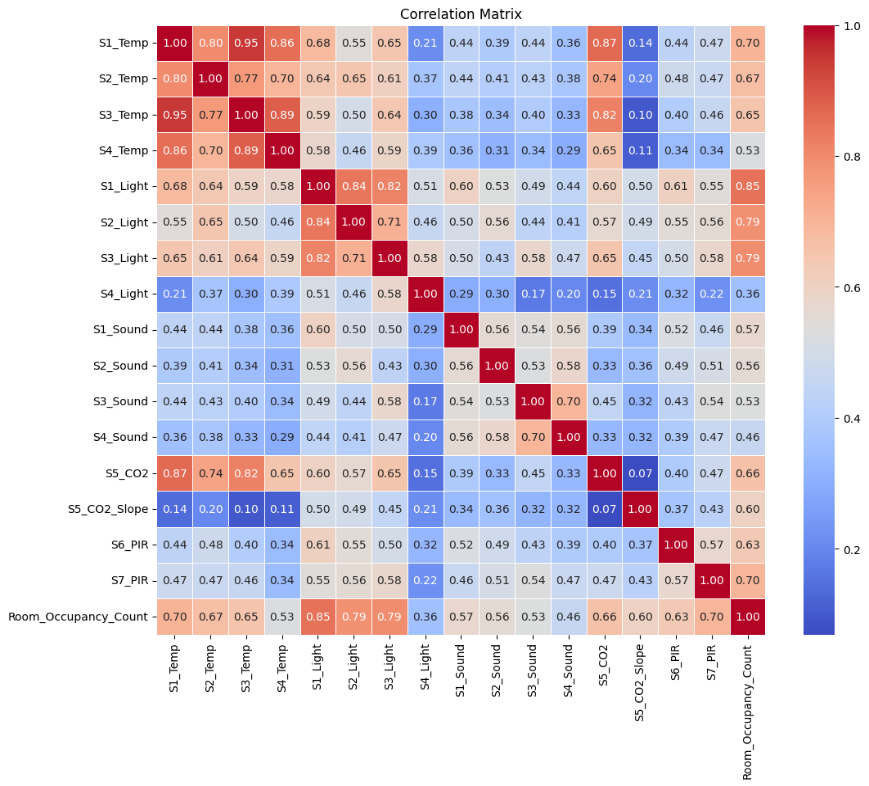
Description automatically generated

This bar plot indicates that class 0 which means there are no people in the room is the most common class. and the classes 1, 2, and 3 are a lot less than the 0 class but they are close to each other.

This might cause a problem while training the models, however it could be normal. but I will try to train the models before balancing the data and I will balance it if needed.

1. Correlation matrix:

I used the correlation matrix to understand the relationship between variables in general, as well as between independent variables and the dependent variable. It shows that the features have good significance on the occupancy and there is no multicollinearity between the data.



* 1. The learning problem I am trying to solve:
* The learning problem involves using machine learning in order to estimate the number of people in A room based on data that was collected from non-intrusive sensors. So, the main goal is to create machine learning algorithms (intelligent systems) that would be able to reduce energy consumption by knowing the number of people in a room and provide energy that would be suitable for the amount of people in the room.
  1. preparation method for data before implementing the machine learning models:

During the preprocessing I:

* I loaded the data using into a data frame named Occupancy.
* I checked for missing values, and this was an important step as missing values could affect the performance of my machine learning algorithms negatively. But there were no missing values.
* I checked for duplicates in the data set which could lead to having biased model training. However, there were no duplicates in the data set.
* I checked for data types in each column of the data set in order to understand my data set better.
* I dropped the date and the time columns from the data set as they want to contribute positively to the occupancy estimation, so their removal would simplify the data set and provide better learning for the algorithms.
* I checked the frequency of each class in the “room occupancy count” column in order to understand the class distribution.
* I calculated the correlation matrix for my data set to understand the relationship between different features and to check for multicollinearity.
* I split the data into training and testing sets in order to use them while evaluating the model’s performance.

1. Methods:
   1. Why the provided models are appropriate to solve this problem:
2. Random Forest:

Random forest has a great ability of handling large datasets, and it is able to capture the nonlinear and complex relationships within the data. It can deal with outliers and overfitting as it is not affected by them (the prediction of individual trees is averaged which helps in reducing variance). Considering our case where features such as light, sound, and temperature are not consistent, random forest can be able to predict energy usage.

1. Support vector machine:

Support vector machine is suitable for classification task it focuses on finding the best hyper plane that is able to separate between different classes which helps in identifying patterns. Moreover, (Support vector machine) doesn't use a lot of memory as it uses only parts of the training data. And it is effective in capturing complex relationships in the data.

1. Gradient Boosting:

Greeting boosting builds multiple trees sequentially with each one of these trees correcting the errors that were done with the previous tree. This helps the model to learn errors and adapt based on them. this method helps in creating better models that are more able to correctly classify new data.

1. XGBoost:

This algorithm is widely used for supervised learning tasks Close to the concept of gradient boosting which builds weak learners (weak models) such as decision trees in parallel allowing for faster training and prediction. It also uses regularization techniques in order to prevent overfitting and it provides various parameters that could be tuned to optimize the model's performance.

* 1. How machine learning applications are going to be tested:

1. Data Splitting:

* Using Train-Test Split the data is split into two subsets: a training set that will be used to train the model and a test set that will be used to evaluate the model’s performance.

1. Model Training: this involves choosing the algorithms that will be used on the dataset. I used random forest, SVM, gradient boosting, and XGBoost. And In each one of them I added hyperparameter tuning to help me find the best optimal hyperparameters for each algorithm in order to provide the best possible results. For example, I used grid search and cross-validation.

* Hyperparameter tuning (Grid search): different models have different hyperparameters. Changing these parameters would change the performance of the machine learning algorithm, either for the best, or for the worst. Tying different hyperparameter manually for each machine learning model will take a lot of time and effort. That is why Grid search is the best option in this case, as it explores different combinations of hyperparameters to find the best possible result.
* Cross validation: cross validation splits the training data to have multiple folds while making sure to take equal proportion of the classes in each fold (stratified).
* Grid search and cross validation: grid search tries all the combination of hyperparameters and for each combination of hyperparameters it performs cross validation. Therefore, for each combination of hyperparameters, the model will be trained and evaluated based on each fold separately.
* Evaluation for the model: the result of the evaluation will be averaged from each fold. Overall, this helps in making sure that the model was tested on different subsets and different hyperparameters of the training data, which helps in assessing its performance better.

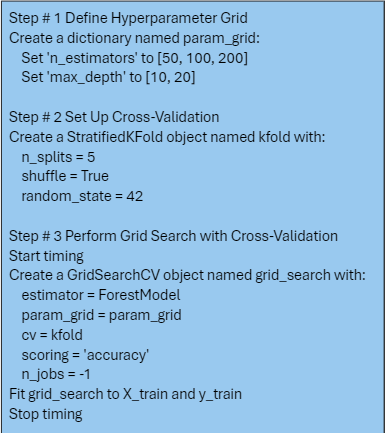
1. Model testing:

The model that is best-tuned will be applied to the test set (data that the model didn't see before)

* 1. Explaining the machine learning algorithm used in details:

1. Random Forest:

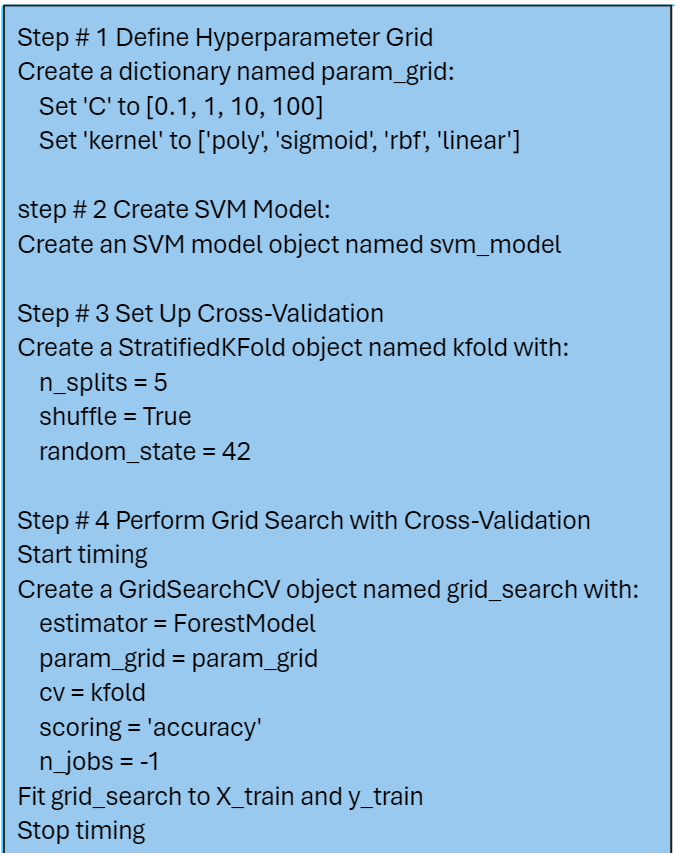
* Random forest uses multiple decision trees in order to make its prediction. Each one of the decision trees will be trained on a different subset of the data. Each tree is built by splitting the data to smaller subsets, however based on the values of the features. This splitting process will continue until a specific condition is met, such as reaching the maximum tree depth. (Meltzer, 2021)
* In order to have diversity within the decision trees, we will take subset of the training data in each tree. This will help in preventing overfitting as well as improving the performance of the model. (graphite-note.com, 2023)
* When making the prediction each decision tree will give a vote for the most common class, and the final prediction will be determined by having a majority vote between individual trees. (graphite-note.com, 2023)
* My pseudocode for the Random Forest model:



* The code works by Employing hyperparameter tuning using grid search along with cross validation. Restart by defining the parameters for the grid search, such as the number of trees(n\_estimators) And the maximum depth of the tree (max\_depth). Then we set up a stratified K-fold cross validation with 5 folds In order to make sure we take proportional samples from each class.
* After that the grid search is executed using the (GridSearchCV). The random first model will be inserted as (ForestModel), and then the grid search will be performed to find the best hyperparameters based on the accuracy scores.

1. Support vector machine:

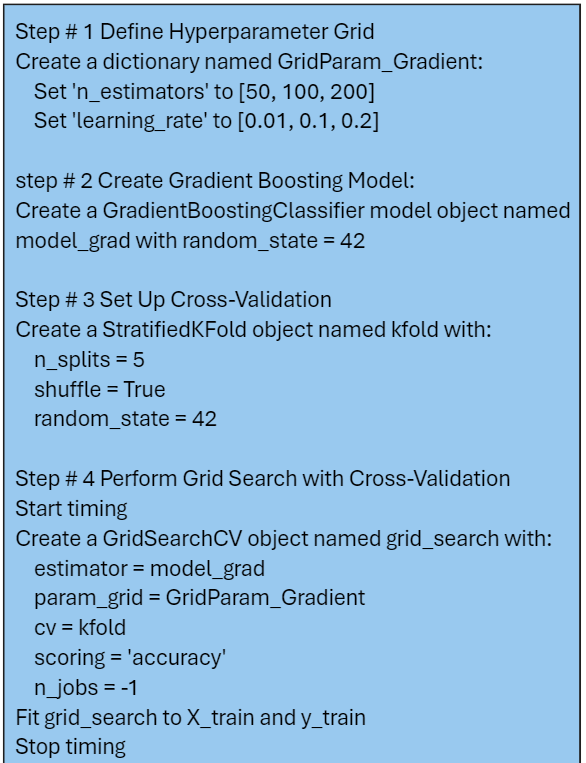
* Support victor machine focuses on finding a hyperplane that separates the data to have different classes while keeping in mind to maximize the margin (the distance between hyperplane and nearest data point from any of the classes) between the classes.
* Machine find the hyperplane by choosing support vectors (data points that are close to the decision boundary). The hyperplane will be chosen so that this margin is maximized while making sure that all the data points are on the correct side of the chosen hyperplane based on their classes.
* When the data is not linearly separable, the support vector machine reduces kernel function to be able to split the data points.
* After the hyperplane is chosen, we can classify new data points by checking at which side of the hyperplane they belonged to. **(Sasidharan, 2021)**
* Pseudocode:



The code for support vector machine conducts hyperparameter tuning and evaluation using grid search approach with cross validation. It starts by specifying parameter for the grid search, for the support vector machine I chose the regularization parameter ( C ) and the kernel function. Then I created the SVM model. In the same way the grid search was initiated using (GridSearchCV) with five-fold cross validation. This VM model is then fitted to the training data and the best parameters are chosen based on the accuracy score.

1. Gradient boosting:

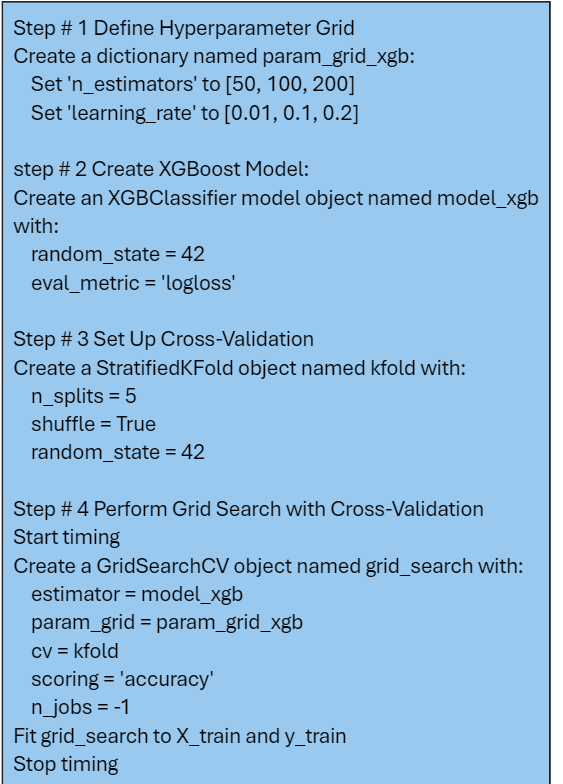
* Gradient boosting is an ensemble method that uses multiple weak learners and combines their prediction no order to create a strong model. The idea behind gradient boosting is to train the weak learners and combine them sequentially by making each week learner correct the error of the previous weak learner. **(Analytics Vidhya, 2021)**
* Gradient boosting classifier uses gradient descent to improve the model. It calculates the gradient for the loss function based on the model's parameters and adjusts them to minimize the loss.
* We train a weak learner that would learn from the mistakes from the previous weak learner by capturing the patterns that were not learned by the previous model.
* Each time we add the prediction of the new week learner did the overall model’s prediction, however it is scaled by the learning rate, which controls the contribution of each weak learner to the overall model.
* We keep repeating the process until a stopping condition is met. And the final prediction would be the sum of each week learner's contribution, and each one of them is scaled by the learning rate.
* Of course, to prevent overfitting we could use what's called early stopping, which would stop the training process if the performance stops improving.
* Pseudocode:



* The code starts by defining hyper parameter grid containing different values for a number of estimator(number of weak learners) that will be used, and different learning rates. And the rest is the same as the previous code.

1. XGBoost:

* The (extreme greedy boosting) algorithm it's similar to the principle of gradient boosting but includes optimization for better performance.
* Xgboost uses regularization terms such as L1(lasso) and L2(ridge) regularization in order to control the complexity of the model.
* Also chooses tree pruning which is done by removing the unimportant branches which would improve the model’s performance.
* So supports early stopping which would stop the training process once the performance stops improving.
* XGBoost uses parallel processing, where the decision trees would be built in parallel instead of being built sequentially.



* We start by initializing hyperparameter great search with values for a number of estimator and values for the learning rate. I created an instance for the XGBoost classifier with the evaluation and metric log loss. And as the rest of the model, I used cross validation in the grid search to find the best optimal results

1. Evaluation:
   1. The performance measures I used to evaluate the effectiveness of the models:

* Accuracy: which are the correctly predicted instances on the total number of predictions. It provides information on how the model is overall performing.
* Precision: the ratio of correctly predicted positives on the total number of predicted positives. It gives information on the accuracy of the positive predictions which would minimize false positives.
* Recall : Which is the ratio of the correctly predictive positive over all the actual positives.
* F1-score : this measure provides an average between precision and recall.
* Confusion matrix
  1. The reason of using these measures:
* Accuracy: to get a general idea of how the model is able to predict occupancy classes correctly.
* Precision : the precision because I want to keep tracking the false positive as I might need it to be minimized. If the false positives are high that means waste of energy.
* Recall : the precision because I want to keep tracking the false negatives as I might need it to be minimized. If the false negatives are high that might also mean waste of energy.
* F1-score : as in my case I think it would provide the best result as it keeps in mind the precision and the recall for each class. So, it provides a metric that consider the overall models ability to correctly classify the instances in all the classes.
* Macro average: finds the precision recall and F1 score for each class and then it gets an average value for all the classes, and it treats all the classes in an equal way regardless of their size, and it is suitable when all classes are considered important.
* Weighted average: it's averages the values of the precision recall and F1 score but it depends on the number of instances for each class, as it gives more importance to larger classes, so this is suitable for class imbalance.
* Confusion matrix: this table provides information on the performance of the classification algorithm. It provides information on the quantity of errors for each class.
  1. How I was able to enhance the model based on these performance measures:

To enhance my model based on the performance measure I used, I started by using grid search and cross validation techniques for the model in order to explore multiple hyperparameter combinations and to ensure a good evaluation across different subsets of the data set. Now by continuously monitoring the key performance metrics such as accuracy, precision, recall , and F1 score I was able to understand how the model is performing. So, I was able to adjust the hyperparameters, and the algorithms themselves. I keep checking whether new hyper parameters would improve the performance measures or reduce them.

1. Results and discussion:
   1. Reliability of the results:

The reliability of the results where established through analyzing the accuracy metrics, and showing the accuracy scores between training and testing sets. The alignment of the results indicates that there is no overfitting. and the accuracy for both of them were high which indicates that there is no underfitting. This shows that the model was the able to capture the patterns in the data. Also, the result’s of the confusion matrix in all algorithms were very high, indicating that the model is generalizing well in all evaluation measures (F1-score, precision, recall).

* 1. effectiveness of the algorithms:

All things considered, the algorithms proved to be very successful in finding the best answers. Each algorithm performed grid search with cross-validation, while trying different combinations of hyperparameters. The models were able to maximize their results by finding the optimal parameters through this method. The outcomes resulted from each algorithm showed not only how well-suited they are to the dataset, but also how they are well-tuned .The algorithms performed exceptionally well because of the use of grid search and cross-validation, shoeing their effectiveness in handling the presented problem.

* 1. conclusions regarding the strengths and weaknesses of the different algorithms:

The following is a table that shows the different algorithms with their best results, and I will compare between them overall:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Metrics🡪  Models | Train accuracy | Test accuracy | Precision avg | Recall avg | F1-score avg | Misclassified data points | Time for training in seconds |
| Random Forest | 0.9998 | 0.9990 | 1 | 1 | 1 | 2 | 27.41 |
| SVM | 0.996 | 0.995 | 0.99 | 0.98 | 0.98 | 10 | 62.63 |
| Gradient Boosting | 1.0 | 0.996 | 0.99 | 0.99 | 0.99 | 6 | 211.06 |
| XGBoost | 1.0 | 0.997 | 0.99 | 0.99 | 0.99 | 7 | 21.67 |

1. Random forest

* Strengths
* Achieved the high accuracy on the testing sets.
* All the performance metrics(precision, recall, F1 score) are perfect.
* Fast training time compared to SVM and gradient boosting.
* Has the least misclassified data points.
* Weaknesses
* Had less training accuracy compared to gradient boosting and XG boost.

1. support vector machine

* Strengths
* It's accuracy on training and testing was high.
* all performance metrics were high.
* Weaknesses
* It had the lowest results in the performance metrics (accuracy, recall, F1 score) compared to the other models.
* It had the most misclassified data points compared to the other models.
* Although it was fast but it had more running time than random forest and XGBoost.

1. gradient boosting

* Strengths
* Achieved the highest accuracy on the training set.
* All the other performance metrics were considered high.
* Weaknesses
* It had more misclassified data points compared to random forests.
* It took the most training time.

1. XGBoost

* Strengths
* Achieved the highest accuracy on the training set.
* All the other performance metrics were considered high.
* It took the least training time among all the machine learning models.
* Weaknesses
* It had more misclassified data points compared to gradient boosting and random forest.

Overall, all the machine learning models performed well on the data set provided. However, I think that random forests provided the best results in comparison to the other models. Most performance measures in it were the highest, the difference between training and test accuracy was very small, and it had the misclassified data points and this is very important.

* 1. Father enhancement that could be done in the future including limitations and future improvement:

Limitations:

1. The data set was somehow small, as the data was collected in four days only.
2. There was a significant difference between the classes in the occupancy feature, as class zero had more instances than the rest of the classes.
3. The features in the data set had a lot of outliers.

Future enhancement:

1. I could perform feature extraction, instead of having multiple features that deliver the same information, such as sound, temperature, and light. I could perform feature extraction to find the feature that delivers the information instead of having 4 columns for each one of these features. For example using (PCA).
2. I could drop the columns that had a correlation above 90%.
3. Collecting more data about such problem with more classes might provide a better and more realistic machine learning algorithms that would be ready to be used to real world scenarios.
4. Trying a bigger range of hyperparameters in each algorithm.
5. Trying to handle class imbalance by using techniques such as oversampling or under sampling.

References:

1. Chartio. (n.d.). A Complete Guide to Violin Plots. [online] Available at: <https://chartio.com/learn/charts/violin-plot-complete-guide/>.
2. JavaTpoint (n.d.). Machine Learning Random Forest Algorithm - Javatpoint. [online] www.javatpoint.com. Available at: <https://www.javatpoint.com/machine-learning-random-forest-algorithm>.
3. Brownlee, J. (2020). Bagging and Random Forest for Imbalanced Classification. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/bagging-and-random-forest-for-imbalanced-classification/>.
4. Paperspace Blog. (2020). Gradient Boosting for Classification. [online] Available at: <https://blog.paperspace.com/gradient-boosting-for-classification/>.
5. scikit learn (2018). 1.4. Support Vector Machines — scikit-learn 0.20.3 documentation. [online] Scikit-learn.org. Available at: <https://scikit-learn.org/stable/modules/svm.html>.
6. Editorial (2022). Pros and cons of Support Vector Machine (SVM). [online] RoboticsBiz. Available at: <https://roboticsbiz.com/pros-and-cons-of-support-vector-machine-svm/>.
7. www.krayonnz.com. (n.d.). Krayonnz : Social Learning Network. [online] Available at: <https://www.krayonnz.com/user/doubts/detail/623b2b7235e21e005f953106/what-are-the-advantages-and-disadvantages-of-XGBoost>.
8. Meltzer, R. (2021). What is Random Forest? [Beginner’s Guide + Examples]. [online] careerfoundry.com. Available at: <https://careerfoundry.com/en/blog/data-analytics/what-is-random-forest/>.
9. raphite-note.com. (2023). A Comprehensive Guide to Random Forest: How It Works and Its Applications | Graphite Note. [online] Available at: <https://graphite-note.com/a-comprehensive-guide-to-random-forest-how-it-works-and-its-applications>
10. Sasidharan, A. (2021). Support Vector Machine Algorithm. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/support-vector-machine-algorithm/>.
11. https://www.facebook.com/jason.brownlee.39 (2018). A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/>.
12. Analytics Vidhya. (2021). Gradient Boosting Algorithm | How Gradient Boosting Algorithm Works. [online] Available at: <https://www.analyticsvidhya.com/blog/2021/04/how-the-gradient-boosting-algorithm-works/>.