**Wine quality prediction using ML Model**

Himanshu Baid SHIVAM JAISWAL UDAY SINGH SHERGILL Vellore Institute of Technology Vellore Institute of Technology Vellore Institute of Technology

Chennai Chennai Chennai

-D.r Triloknath Panday

Vellore institute of technologyChennai

***Abstract***—**Drinking wine may not be something we Indians may look up to but when we consider the broader world, wine drinking is a culture followed by many countries and people regard it essential for their living, and such wine is not cheap either. So, people expect to get quality goods for the money they pay. But there always are a set of people who aim to cheat people out of their money and earn some profit. It also involves health risks since drinking bad wine may cause altitude of problems for both those who drink it and are around them in various ways.**

**Keywords:- boosting, bagging, machine learning. Customer relationship, data analysis.**

# **INTRODUCTION**

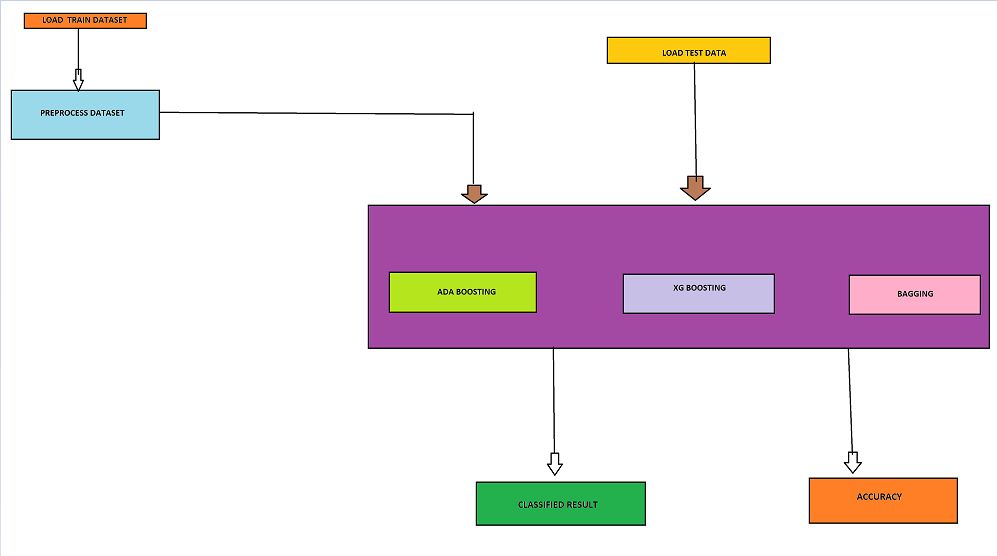
Wine industry shows a recent growth spurt as social drinking is on the rise. The price of wine depends on a rather abstract concept of wine appreciation by wine tasters, opinions among whom may have a high degree of variability. Pricing of wine depends on such a volatile factor to some extent. Another key factor in wine certification and quality assessment is physicochemical tests which are laboratory-based and take into account factors like acidity, pH level, the presence of sugar and other chemical properties ensure that the standards are not established subjectively.

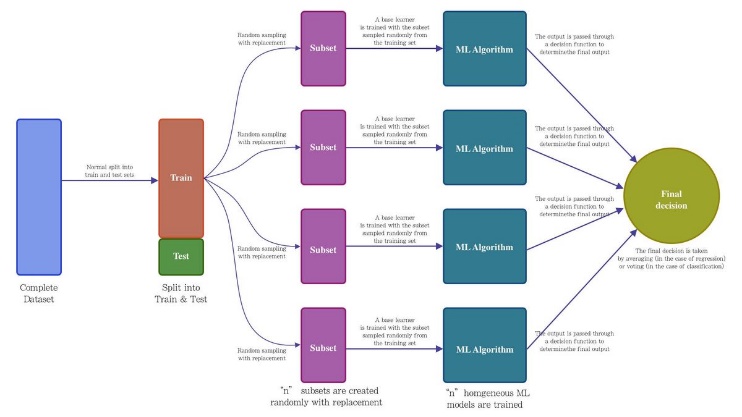
For the wine market, it would be of interest if human quality of tasting can be related to the chemical properties of wine so that certification and quality assessment and assurance process is more controlled. So, we want to study the importance of the features for the prediction of wine quality to see which yields the highest accuracy and to determine which features are the most indicative of a good quality wine.

II. **DATASET DETAILS**

We collected the dataset and used it as a source of description about each type of wine to analyze each variable. A predictive model developed on the dataset is expected to provide guidance to vineyards regarding quality and price expected on their produce without heavy reliance on the volatility of wine tasters. A prescriptive model developed on the dataset is expected to provide knowledge to vineyards about best quality wines and prescribing which variable is their wine lacking in order make it better quality for the consumers.

III. **ARCHITECTURE DESIGN**





**IV. LITERATURE REVIEW**

1. A literature review published in the "**IEEE Access**" journal in 2020, the authors reviewed the recent research on wine quality prediction using machine learning techniques. They found that various machine learning algorithms, such as support vector machines, random forests, and artificial neural networks, have been used for wine quality prediction. The review also highlighted the importance of feature selection and data preprocessing for improving the accuracy of wine quality prediction models.
2. A literature review published in the "**IEEE Transactions on Industrial Informatics**" in 2019 analyzed the use of machine learning for wine quality prediction based on sensor data. The authors found that machine learning models such as support vector machines, decision trees, and random forests have been used for wine quality prediction using sensor data such as temperature, pH, and sugar content. The review also highlighted the potential of machine learning for optimizing wine production processes and improving wine quality management.
3. A literature review published in the "**IEEE Sensors Journal**" in 2021 analyzed the use of machine learning for wine quality prediction based on electronic nose data. The authors found that machine learning models such as artificial neural networks, support vector machines, and decision trees have been used for wine quality prediction using electronic nose data. The review also highlighted the potential of machine learning for developing low-cost and non-invasive methods for wine quality assessment.

The risk in this process is, since the vineyards are old players, they may try to bribe the central institution's board to change their results which may lead to improper data collection which makes the whole process meaningless. There won't be any ethical issues since the data is about an inanimate object and it’s a customer commodity.

**VI. MODULES USED**

* + 1. **Decision Tree**

Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Construction of Decision Tree: By dividing the source set into subgroups based on an attribute value test, a tree can be "learned". It is known as recursive partitioning to repeat this operation on each derived subset. When the split no longer improves the predictions or when the subset at a node has the same value for the target variable, the recursion is finished. Decision tree classifier building is ideal for exploratory knowledge discovery because it doesn't require parameter configuration or domain understanding. High-dimensional data can be handled via decision trees. Classifiers in decision trees typically have good accuracy. A popular inductive method for learning classification information is decision tree induction.

Decision Tree Representation: Decision trees categorise instances by arranging them in a tree from the root to a leaf node, which gives the instance's categorization. As seen in the above diagram, to classify an instance, one tests the attribute given by the root node of the tree before continuing down the branch of the tree that corresponds to the attribute's value. The subtree rooted at the new node is then subjected to the same procedure once more.

According to whether a specific morning is ideal for playing tennis, the decision tree in the above diagram assigns a classification to each leaf and returns that classification.(in this case Yes or No).

This code imports necessary libraries for data processing, model training, and evaluation.

Here, the data is split into input variables (X) and target variable (y). The input variables consist of all the columns except for the 'quality' column, which is the target variable.

Split the data into training and testing sets using the train\_test\_split function from the sklearn library. The test\_size parameter specifies that 20% of the data will be used for testing, and random\_state sets the seed for the random number generator used by the function for reproducibility.

Then we use the trained decision tree to make predictions on the test set and calculates the accuracy of the model using the accuracy\_score function from the sklearn library. The accuracy is then printed to the console, and the lengths of y\_test and y\_pred are printed for information.

After that we converts y\_pred and y\_test to numpy arrays, subtracts them element-wise, and converts the resulting numpy array to a list called 'differece', which is then printed to the console.

Plot histograms of the differences between y\_pred and y\_test, as well as histograms of y\_test and y\_pred separately, using the matplotlib library.

Extract the number of times the value '4' appears in both y\_test and y\_pred separately.

Finally, plot a scatter to represent the relationship.

* + 1. **ADA BOOSTING**

AdaBoost, also called Adaptive Boosting, is a technique in Machine Learning used as an Ensemble Method. The most common estimator used with AdaBoost is decision trees with one level which means Decision trees with only 1 split. These trees are also called **Decision Stumps.**

To apply AdaBoost to wine quality prediction, you can follow these general steps:

* Collect and preprocess the wine quality data. This may include cleaning the data, removing duplicates and outliers, and splitting the data into training and testing sets.
* Choose an appropriate machine learning model to use as the "weak" classifier for AdaBoost. For wine quality prediction, you can use algorithms such as decision trees, random forests, or support vector machines (SVMs).
* Train the weak classifier on the training data.
* Use the AdaBoost algorithm to combine multiple instances of the weak classifier into a single "strong" classifier. In AdaBoost, each instance of the weak classifier is given a weight based on its accuracy, and the final prediction is made by weighting the predictions of each weak classifier.
* Evaluate the performance of the AdaBoost classifier on the testing data. You can use metrics such as accuracy, precision, recall, and F1-score to measure the performance.
* If necessary, tune the hyperparameters of the AdaBoost algorithm to improve the performance.
* Once you are satisfied with the performance of the AdaBoost classifier, you can use it to make predictions on new, unseen wine quality data.

The code is using the AdaBoost classifier algorithm from the scikit-learn library to build a machine learning model to predict outcomes in two different datasets.

In the first dataset, the code reads wine quality data from a CSV file and splits it into training and testing sets. The AdaBoost Classifier model is then trained on the training data, and its accuracy is evaluated on the testing data.

In the second dataset, the code generates a synthetic dataset using the make classification function from scikit-learn. Again, the AdaBoostClassifier model is trained on the training data and evaluated on the testing data.

In the third example, the code uses the AdaBoost classifier to train a decision tree classifier on a breast cancer dataset. The accuracy of the model is evaluated, and then k-fold cross-validation is used to further validate the model's performance. Finally, the confusion matrix and classification report are printed to show the model's performance on the test set, and a heatmap of the confusion matrix is plotted using the seaborn library.

* + 1. **XG BOOSTING**

XGBoost is a distributed gradient boosting library that has been optimised for quick and scalable machine learning model training. A number of weak models' predictions are combined using this ensemble learning technique to get a stronger prediction. Extreme Gradient Boosting, or XGBoost, is one of the most well-known and widely used machine learning algorithms because it can handle large datasets and perform at the cutting edge in many machine learning tasks like classification and regression.

Its effective handling of missing values, which enables it to handle real-world data with missing values without requiring a lot of pre-processing, is one of the key characteristics of XGBoost. Additionally, XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time.

Applications for XGBoost include click-through rate prediction, recommendation systems, and Kaggle competitions among others. Additionally, it is quite adaptable and enables speed optimisation by allowing for fine-tuning of numerous model parameters.

Extreme Gradient Boosting, or XgBoost, is a concept put out by University of Washington researchers. It is a C++ library that enhances the training process for gradient boosting.

Decision trees are generated sequentially in this approach. Weights are significant in XGBoost. Each independent variable is given a weight before being fed into the decision tree that forecasts outcomes. Variables that the tree incorrectly predicted are given more weight before being placed into the second decision tree. These distinct classifiers/predictors are then combined to produce a robust and accurate model. It can be used to solve problems including regression, classification, ranking, and custom prediction.

This code block imports the necessary libraries for the code to run, including pandas and numpy for data manipulation, GridSearchCV and KFold for cross-validation, train\_test\_split for splitting data into training and testing sets, XGBoost for the gradient boosting algorithm, and various metrics functions.

This code block imports the seaborn library, which is used for data visualization, but it doesn't actually prepare the dataset.

This code block splits the dataset into training and testing sets, where 30% of the data is reserved for testing. X\_train and y\_train are the training features and labels, respectively, while X\_test and y\_test is the testing features and labels, respectively.

This code block trains an XGBoost regression model using the training data. The model is defined with the objective of minimizing the squared error, with 100 estimators, a learning rate of 0.05, and a maximum depth of 5.

This code block evaluates the XGBoost model using the testing data. The predicted values for the testing data are generated using the predict() method, and accuracy and mean squared error metrics are calculated using the predicted values and the actual labels. The results are then printed.

This code block performs hyper parameter tuning for the XGBoost model. The hyper parameters to tune are maximum depth, learning rate, and number of estimators. A GridSearchCV object is created with the xgb\_model\_tuned as the estimator, params as the parameter grid to search, and a cross-validation of 3. The best parameters are then printed.

* + 1. **BAGGING**

A machine learning ensemble meta-algorithm called bootstrap aggregating, commonly referred to as bagging, is created to increase the stability and accuracy of machine learning algorithms used in statistical classification and regression. It reduces variation and aids in preventing overfitting. Usually, decision tree approaches are where it is used. A specific instance of the model averaging approach is bagging.

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.

To implement bagging for wine quality prediction, you can follow these steps:

* Split the wine dataset into training and test sets.
* Create multiple subsets of the training data using random sampling with replacement.
* Train a decision tree model on each subset of the training data.
* Use each model to predict the quality of the wine in the test set.
* Combine the predictions from all the models using either averaging or majority voting to get the final prediction.

The code is performing various operations on the wine dataset, using several machine learning algorithms and evaluating their performance.

It’s using the sklearn library to create an ensemble Bagging classifier that is trained on the wine dataset. It first sets a random seed, then creates a Decision Tree Classifier with a maximum depth of 5 and sets the number of estimators to 6. The classifier is then fit on the training data (X\_train, y\_train), and the resulting model is stored in the variable mdl.

It again creates a Bagging classifier, but with different parameters. This time it sets the number of estimators to 100, the maximum samples to 0.8, and enables the out-of-bag score calculation. The classifier is then fit on the test data (X\_test, y\_test), and the out-of-bag score is printed.

Then we create a Decision Tree Classifier and fits it on the entire wine dataset (X, Y), and then calculates the accuracy score of the model.

The next block of code uses the Decision Tree Classifier to create a visual representation of the decision tree.

There we use the Bagging classifier with a Decision Tree Classifier as an estimator to perform 5-fold cross-validation on the wine dataset, and prints the average accuracy score.

We use the same Bagging classifier as in the previous step, but this time, it uses 10-fold cross-validation to evaluate the model's accuracy.

The final block of code calculates the confusion matrix and classification report for the model using the test data, and then visualizes the confusion matrix using a heatmap.

In summary, the code is performing various operations on the wine dataset, including training and evaluating machine learning models, visualizing decision trees, and calculating performance metrics such as accuracy, confusion matrix, and classification report.

**Similarities between Bagging and Boosting:**Both of the widely used approaches, bagging and boosting, are consistently categorised as ensemble methods.

* Both ensemble procedures start with one student and produce N learners.
* Both use random sampling to produce various training data sets.
* By averaging the N learners, both get to their final judgement. (or taking the majority of them i.e Majority Voting).
* Both effectively lessen variance and offer greater consistency.

**Differences between Bagging and Boosting**

1. Bagging is the simplest way of combining predictions that belong to the same type whereas boosting is a way of combining predictions that belong to the different types.

2. Bagging aims to decrease variance, not bias whereas Boosting Aim to decrease bias, not variance.

3. In Bagging each model receives equal weight whereas in boosting Models are weighted according to their performance.

4. In bagging each model is built independently whereas in boosting New models are influenced

by the performance of previously built models.

5. In bagging different training data subsets are selected using row sampling with replacement and random sampling methods from the entire training dataset whereas in boosting every new subset contains the elements that were misclassified by previous models.

6. Bagging tries to solve the over-fitting problem whereas Boosting tries to reduce bias.

7. If the classifier is unstable (high variance), then apply bagging and If the classifier is stable and simple (high bias) the apply boosting.

8. In Bagging base classifiers are trained parallelly. In Boosting base classifiers are trained sequentially.

9 Example: The Random forest model uses Bagging. Example: The AdaBoost uses Boosting techniques

**Comparison of XGBoost and AdaBoost**

1. **Loss function:**

A variety of loss functions are used in the boosting technique. AdaBoost, also known as adaptive boosting, minimises the exponential loss function, which can make the algorithm more susceptible to outliers. Any differentiable loss function can be used with gradient boosting. AdaBoost is less resistant to outliers than the gradient boosting algorithm.

1. **Flexibility:**

The first boosting algorithm with a specific loss function was called AdaBoost. Gradient Boosting, on the other hand, is a general technique that helps in the search for approximations to the additive modelling issue. Gradient Boosting is hence more adaptable than AdaBoost.

1. **Benefits:**

AdaBoost works best with weak learners and minimises the loss function associated with every classification error. The technique can be used to improve the performance of decision trees and was primarily created for binary classification issues. The problem of the differentiable loss function is solved using gradient boosting. The method can be applied to problems involving classification and regression.

1. **Shortcomings:**

Gradient Boosting uses gradients to identify the weaknesses of the current weak learners, while AdaBoost uses high-weight data points to do the same.

1. **Performance Metrics:**

In data analytics, accuracy, precision, recall, support, and F1-score are commonly used metrics to evaluate the performance of different ensemble techniques. Here's what each of them represents:

**Accuracy:** Accuracy is the proportion of correct predictions out of the total number of predictions made. In other words, it measures how well the model is able to classify the data correctly.

**Precision:** Precision is the proportion of true positive predictions out of the total number of positive predictions made. In other words, it measures how many of the positive predictions made by the model are actually correct.

**Recall:** Recall is the proportion of true positive predictions out of the total number of actual positive instances in the dataset. In other words, it measures how well the model is able to identify all the positive instances in the dataset.

**Support:** Support is the number of instances in the dataset that belong to a particular class.

**F1-score:** F1-score is the harmonic mean of precision and recall. It provides a single score that balances both precision and recall, and is often used as an overall measure of the model's performance.

When comparing different ensemble techniques, you would typically look at these metrics for each technique and choose the one that performs the best overall. However, it's also important to consider other factors such as computational complexity, scalability, and interpretability when selecting an ensemble technique.

In order of highest correlation, these variables are:

* Alcohol: the amount of alcohol in wine
* Volatile acidity: are high acetic acid in wine which leads to an unpleasant vinegar taste
* Sulphates: a wine additive that contributes to SO2 levels and acts as an antimicrobial and antioxidant
* Citric Acid: acts as a preservative to increase acidity (small quantities add freshness and flavour to wines)
* Total Sulphur Dioxide: is the amount of free + bound forms of SO2
* Density: sweeter wines have a higher density
* Chlorides: the amount of salt in the wine
* Fixed acidity: are non-volatile acids that do not evaporate readily
* pH: the level of acidity
* Free Sulfur Dioxide: it prevents microbial growth and the oxidation of wine
* Residual sugar: is the amount of sugar remaining after fermentation stops. The key is to have a perfect balance between — sweetness and sourness (wines > 45g/ltrs are sweet)

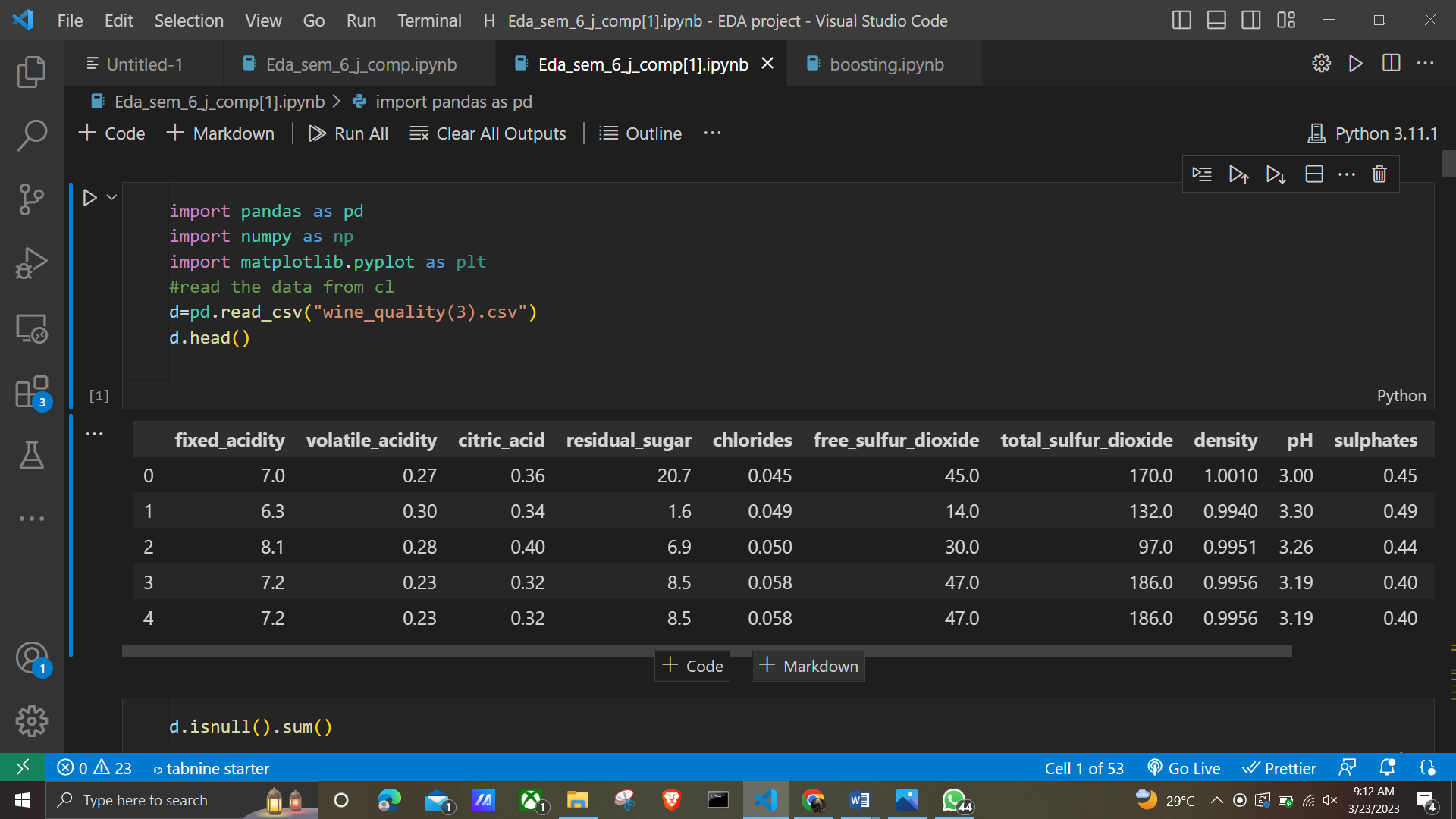
**Applying a basic machine learning algorithm:**

**Decision tree**

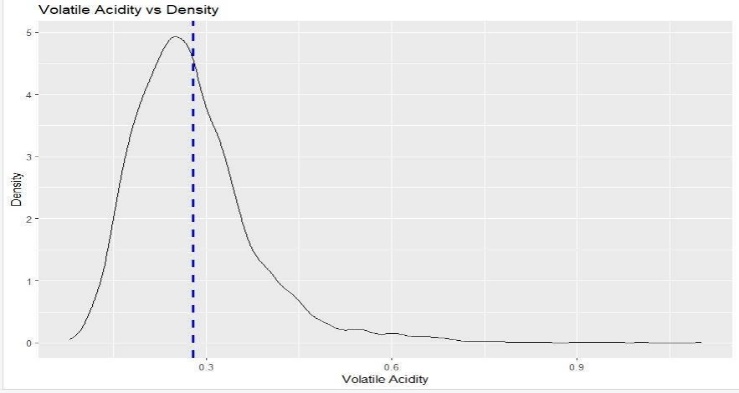
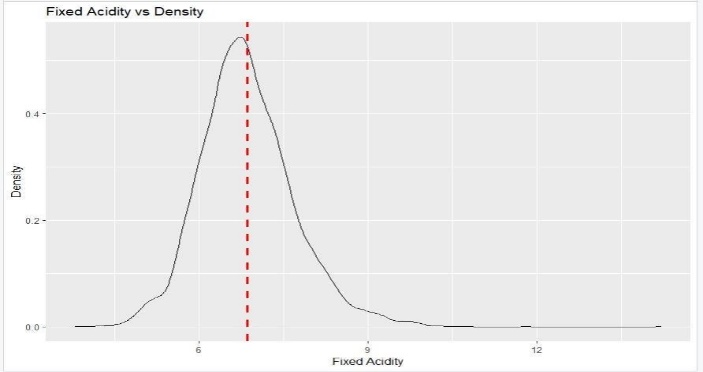
We took the wine quality dataset from kaggle that has attributes (fixed\_acidity, volatile\_acidity,

citric\_acid, residual\_sugar chlorides, free\_sulfur\_dioxide, total\_sulfur\_dioxide, density, pH, sulphates, alcohol , quality)

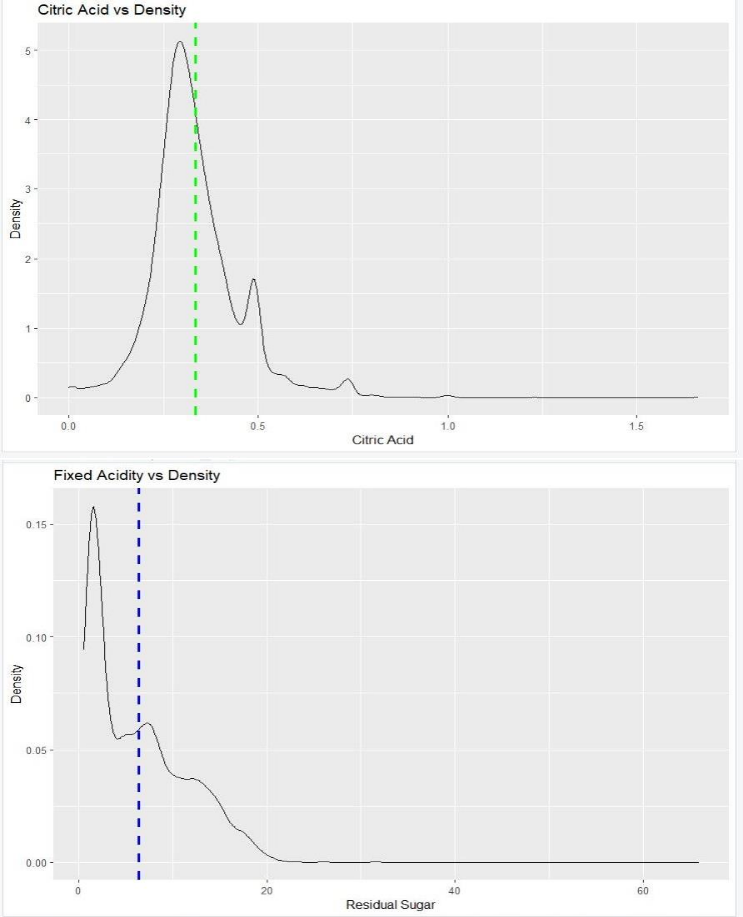
**Dataset**



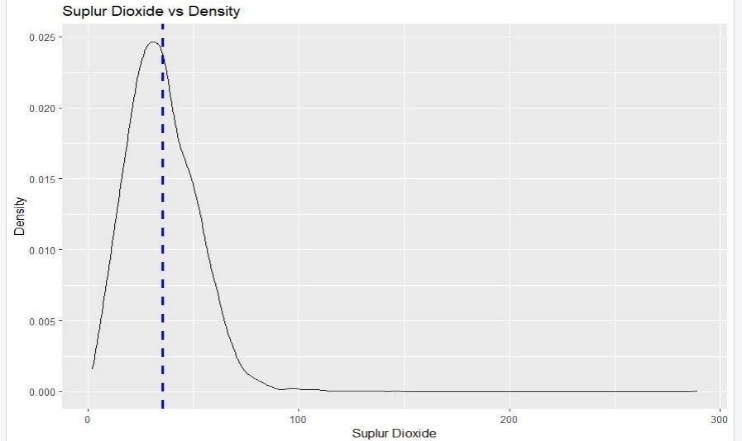
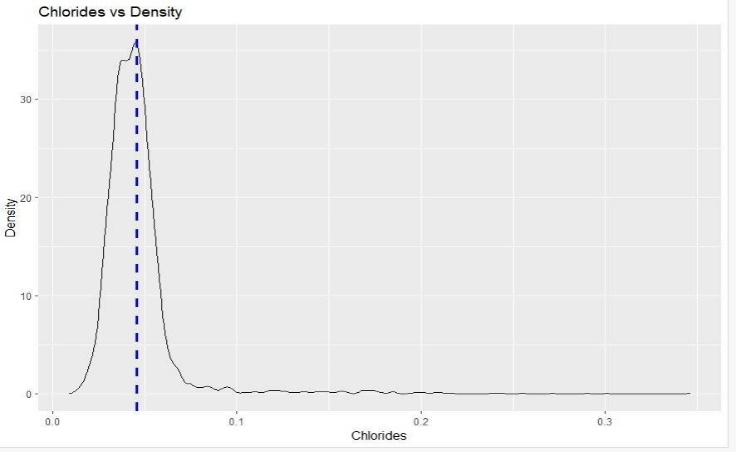
**Analyzing the attributes using R**



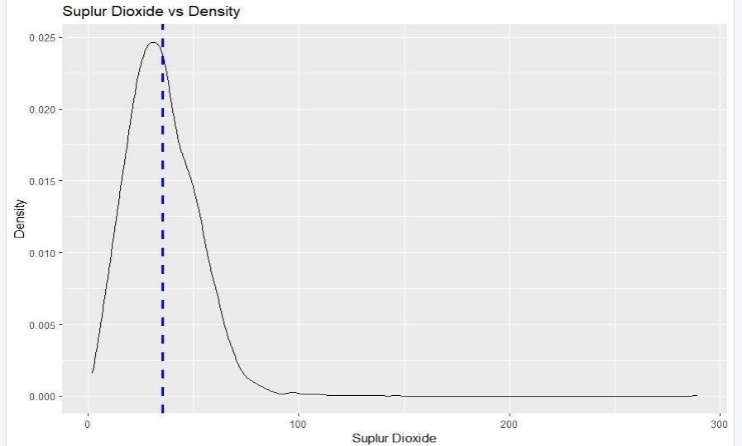
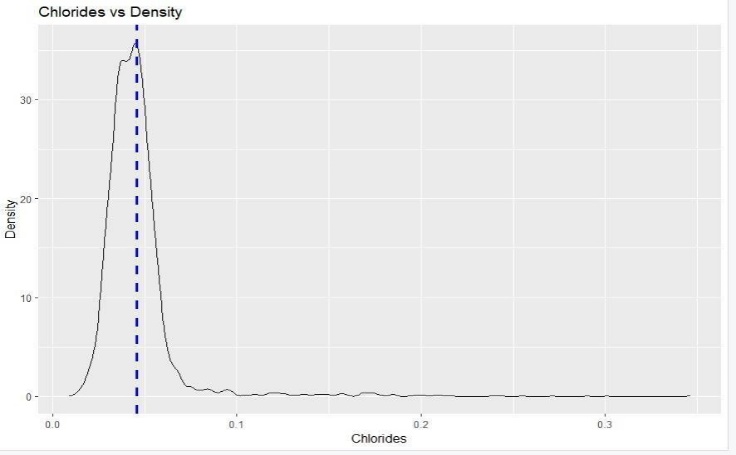
**Figure 1**



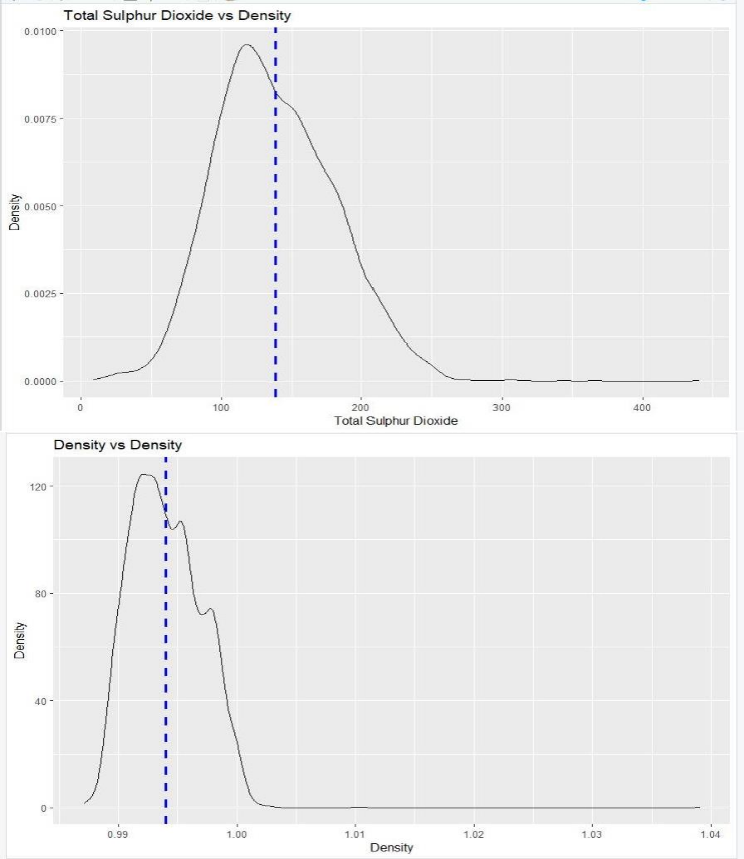
**Figure 2**



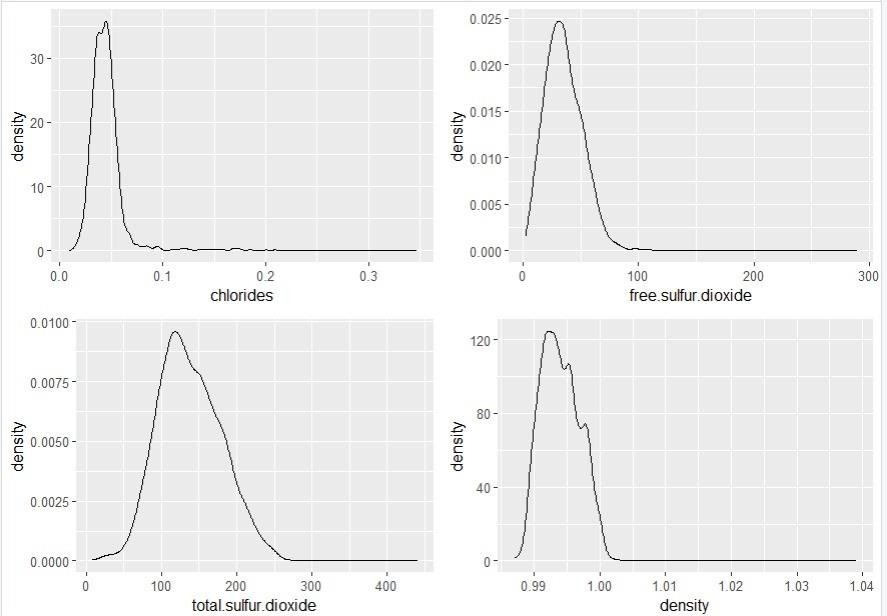
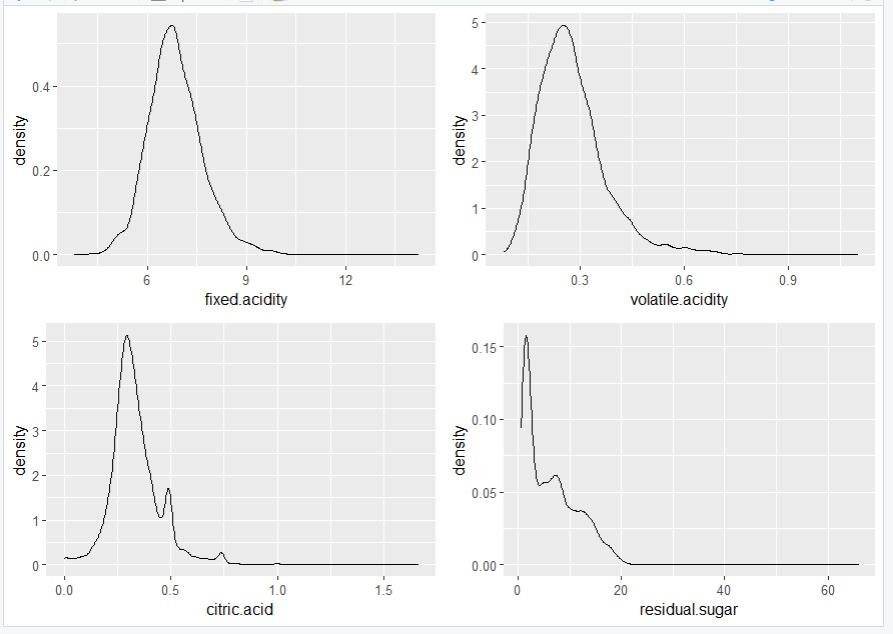
**Figure 3**



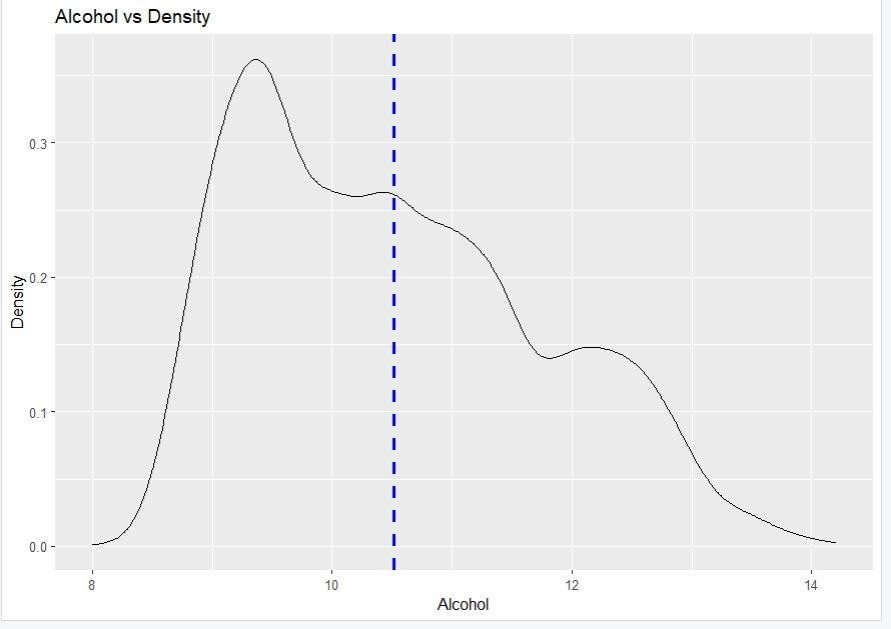
**Figure 4**



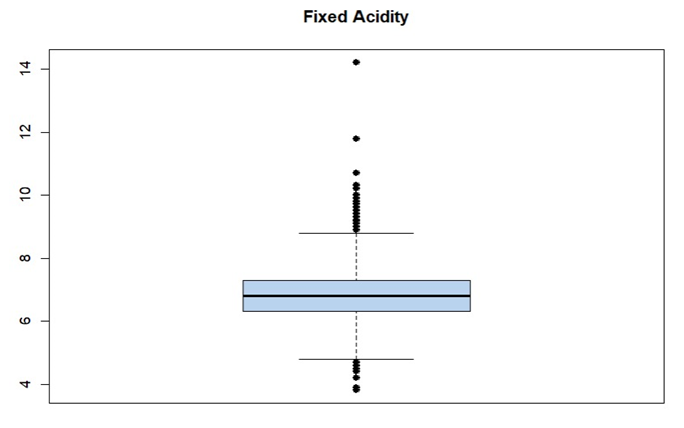
**Figure 5**



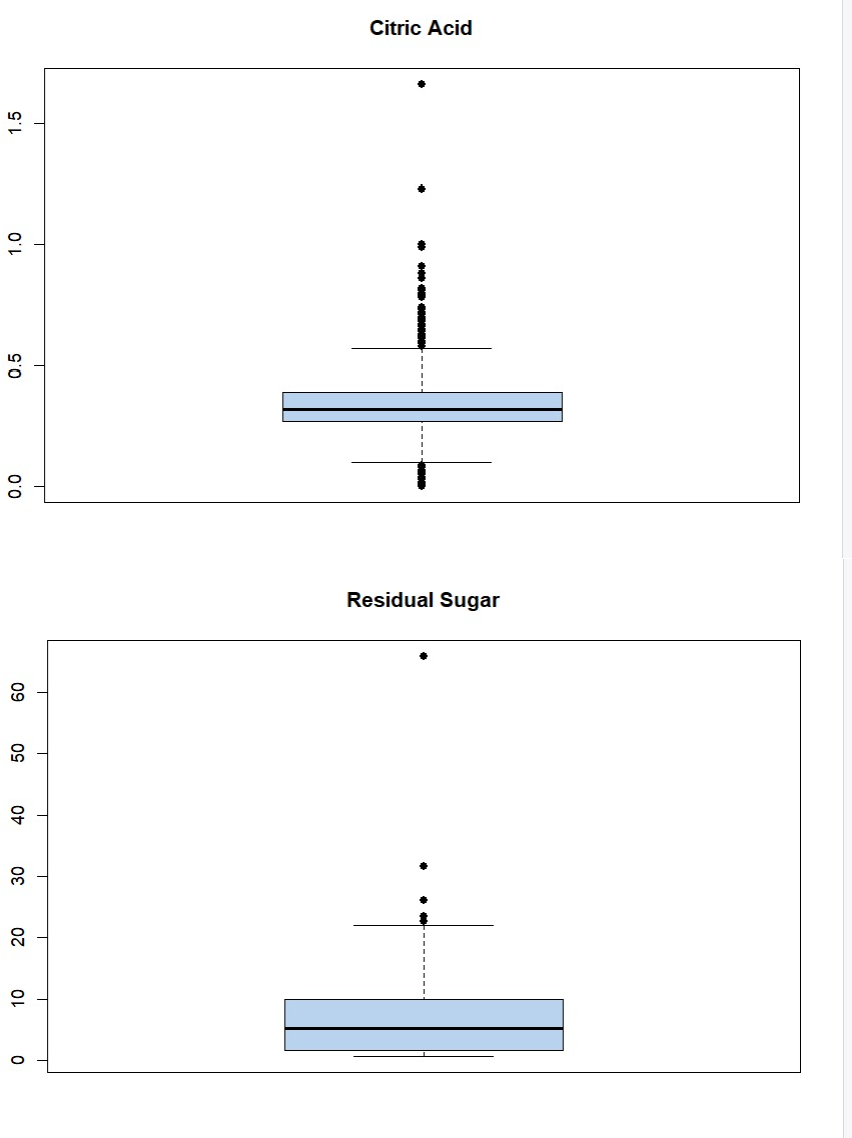
**Figure 6**



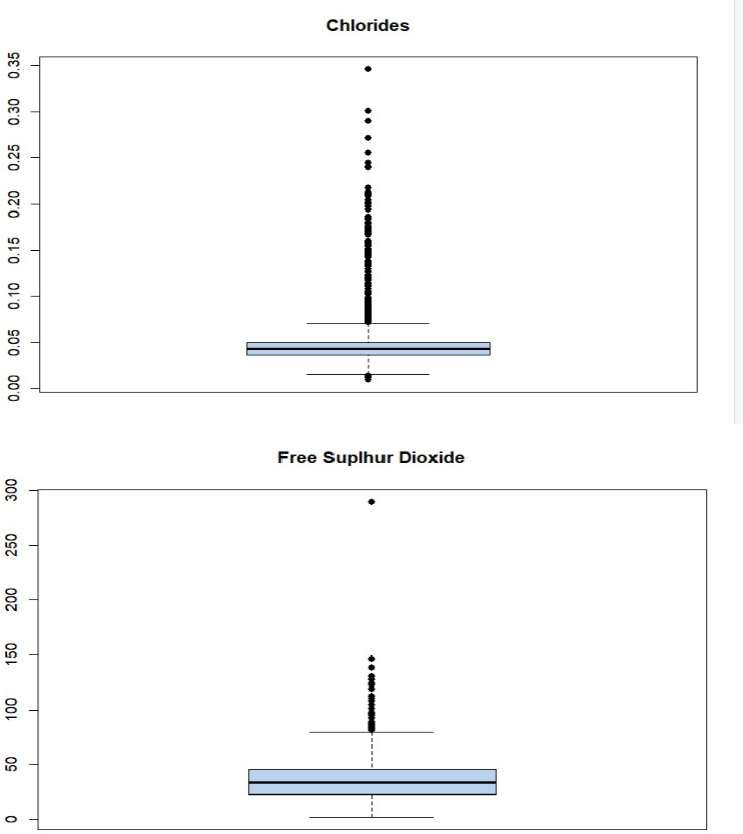
**Figure 7**



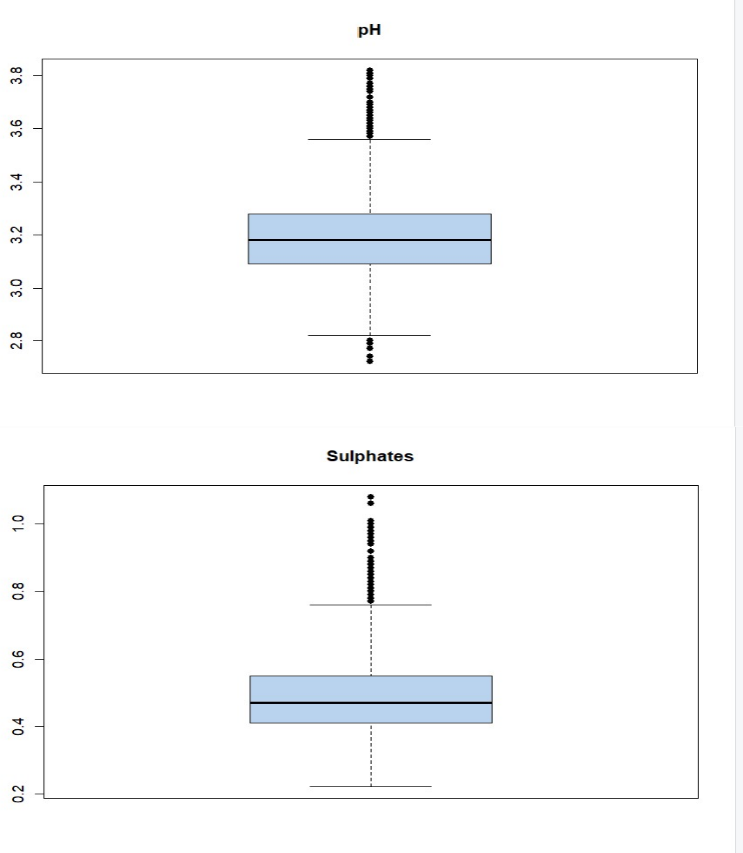
**Figure 8**



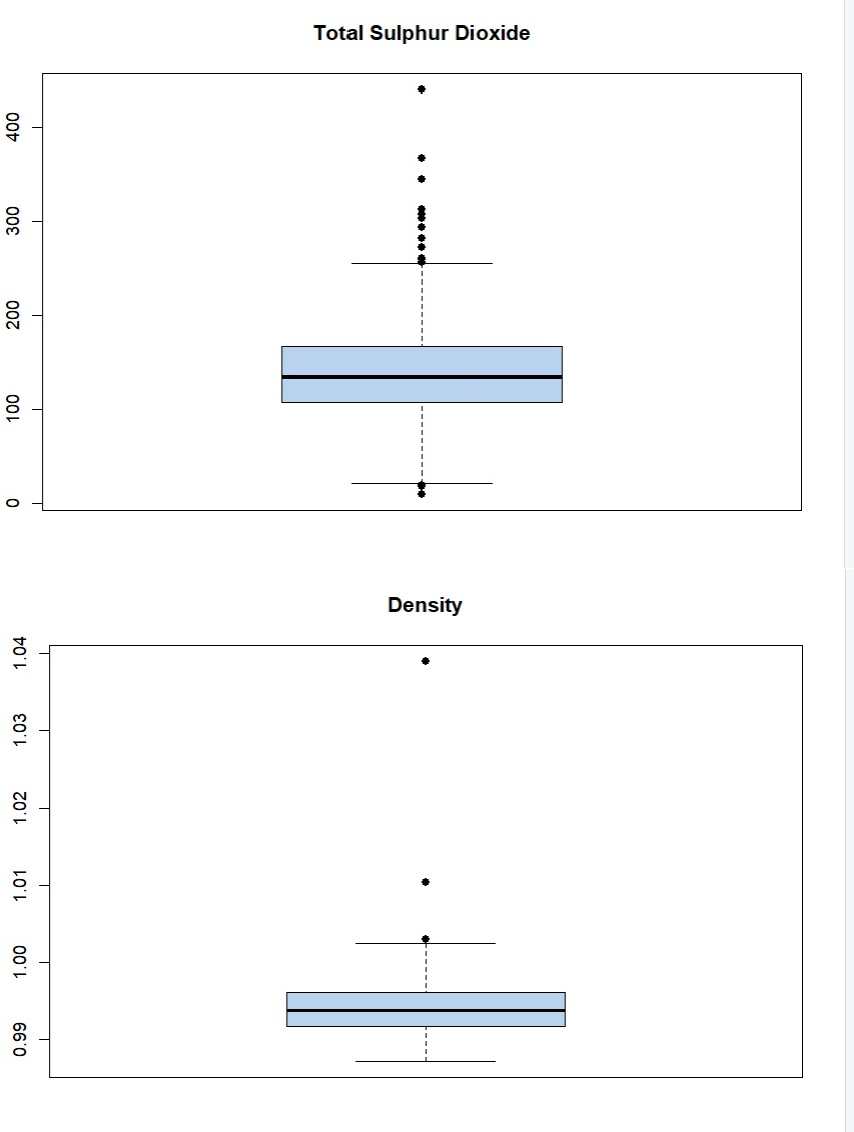
**Figure 9**



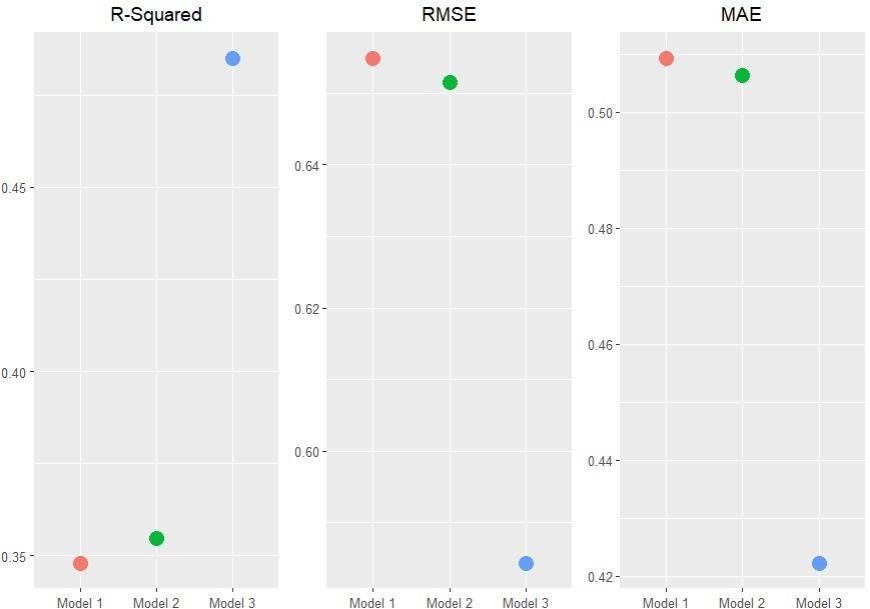
**Figure 10**



**Figure 11**

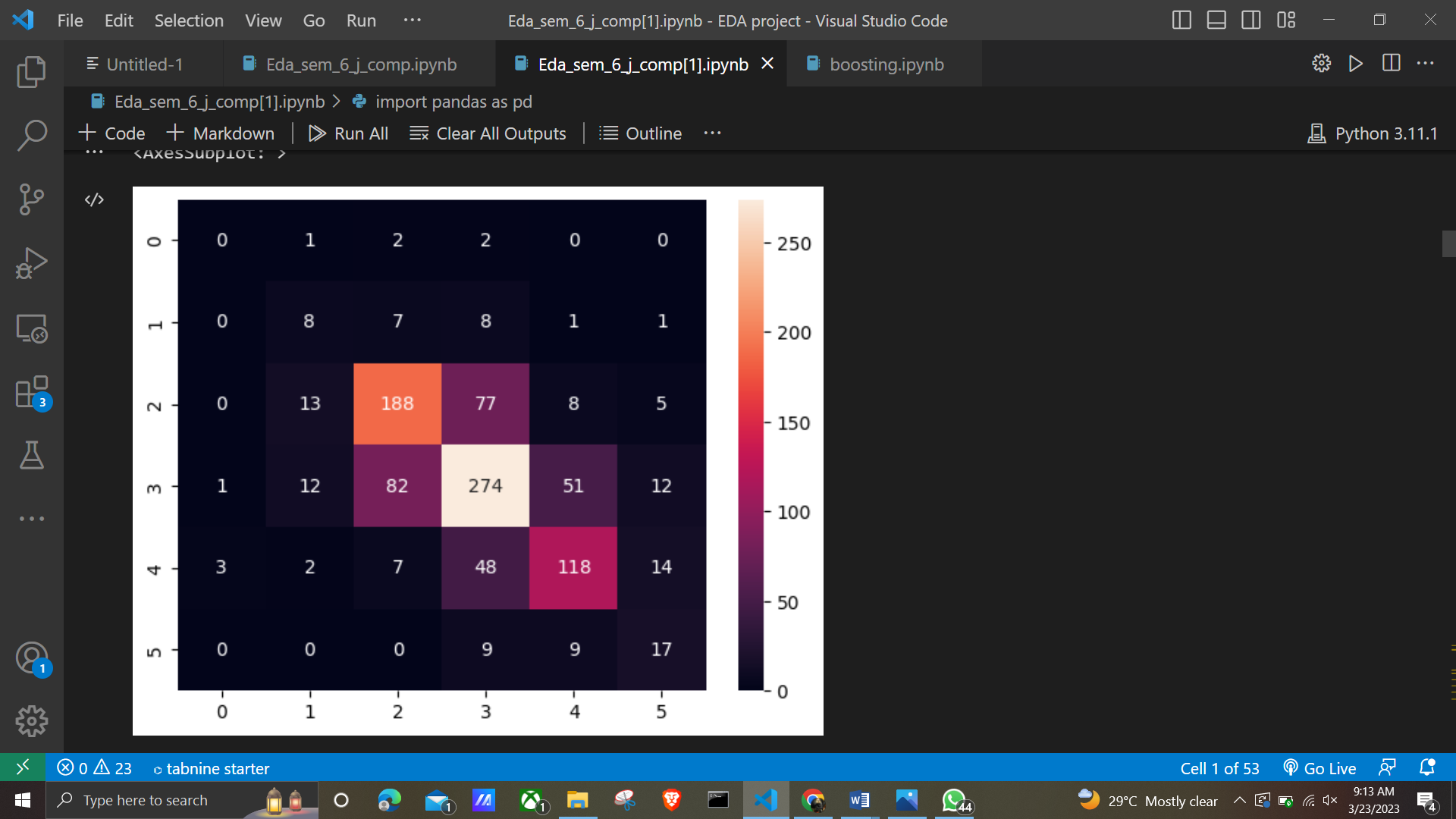


**Figure 12**



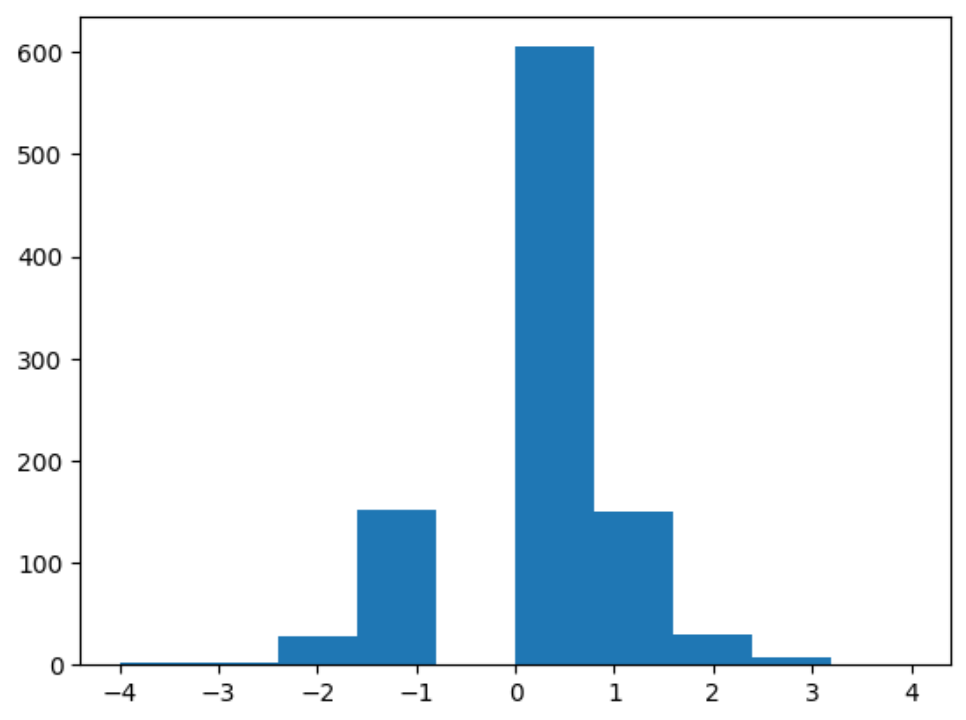
**Figure 13**

**Confusion matrix:**

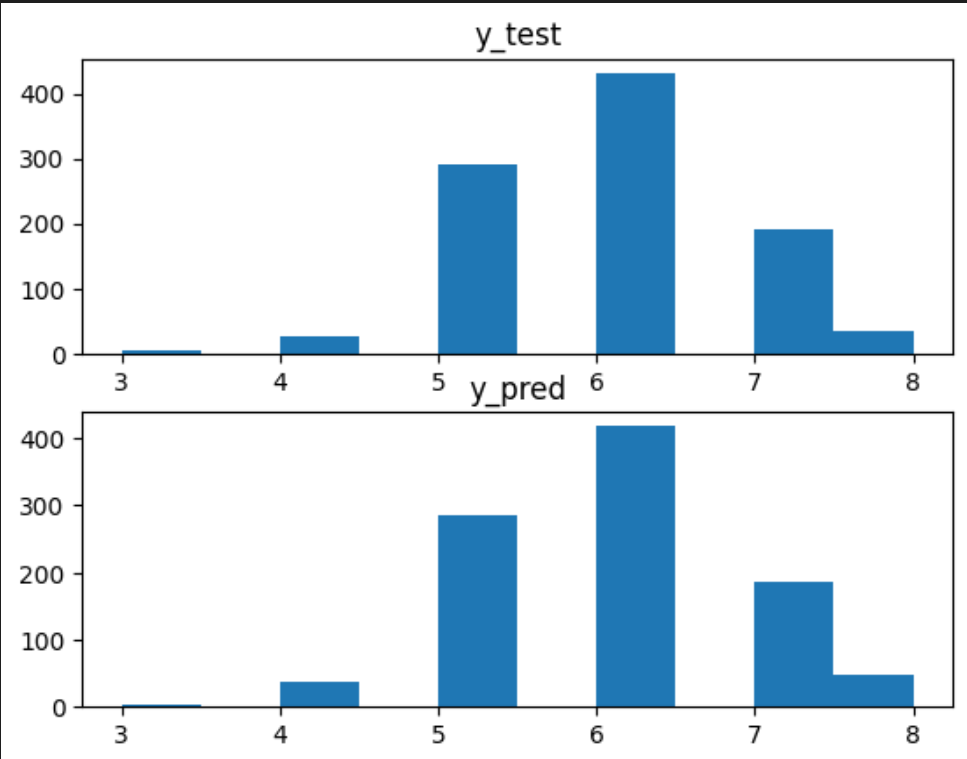


**Figure 14**

**Y\_test vs Y-pred(bar plot)**

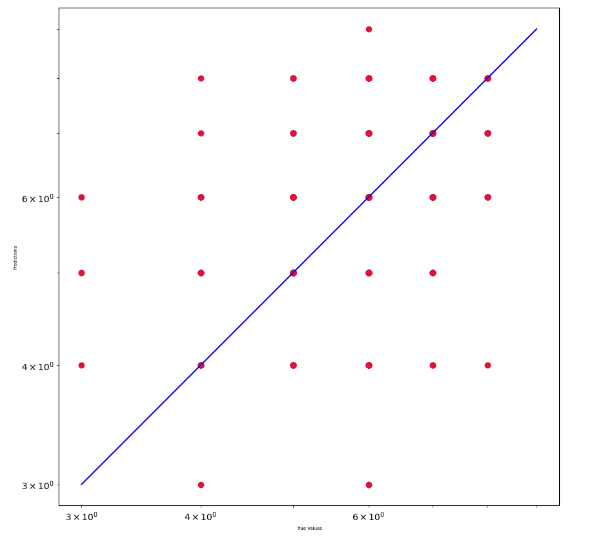


**Figure 15**



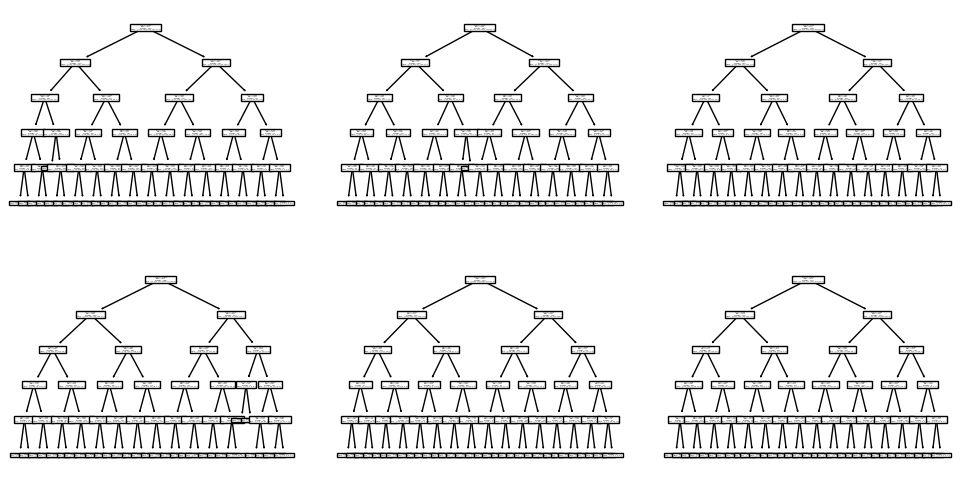
**Figure 16**

**Scatter plot:**



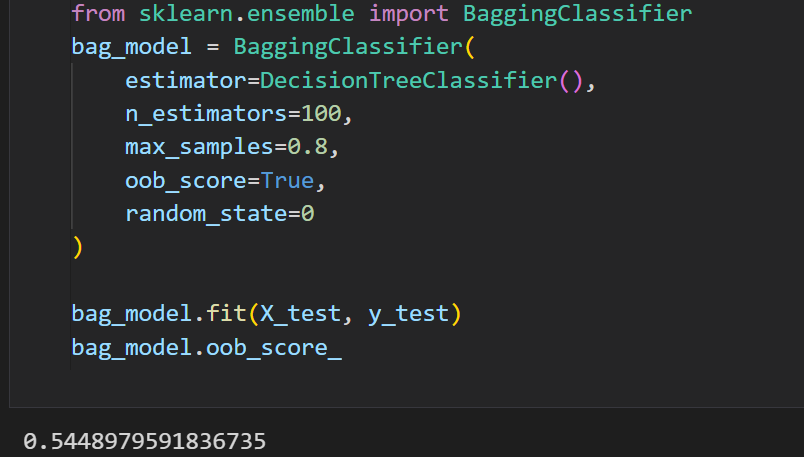
**Figure 17**

**Decision tree:**

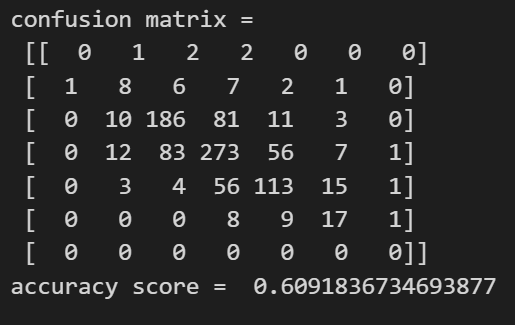


**Figure 17**

**Bagging model score:**

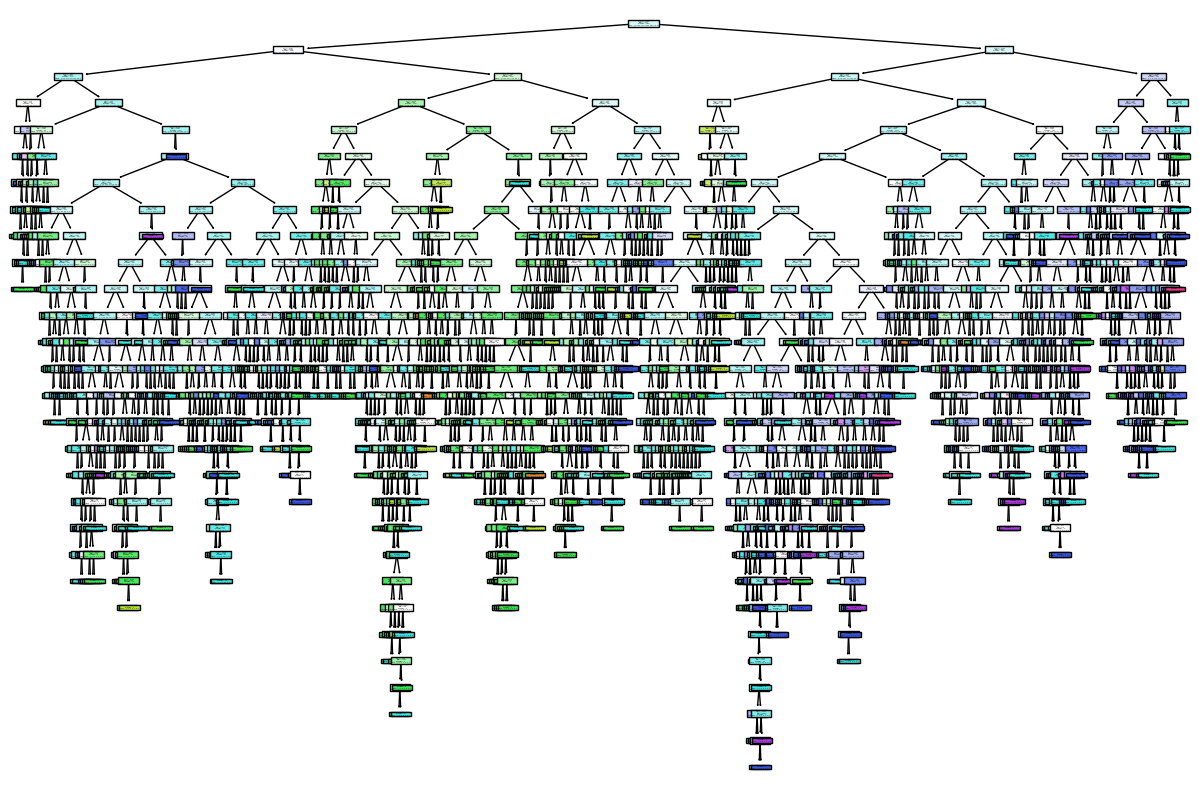
**Figure 18**

**Confusion Matrix**



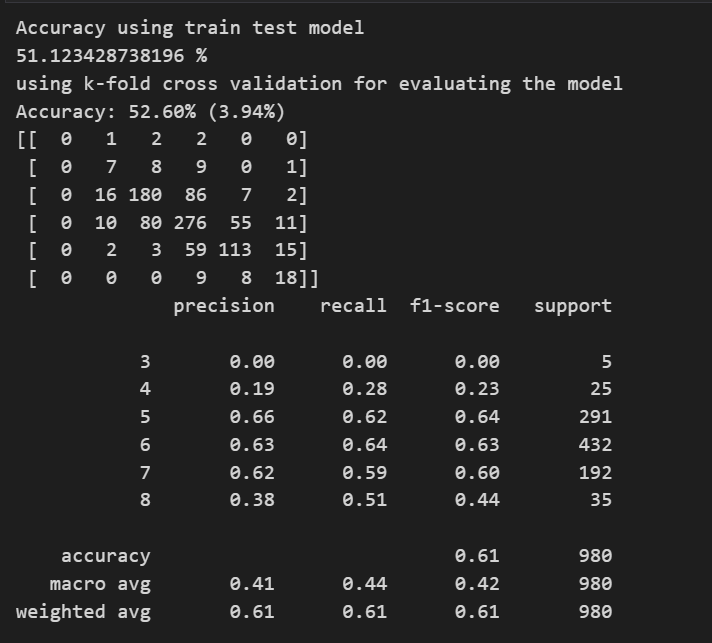
**Figure 19**

**Plot**

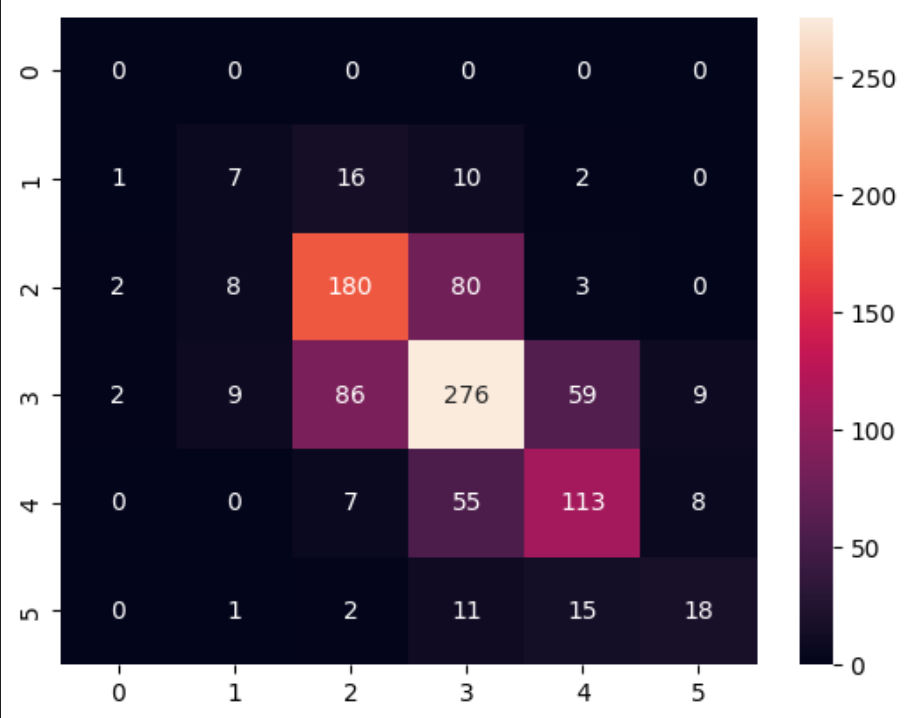


**Figure 20**

**Evaluating Bagging Model:**

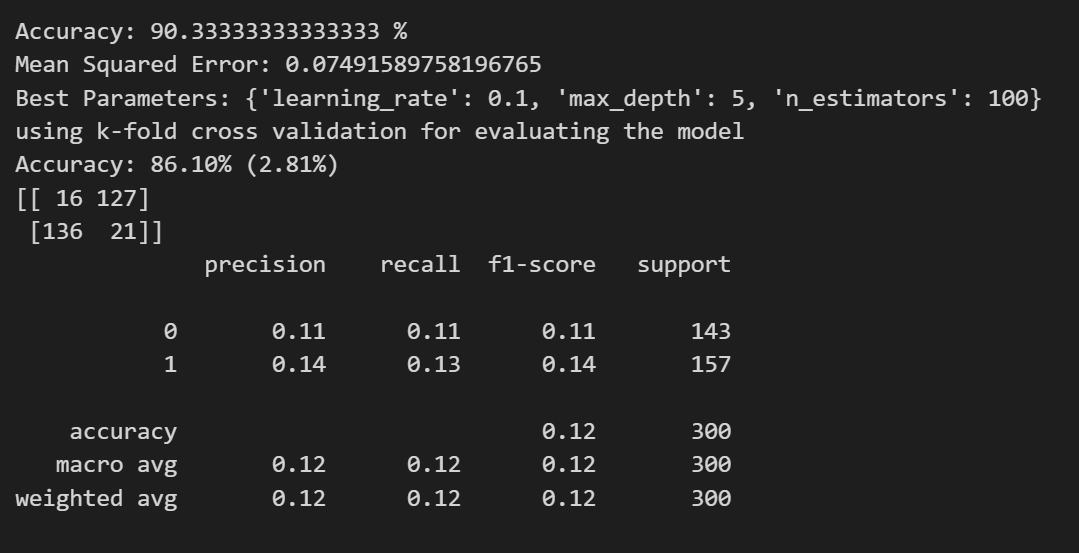
****

**Figure 21**

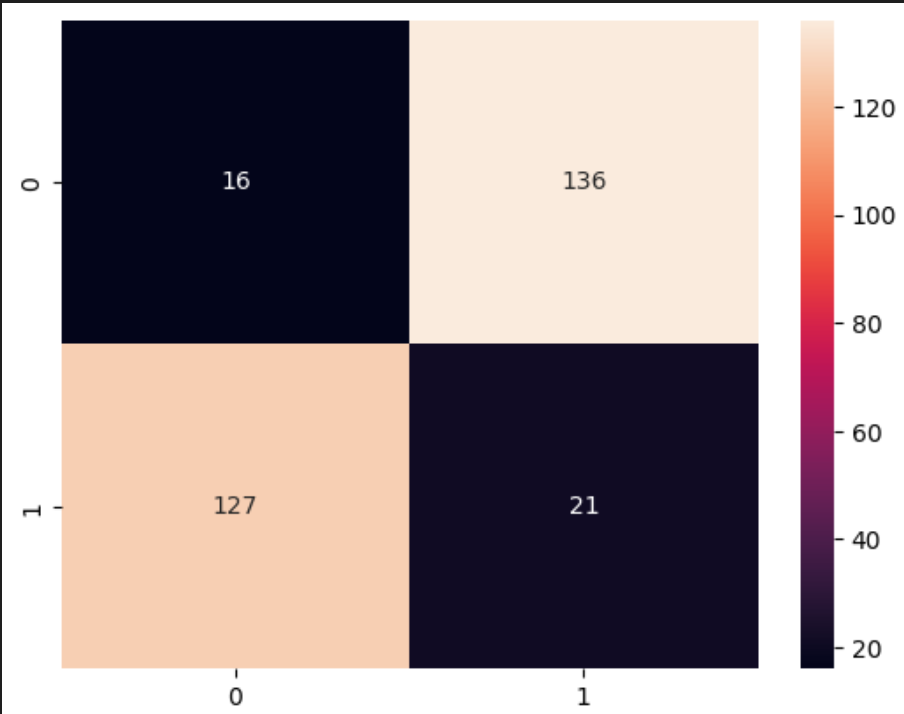
****

**Figure 22**

**Evaluating XGBoost:**

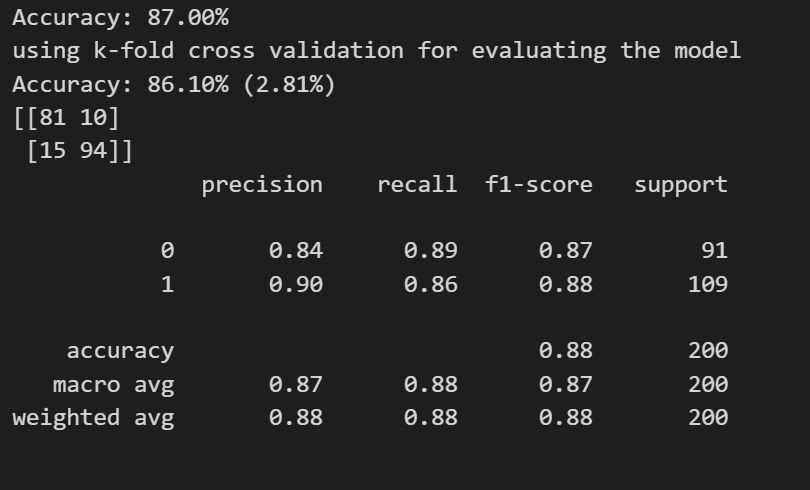
****

**Figure 23**

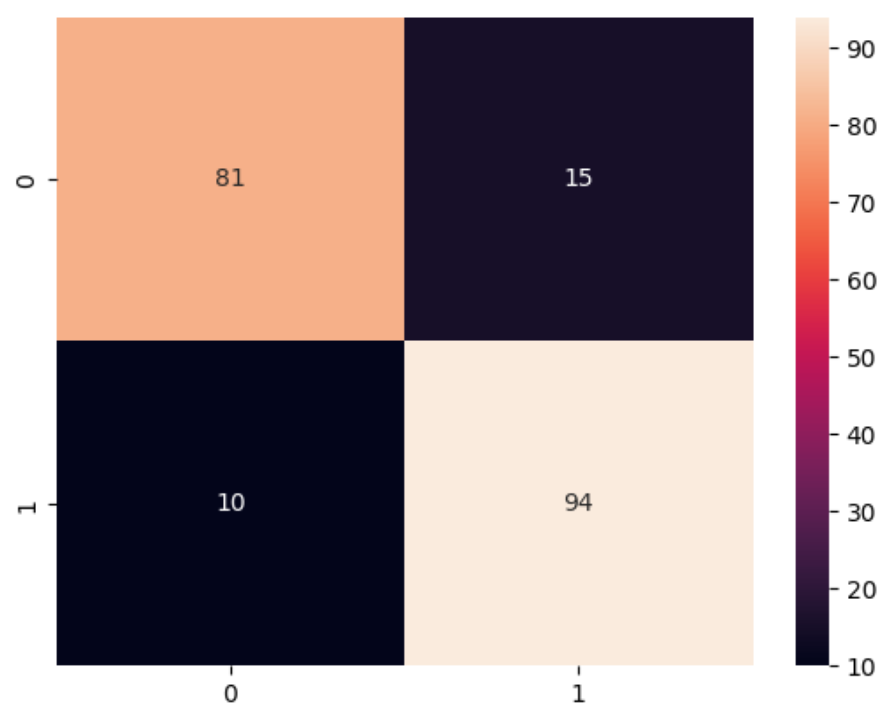
****

**Figure 24**

**Evaluating AdaBoost:**

****

**Figure 25**

****

**Figure 26**

**Evaluating All the Models**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Evaluation measures** | **Accuracy** | **Precision** | **F1score** | **Recall** | **Support** |
| **ML MODEL** |  |  |  |  |  |  |
| **Bagging** |  | 52.60% | 41.33% | 42.33% | 44% | 163.3 |
| **XGBOOST** |  | 90.33% | 12.55% | 12.55% | 12% | 150 |
| **AdaBoost** |  | 87% | 87% | 87.5% | 87.5% | 100 |
|  |  |  |  |  |  |  |

**Table 1**

**Conclusion:**

Accuracy: XGboost > AdaBoost > Bagging

Precision: AdaBoost > Bagging > XGBoost

F1\_Score: AdaBoost > Bagging > XGBoost

Recall: AdaBoost > Bagging > XGBoost

Support: Bagging > XGBoost > AdaBoost

By analyzing all the three ML Models on the basis of accuracy, precision, F1\_score, recall and support, the best model fitted for the wine quality dataset is AdaBoost.

**References:**

* **Multivariate Methods Based Soft Measurement for Wine Quality Evaluation • https://www.hindawi.com/journals/aaa/2014/740754/ Improving Wine Data, Meredith Galante, Jan 28, 2020, Dimensional Insight:**
* **• https://www.dimins.com/blog/2020/01/28/improving-wine-data/ Multivariate Statistical Analysis Applied in Wine Quality Evaluation:**
* **• https://www.researchgate.net/publication/282982301\_Multivariate\_Stat istical\_Analysis\_Applied\_in\_Wine\_Quality\_Evaluation**
* [**https://www.w3schools.com/python/python\_ml\_bagging.asp**](https://www.w3schools.com/python/python_ml_bagging.asp)
* [**https://towardsdatascience.com/predicting-wine-quality-with-several-classification-techniques-179038ea6434**](https://towardsdatascience.com/predicting-wine-quality-with-several-classification-techniques-179038ea6434)
* [**https://www.kaggle.com/datasets/yasserh/wine-quality-dataset**](https://www.kaggle.com/datasets/yasserh/wine-quality-dataset)
* [**https://blog.devgenius.io/red-wine-quality-prediction-using-classification-and-regression-model-f19337821b71**](https://blog.devgenius.io/red-wine-quality-prediction-using-classification-and-regression-model-f19337821b71)
* [**https://towardsdatascience.com/predicting-wine-quality-with-gradient-boosting-machines-a-gmb-tutorial-d950b1542065**](https://towardsdatascience.com/predicting-wine-quality-with-gradient-boosting-machines-a-gmb-tutorial-d950b1542065)
* **https://towardsdatascience.com/predicting-wine-quality-with-several-classification-techniques-179038ea6434**
* [**https://www.geeksforgeeks.org/wine-quality-prediction-machine-learning/**](https://www.geeksforgeeks.org/wine-quality-prediction-machine-learning/)
* **https://www.analyticsvidhya.com/blog/2021/04/wine-quality-prediction-using-machine-learning/**