A blue text on a black background

Description automatically generated

A Report On: -

## Impact of Hyperparameter Tuning on Model Performance

|  |  |
| --- | --- |
| **Submitted By: -**  **Rajan Ghimire**  **C0924991** | **Submitted To: -**  **Victoria Shtern** |
|  |  |

Contents

[Impact of Hyperparameter Tuning on Model Performance 1](#_Toc173105896)

[Abstract 1](#_Toc173105897)

[Introduction: 1](#_Toc173105898)

[Objectives: 1](#_Toc173105899)

[About the Dataset: 1](#_Toc173105900)

[Data Validation: 2](#_Toc173105901)

[Valid Data Types: 2](#_Toc173105902)

[Missing Values: 2](#_Toc173105903)

[Duplicate Values: 3](#_Toc173105904)

[Exploratory Data Analysis: 3](#_Toc173105905)

[Target Distribution: 3](#_Toc173105906)

[Co-relation: 4](#_Toc173105907)

[Data Engineering from Data Mining: 5](#_Toc173105908)

[Modeling: 6](#_Toc173105909)

[DecisionTreeClassifier: 6](#_Toc173105910)

[RandomForestClassifier: 7](#_Toc173105911)

[LogisticRegression 8](#_Toc173105912)

[Models Comparison: 10](#_Toc173105913)

[Conclusion: 10](#_Toc173105914)

## Abstract

We aim to evaluate and compare the performance of three classification algorithms, DecisionTreeClassifier, RandomForestClassifier and LogisticRegression, on the Wine Quality Dataset. This dataset contains physicochemical properties of various Portuguese "Vinho Verde" wines and their quality ratings. The goal is to find the optimal hyperparameters for each algorithm and determine which model provides the best accuracy in predicting wine quality.

## Introduction:

Hyperparameter tuning is essential for optimizing the performance of machine learning models. This process involves selecting the best set of hyperparameters for a model to achieve the highest accuracy and generalization to unseen data. In this assignment, we evaluate and compare the performance of three classification algorithms: Decision Tree, Random Forest, and Logistic Regression. By systematically tuning their hyperparameters, we aim to identify the most accurate model for predicting wine quality based on its physicochemical properties.

## Objectives:

* Use Grid Search or Random Search to find the optimal hyperparameters for each algorithm.
* Train the models on the training set and evaluate their performance on the testing set using accuracy and other relevant metrics.

## About the Dataset:

The dataset used in this study is the Wine Quality Dataset from the UCI Machine Learning Repository. It contains 1,143 samples of red "Vinho Verde" wine, each described by 11 physicochemical properties and a quality rating. The features and their descriptions are : 'fixed\_acidity', 'volatile\_acidity', 'citric\_acid', 'residual\_sugar', 'chlorides, 'free\_sulfur\_dioxide', 'total\_sulfur\_dioxide', 'density', 'ph', 'sulphates', 'alcohol', 'quality'.

## Data Validation:

### Valid Data Types:

A screenshot of a computer program

Description automatically generated

There were no Incorrect data, and all data was correct and was in correct format.

### Missing Values:

A screen shot of a computer code

Description automatically generated

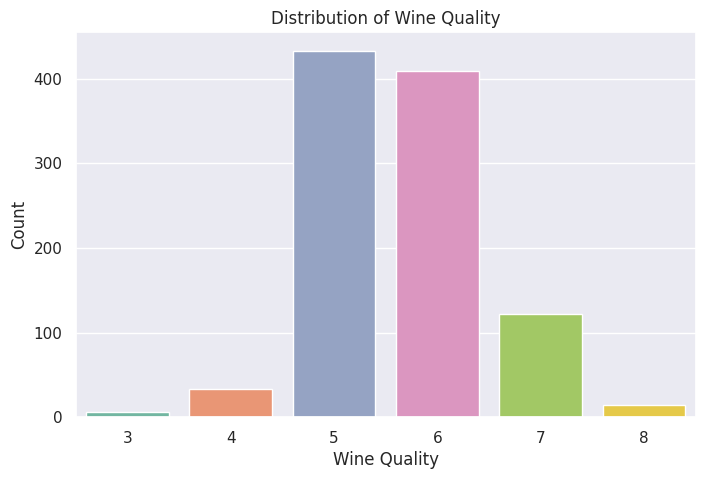
There are no missing values in the dataset.

### Duplicate Values:

There were **125** duplicate values in the dataset. And the total percentage of duplicate values was **10%** and has been removed in the final dataset.

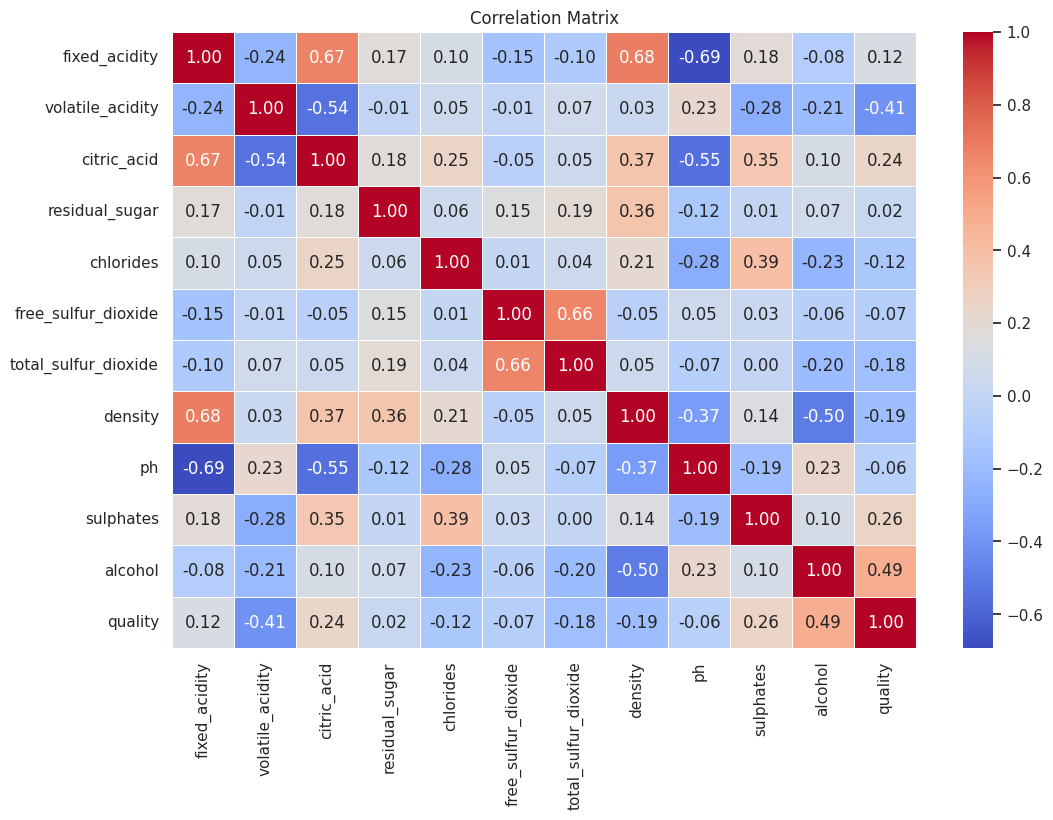
## Exploratory Data Analysis:

### Target Distribution:



The dataset is not well balanced with majority of class belonging to 5 and 6 labels.

### Co-relation:



This correlation matrix shows the relationships between various features in the wines dataset. The most positively correlated features are 'density' and 'fixed\_acidity' (0.68), while 'quality' is highly correlated with 'alcohol' (0.49). The most negatively correlated features are 'density' and 'alcohol' (-0.50), and 'fixed\_acidity' and 'pH' (-0.69). 'Volatile\_acidity' and 'quality' also show a strong negative correlation (-0.41).

## Data Engineering from Data Mining:



The feature engineering code creates a new feature `Total\_sulphur\_Dioxide` by summing `free\_sulfur\_dioxide` and `total\_sulfur\_dioxide`, and an `Acidity` feature by summing `fixed\_acidity`, `volatile\_acidity`, and `citric\_acid`, then drops the original columns. It categorizes `residual\_sugar` into 'low', 'medium', and 'high', and `ph` into 'acidic', 'neutral', and 'basic', followed by one-hot encoding these categorical variables.

Finally, the dataset is split into features (`X`) and target (`y`), and then into training and testing sets using a **75-25** split with `train\_test\_split`, ensuring a robust evaluation of the model on unseen data.

## Hyperparameter Experiments:

### DecisionTreeClassifier:

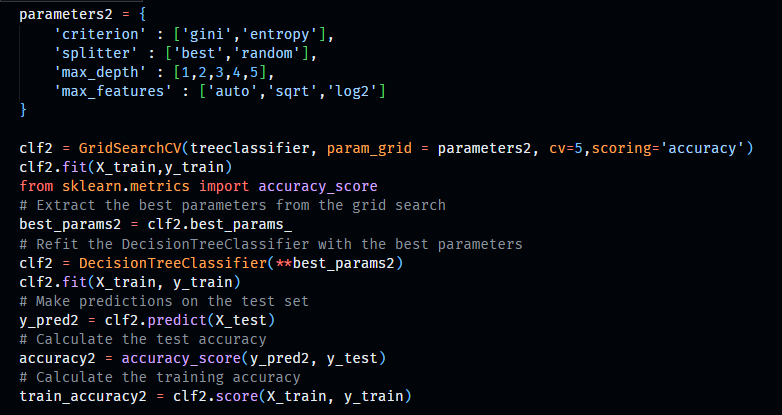
Before the Hyperparameter Tuning:

A computer screen shot of code

Description automatically generated

Here we got Training Accuracy for DecisionTreeClassifier as **100%** and Test Accuracy for DecisionTreeClassifier: as **43.52%**.

After Hyperparameter Tuning:



After Hyperparameter Tuning the best parameters were: **'criterion': 'gini', 'max\_depth': 5, 'max\_features': 'log2', 'splitter': 'best'.** And Training Accuracy for DecisionTreeClassifier was **66.97%** Test Accuracy for was **46.27%.**

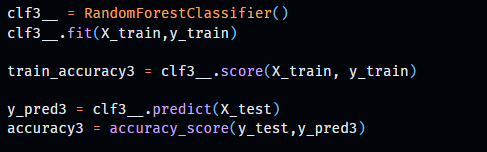
#### Testing on Data:

A computer screen with text and numbers

Description automatically generated

### RandomForestClassifier:

Before the Hyperparameter Tuning:



Here we got Training Accuracy for RandomForestClassifier as **100%** and Test Accuracy for RandomForestClassifier: as **50.1%**.

After Hyperparameter Tuning:

A computer screen with colorful text

Description automatically generated

After Hyperparameter Tuning the best parameters were: **'n\_estimators': 300, 'max\_depth': 6, 'criterion': 'entropy'.** And Training Accuracy for RandomForestClassifier was **74.5%** Test Accuracy for was **50.41%.**

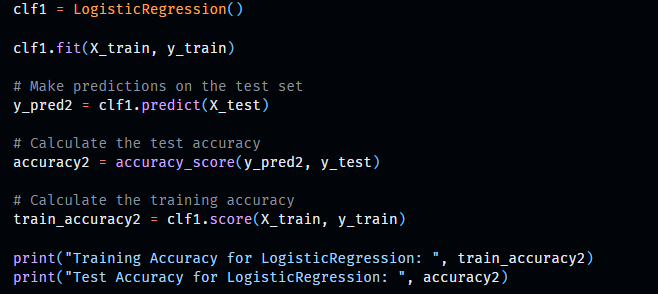
#### Testing on Random Data:

A computer screen with white and blue text

Description automatically generated

### LogisticRegression

Before the Hyperparameter Tuning:



Here we got Training Accuracy for LogisticRegression as **50.12%** and Test Accuracy for LogisticRegression: as **42.3%**.

After Hyperparameter Tuning:

A computer screen with text on it

Description automatically generated

After Hyperparameter Tuning the best parameters were: **'C': 50, 'penalty': 'l2.** And Training Accuracy for LogisticRegression was **51.37%** Test Accuracy for was **43.13%.**

#### Testing on Data:

A computer screen with white text and numbers

Description automatically generated

## Comparison:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Models | Before Hyperparameter | | | After Hyperparameter |  |  | % Test Improvement |
| Accuracies | Train | Test | Overfit/Underfit | Train | Test | Overfit/Underfit |  |
| DecisionTreeClassifier | 100% | 43.52% | Overfit | 66.97% | 46.27% | No | 2.75% |
| RandomForestClassifier | 100% | 50.10% | Overfit | 74.50% | 50.41% | No | 0.31% |
| LogisticRegression | 51.12% | 42.30% | No | 51.37% | 43.13% | No | 0.83% |

## Conclusion:

The study concludes that hyperparameter tuning significantly reduced overfitting in both the DecisionTreeClassifier and RandomForestClassifier, resulting in slight improvements in test accuracy. The DecisionTreeClassifier's test accuracy increased from **43.52%** to **46.27%**, while the RandomForestClassifier's test accuracy improved marginally from **50.10%** to **50.41%**. LogisticRegression showed a minimal improvement in test accuracy from **42.30%** to **43.13%**, indicating a more stable performance with no overfitting both before and after tuning. Overall, hyperparameter tuning led to better model generalization and slight test performance enhancements.