VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**MACHINE LEARNING MIDTERM**

*Instructor*: **PhD.Le Anh Cuong**

*Implementer*: **Le Hai Dang- 522H0112**

**Chau Bao Nhan - 522H0093**

*Group:*  **11**

VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**MACHINE LEARNING MIDTERM**

*Instructor*: **PhD.Le Anh Cuong**

*Implementer*: **Le Hai Dang- 522H0112**

**Chau Bao Nhan - 522H0093**

*Group:*  **11**

**Declaration of Originality**

**TON DUC THANG UNIVERSITY**

I hereby declare that this thesis is the result of our own work and has been generated under the supervision of PhD. Le Anh Cuong. The research, findings, and conclusions presented in this thesis are original and have not been previously published in any form. All data presented in tables and figures for analysis, evaluation, and assessment were collected by the author(s) from various sources, which are clearly cited in the references.

Furthermore, any comments, evaluations, or data from other authors or organizations used in this thesis are properly cited and referenced.

I take full responsibility for any academic misconduct found in this thesis. Ton Duc Thang University is not liable for any copyright infringements caused by me during the course of this work (if any).

**SUMMARY**

**Middle Exam Project: Analysis of the Wine Quality Dataset**

This report leverages the Wine Quality Dataset to fulfill the main project requirements, which include:

* **Solving the Classification Problem**: Predicting wine quality based on a variety of chemical characteristics, treating the quality scores as categorical classes.
* **Solving the Regression Problem**: Developing a regression model to analyze how specific chemical attributes correlate with wine quality ratings, providing insights into key factors affecting wine quality.
* **Understanding and Addressing Overfitting**: Applying techniques, such as regularization and cross-validation, to mitigate overfitting, improving model generalization and accuracy on unseen test data.
* **Applying Feature Selection through Correlation Analysis**: Conducting correlation analysis to identify significant chemical properties that most impact wine quality, optimizing model performance by focusing on these key features.

By implementing various machine learning models, this report offers targeted insights and detailed evaluations of model performance based on the dataset’s predictive outcomes.

**CATEGORY**

[Declaration of Originality 1](#_Toc29822)

[SUMMARY 2](#_Toc25096)

[CATEGORY 1](#_Toc31612)

[List of image 2](#_Toc11997)

[QUESTION 1 – CLASSIFICATION & REGRESSION 3](#_Toc25521)

*[Requirements](#_Toc11712)* [3](#_Toc11712)

*[1.Problem solving approach](#_Toc21919)* [3](#_Toc21919)

*[2. Explore Data](#_Toc19663)* [3](#_Toc19663)

[I.Data visualization: 5](#_Toc22536)

[II.Data preprocessing: 6](#_Toc27766)

[III.Training and evaluation sets 6](#_Toc13196)

[IIII.Build a Machine Learning Model 6](#_Toc7444)

*[3. Classification](#_Toc80)* [7](#_Toc80)

[Observations: 18](#_Toc8850)

[Conclusion: 19](#_Toc5536)

*[4. Regression](#_Toc13797)* [19](#_Toc13797)

[QUESTION 2 – OVERFITTING 35](#_Toc5107)

*[Requirements:](#_Toc31651)* [35](#_Toc31651)

[1.](#_Toc17336) *[Overfitting definition](#_Toc17336)* [35](#_Toc17336)

*[2.Method on solving overfitting](#_Toc8190)* [36](#_Toc8190)

[QUESTION 3 – CORRELATION ANALYSIS 43](#_Toc6093)

*[Requirements:](#_Toc19435)* [43](#_Toc19435)

[1.](#_Toc22688) *[Correlation matrix definition.](#_Toc22688)* [43](#_Toc22688)

[2. Apply into exercise 1 45](#_Toc1092)

[REFERENCES 47](#_Toc25916)

**List of image**

[Image 1 .Data visualization 5](#_Toc31035)

[Image 2 .Formula Accuracy 11](#_Toc19715)

[Image 3 .Formula Precision and Recall 11](#_Toc18182)

[Image 4 .Formula F1 Score 11](#_Toc27550)

[Image 5 . Libraries use for project 12](#_Toc2518)

[Image 6 .All columns in data 13](#_Toc28683)

[Image 7 .Decribe data 13](#_Toc16048)

[Image 8 . Check null value 14](#_Toc4626)

[Image 9 .Correlation features 15](#_Toc17227)

[Image 10 .Compare models 18](#_Toc16842)

[Image 11 .Cross-Validation Accuracy 19](#_Toc21015)

[Image 12 .Formula MSE 22](#_Toc9149)

[Image 13 .Formula RMSE 22](#_Toc32639)

[Image 14 .Formula MAE 22](#_Toc6215)

[Image 15 .Formula R2 23](#_Toc7275)

[Image 16 . Libraries use for project 25](#_Toc21055)

[Image 17 .All columns in data 26](#_Toc8714)

[Image 18 .Decribe data 26](#_Toc16601)

[Image 19 . Check null value 27](#_Toc2766)

[Image 20 .MSE 31](#_Toc20683)

[Image 21 .RMSE 32](#_Toc15544)

[Image 22 .MAE 33](#_Toc18607)

[Image 23 .R2 34](#_Toc26837)

[Image 24 .Compare before and after solving overfitting except SVR 41](#_Toc17577)

[Image 25 .Compare before and after solving overfitting 42](#_Toc1033)

[Image 26 .New Cross-Validation Accuracy 43](#_Toc5449)

[Image 27 .Correlation matrix 44](#_Toc15627)

[Image 28 .Correlation features 46](#_Toc1459)

**QUESTION 1 – CLASSIFICATION & REGRESSION**

***Requirements***

* Provide a statistical summary of the data using data visualization.
* Preprocess the data: type conversion and data normalization.
* Split the data into training and evaluation sets.
* Perform classification (and regression).
* Evaluate and compare the results of the methods: compare classification methods with each other; compare regression methods with each other.
* Use charts to visually display the comparison.

***1.Problem solving approach***

1. Understand and Explore the Data
2. Data Preprocessing
3. Training and evaluation sets
4. Build a Machine Learning Model
5. Evaluate the Model
6. Compare
7. Draw conclusions

***2. Explore Data***

About the dataset:

* Numerical Attributes:
  + Fixed Acidity: Fixed acidity level of the wine (numeric).
  + Volatile Acidity: Volatile acidity level (numeric).
  + Citric Acid: Citric acid content (numeric).
  + Residual Sugar: Amount of residual sugar (numeric).
  + Chlorides: Chloride content (numeric).
  + Free Sulfur Dioxide: Amount of free sulfur dioxide (numeric).
  + Total Sulfur Dioxide: Total amount of sulfur dioxide (numeric).
  + Density: Density of the wine (numeric).
  + pH: pH level of the wine (numeric).
  + Alcohol: Alcohol content (numeric).
  + Quality: Quality of the wine (numeric), rated from 0 to 10.
* Categorical Attribute:
  + Color: Color of the wine ( categorical), red or white.

**I.Data visualization:**

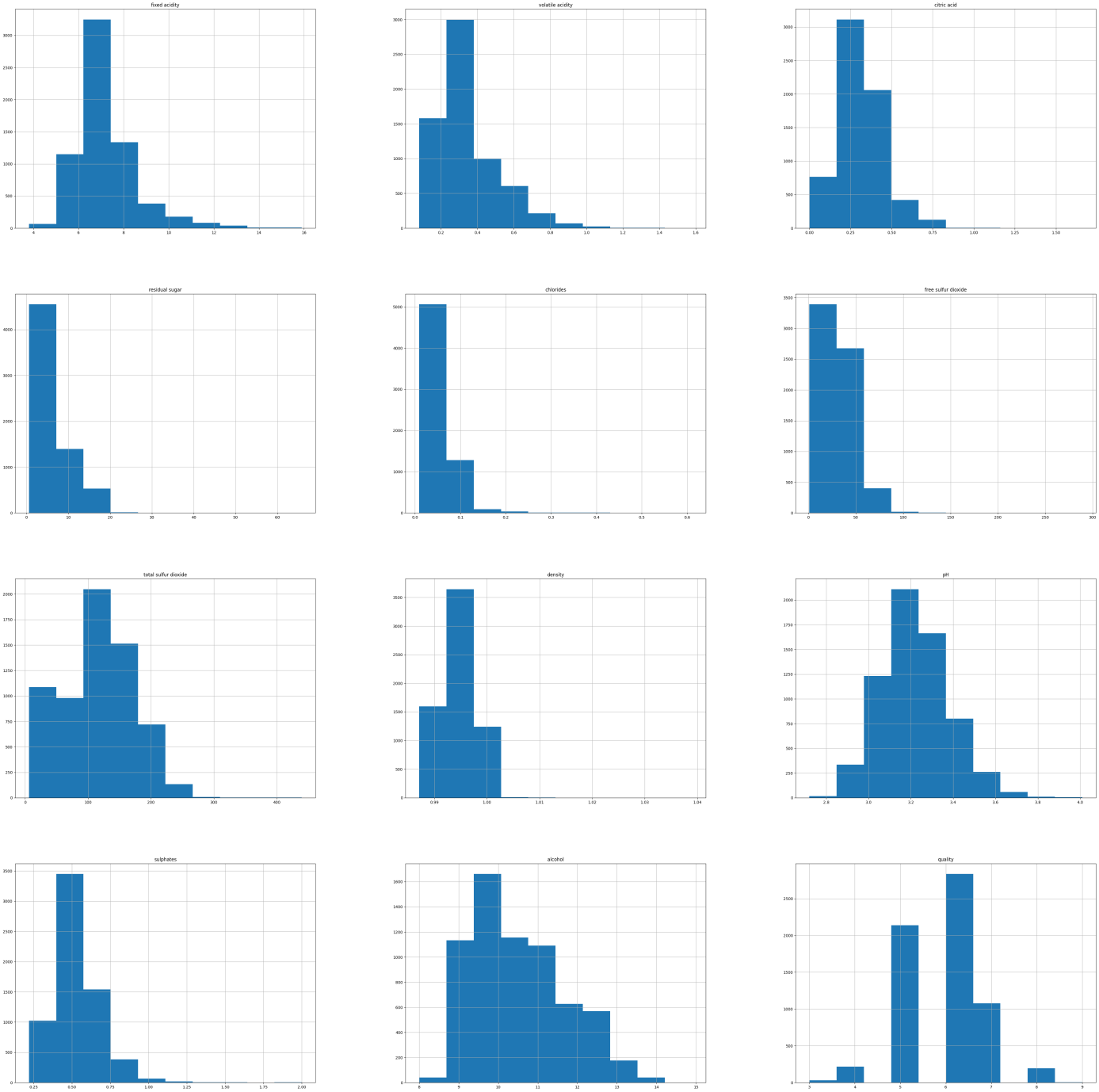


Image 1.Data visualization

**II.Data preprocessing:**

* Check dublicates and drop.
* Handling missing values.
* Covert data type
* Correlation features and drop.
* Perform classification (and regression).
* Data normalization

**III.Training and evaluation sets**

**Feature Selection**:

* 1. Selected all columns in the dataset as feature variables except for the quality column.
  2. **Features Selected**: All columns except quality.

**Target Variable**:

The quality column was set as the target variable, serving as the outcome we aim to predict.

**Data Split**:

* 1. We divided the dataset into training and testing sets with a **test size of 0.2**, meaning 20% of the data is reserved for testing, and 80% is used for training.
  2. **Random State**: Set to 42 to ensure consistent results across runs.

**IIII.Build a Machine Learning Model**

* Classification
  + Logistic regression
  + Decision tree classification
  + K-Nearest Neighbors classification
  + Random forest classification
* Regression
  + Support Vector Regression
  + decision tree regression
  + K-Nearest Neighbors regression
  + Random forest regression

***3. Classification***

In Machine Learning (ML), **classification** is a type of supervised learning where the goal is to predict the category or label of a given data point from a predefined set of possible outcomes. In a classification problem, the algorithm learns from a labeled dataset (where each input is associated with a specific category or label) and then predicts the label for new, unseen data.

*3.1 Classification Algorithms*

* **Logistic Regression**: Logistic regression is a linear model specifically designed for binary classification. It finds a linear boundary (a decision boundary) that separates the two classes. However, unlike simple linear regression, logistic regression doesn’t output values directly; instead, it applies a logistic function (sigmoid) to produce probabilities between 0 and 1.

Key formula:

* **Linear Model Equation**: Logistic regression calculates a weighted sum of the input features.

A black text with a plus and a number

Description automatically generated

* + - z is the weighted sum of the input features
    - w represents weights
    - x the features
    - b the bias term.
  + **Sigmoid (Logistic) Function**: Maps the output to a probability between 0 and 1.

A mathematical equation with numbers

Description automatically generated

* + - where σ(z) is the predicted probability that the input belongs to a certain class
* **Binary Cross-Entropy Loss**: Used as the cost function to optimize weights w\_i​ and bias b by minimizing error.

A black text on a white background

Description automatically generated

A white background with black text

Description automatically generated

* **Decision Trees**: Splits data into branches to decide the class of a data point.

Decision trees create a flowchart-like structure. They start with a question (split) that maximizes separation of the classes. For each split, a metric like **Gini- impurity** or **entropy** is used to evaluate how "pure" each branch is:

* **Gini impurity**: Measures the probability of a wrong classification if randomly chosen.
* **Entropy**: Measures the disorder, aiming for splits that yield branches with minimal disorder.

Key formula:

* + **Gini Impurity**:

A black and white math symbol

Description automatically generated

* + - * pi​ is the proportion of class i in the dataset and C is the number of classes.
  + **Entropy:**

A black and white math symbol

Description automatically generated

* + - pi​ is the proportion of class i in the subset.
* **Random Forest**: Random Forest builds an ensemble (group) of many decision trees, often hundreds, each trained on different random subsets of the data and features (this is called **bagging** and **feature sampling**). Each tree provides a classification, and the forest takes the majority vote as the final prediction.

A black text on a white background

Description automatically generated

* **K-Nearest Neighbors (KNN)**: KNN is a **lazy** learner, meaning it doesn’t create a model during training but instead stores all the data points. When a new point is classified, it calculates the distance to all points in the training data, finds the K nearest points, and takes a "vote" on their classes to assign the new point a class label.

Key formula:

* + **Distance Metric**: KNN classifies a new point by finding the K nearest points in the dataset based on a distance metric, commonly **Euclidean distance**:

A black square with numbers and symbols

Description automatically generated

* + - x and x′ are the feature vectors of two points.
* **Voting Mechanism**: Once the K nearest neighbors are found, the algorithm assigns the new point to the majority class among the neighbors:

A black text on a white background

Description automatically generated

* + - y1​,y2​,…,yK​ are the classes of the K nearest neighbors.

*3.2 Evaluation Metrics*

* **Accuracy**: Proportion of correct predictions.

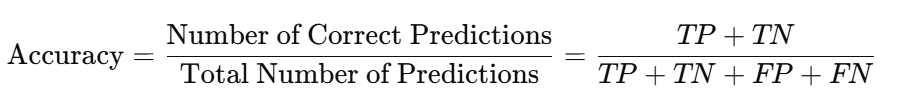


Image 2.Formula Accuracy

* + - TP: True Positives
    - TN: True Negatives
    - FP: False Positives
    - FN: False Positives
* **Precision and Recall**: Precision measures the correctness among positive predictions, while recall measures how many actual positives were correctly identified.

A mathematical equation with black text

Description automatically generatedA math equation with black text

Description automatically generated

Image 3.Formula Precision and Recall

* **F1 Score**: Harmonic mean of precision and recall, balancing the two.

A close-up of a sign

Description automatically generated

Image 4.Formula F1 Score

*3.3 Steps in Classification*

1. **Data Collection:** Gathering labeled data that can be used for training the model.
2. **Data Preprocessing**:
   * Handling Missing Values: Replace or remove missing data.
   * Feature Scaling: Standardize or normalize features to bring them to a similar range (especially for algorithms like KNN).
   * Encoding Categorical Variables: Use techniques like Label Encoding or One-Hot Encoding for categorical features.
3. **Splitting the Dataset**: Split data into training and test sets (commonly 70-30 or 80-20), and often use cross-validation.
4. **Model Selection**: Choose the classification algorithm(s) based on the problem requirements.
5. **Model Training**: Fit the selected model to the training data.
6. **Model Evaluation**: Use metrics to evaluate model performance on unseen test data.

*3.4 Implement*



Image 5. Libraries use for project

These libraries together offer a comprehensive toolkit for data analysis, visualization, preprocessing, and machine learning model development.

- Check unique data for each feature in the dataFrame:

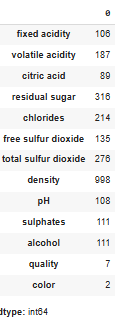


Image 6.All columns in data

**Observation:**

* We can see that all columns do not contain null values.

Quick summarize data:

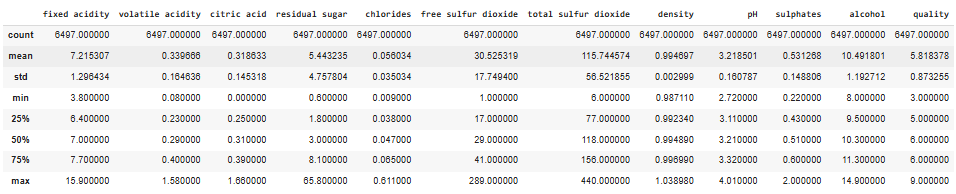


Image 7.Decribe data

Handling duplicate values :

df.drop\_duplicates(inplace= True)

df.duplicated().sum()

Handling missing values:

df.isnull().sum()

print(df.isnull().sum())

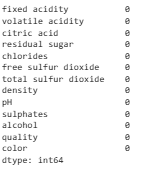


Image 8. Check null value

-This dataset has no null values.

Convert data type:

label\_encoder = LabelEncoder()

df['color'] = label\_encoder.fit\_transform(df['color'])

df.head(20)

Correlation feature and drop:

corr\_matrix = df.corr()

fig, ax = plt.subplots(figsize=(10,10))

sns.heatmap(corr\_matrix,annot=True,linewidths=.5,ax=ax)

plt.show()

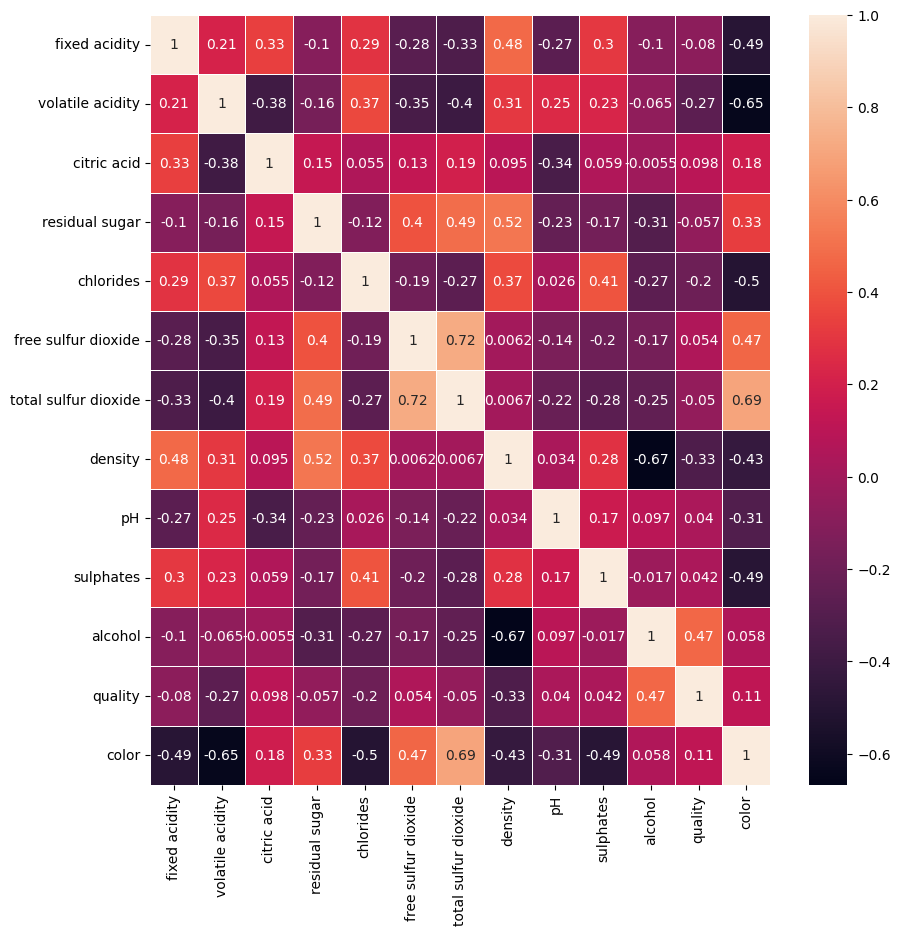


Image 9.Correlation features

-Strong correlation between alcohol,volatile acidity and quality

-Let's drop features which have less than 0.15 correlation

df.drop(['free sulfur dioxide', 'residual sugar', 'pH', 'chlorides', 'fixed acidity'], axis=1, inplace=True)

Normalization data:

scaler = StandardScaler()

numerical\_cols = df.select\_dtypes(include=['float64', 'int64']).columns

numerical\_cols = numerical\_cols[numerical\_cols != 'quality']

df[numerical\_cols] = scaler.fit\_transform(df[numerical\_cols])

Split data into training and evaluation:

x\_train\_cls, x\_test\_cls, y\_train\_cls, y\_test\_cls = model\_selection.train\_test\_split(

x, y, test\_size = 0.2, random\_state = 42

)

x\_train\_cls.shape, x\_test\_cls.shape, y\_train\_cls.shape, y\_test\_cls.shape

x = df.drop(['quality', 'quality\_category'], axis=1)

y = df['quality\_category']

x\_train\_cls, x\_test\_cls, y\_train\_cls, y\_test\_cls = train\_test\_split(x, y, test\_size=0.2, random\_state=42)

y\_test\_cls = y\_test\_cls.astype(str)

y\_pred = y\_pred.astype(str)

-Splits the dataset (x and y) into training (80%) and testing sets (20%) to prepare for model training and evaluation.

*For this report we have chosen Logistic Regression, Random Forest, Decision Tree and K-Nearest Neighbors to solve classification problem*

-Create model for each algorithms:

logistic\_regression\_clf = LogisticRegression()

decision\_tree\_clf = DecisionTreeClassifier()

knn\_clf = KNeighborsClassifier()

random\_forest\_clf = RandomForestClassifier()

-Training models:

logistic\_regression\_clf.fit(x\_train, y\_train)

decision\_tree\_clf.fit(x\_train, y\_train)

knn\_clf.fit(x\_train, y\_train)

random\_forest\_clf.fit(x\_train, y\_train)

-Classification report for each model:

Logistic Regression:

Classification report for logistic regression:

precision recall f1-score support

High 0.57 0.27 0.37 216

Low 0.64 0.62 0.63 390

Medium 0.50 0.64 0.56 458

accuracy 0.55 1064

macro avg 0.57 0.51 0.52 1064

weighted avg 0.56 0.55 0.54 1064

Decision Tree:

Classification report for decision tree:

precision recall f1-score support

High 0.45 0.40 0.42 216

Low 0.57 0.57 0.57 390

Medium 0.50 0.52 0.51 458

accuracy 0.51 1064

macro avg 0.50 0.50 0.50 1064

weighted avg 0.51 0.51 0.51 1064

KNN:

Classification report for KNN:

precision recall f1-score support

High 0.47 0.45 0.46 216

Low 0.61 0.63 0.62 390

Medium 0.53 0.52 0.53 458

accuracy 0.55 1064

macro avg 0.54 0.54 0.54 1064

weighted avg 0.55 0.55 0.55 1064

Random Forest:

Classification report for random forest:

precision recall f1-score support

High 0.61 0.43 0.51 216

Low 0.64 0.66 0.65 390

Medium 0.54 0.60 0.57 458

accuracy 0.59 1064

macro avg 0.60 0.56 0.57 1064

weighted avg 0.59 0.59 0.59 1064

-Compare all models with each other by F1-score:

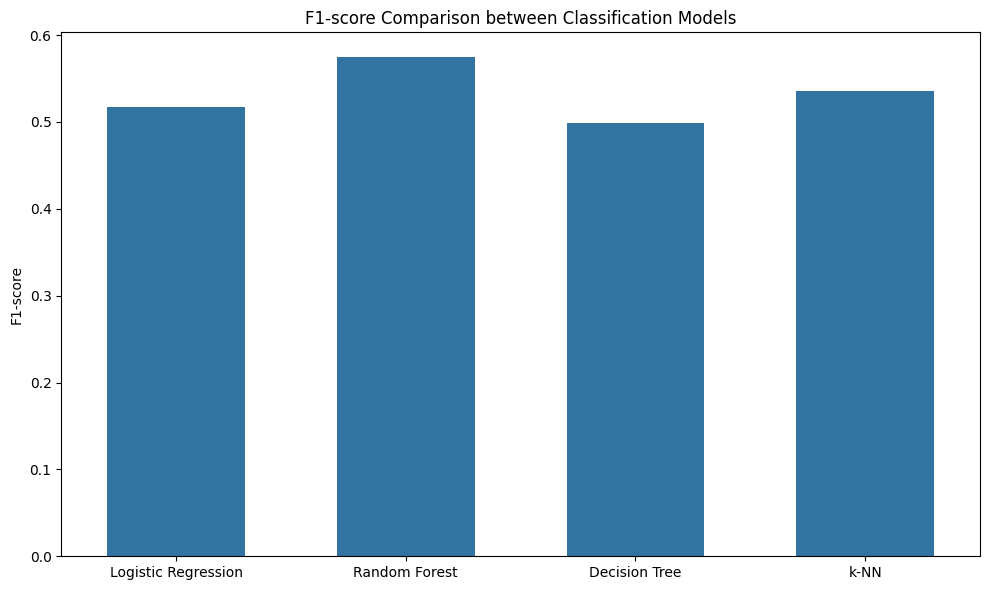


Image 10.Compare models

### Observations:

1. **Random Forest** has the highest F1-score among the models, indicating it may perform better in handling both precision and recall in this specific classification task.
2. **Logistic Regression** and **k-NN** models show similar F1-scores, with LogisticRegression slightly outperforming k-NN.
3. **Decision** **Tree** has the lowest F1-score, suggesting it may not capture the data's patterns as effectively as the other models.

-Compare all models with each other by Accuracy:

A graph with different colored lines

Description automatically generated

Image 11.Cross-Validation Accuracy

### Conclusion:

The **Random Forest and Logistic Regression** models appear to perform the best for this classification problem based on the F1-score and Accuracy.

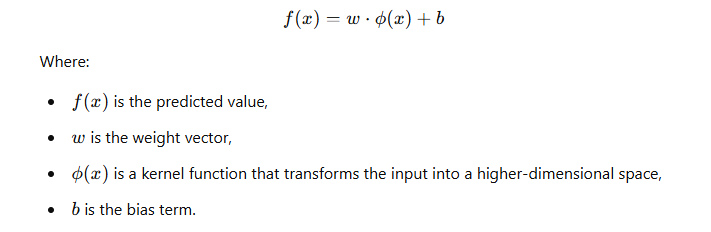
***4. Regression***

Regression refers to a type of supervised learning task focused on predicting continuous, numerical outcomes based on input features. Unlike classification, which categorizes inputs into discrete classes, regression aims to estimate a quantitative relationship between input variables (predictors) and a continuous target variable. Examples of regression tasks include predicting house prices, stock prices, temperature, or even continuous ratings.

*4.1 Regression algorithms:*

* **Support Vector Regression**: Support Vector Regression is a machine learning algorithm used for regression analysis. SVR Model in Machine Learning aims to find a function that approximates the relationship between the input variables and a continuous target variable while minimizing the prediction error.

Formula:



* **Decision Tree Regression**: Decision Tree Regression is a non-linear regression algorithm that builds a model in the form of a tree structure, where each internal node represents a feature, each branch represents a decision rule, and each leaf node represents an outcome (predicted value). The model splits the data based on feature values to make predictions.

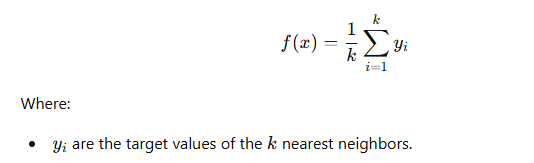
Formula:

There is no specific formula for decision trees, but the prediction can be represented as:



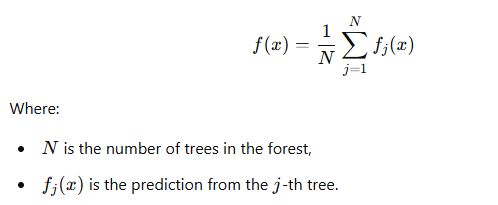
* **K-Nearest Neighbors Regression**: K-Nearest Neighbors Regression is a simple, instance-based learning algorithm that predicts the value of a target variable based on the average values of the k closest training examples in the feature space. The distance can be measured using various metrics, such as Euclidean distance.

Formula:



* **Random Forest Regression**:Random Forest Regression is an ensemble learning method that constructs a multitude of decision trees during training and outputs the average prediction from these trees. It helps to improve prediction accuracy and control overfitting by aggregating the results from multiple trees.

Formula:



*4.2 Evaluation Metrics*

* **Mean squared error**: the average of the squared differences between predicted values and actual values.

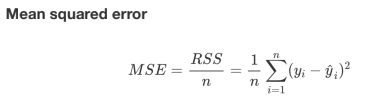
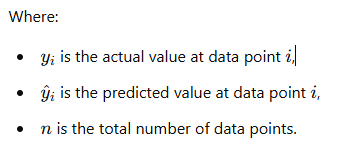


Image 12.Formula MSE



MSE measures the average squared deviation between the predicted and actual values. A smaller MSE indicates a more accurate model.

* **Root Mean squared error**: Proportion of correct predictions.

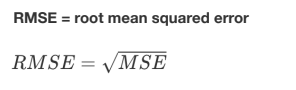


Image 13.Formula RMSE

RMSE provides a more intuitive measure of average deviation because it is in the same unit as the predicted values.

* **Mean absolute error**: Proportion of correct predictions.

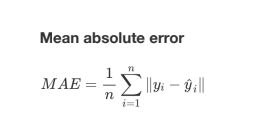
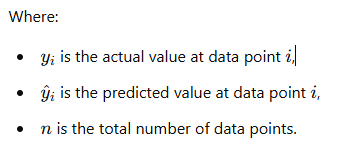


Image 14.Formula MAE



MAE measures the average absolute deviation of predictions from actual values. It is easy to interpret and is less sensitive to outliers compared to MSE.

* **R-squared**:

Definition:

* You can interpret the R2 as the proportion of variation in the dependent variable that is predicted by the statistical model
* It indicates how well the model’s predictions match the observed data.

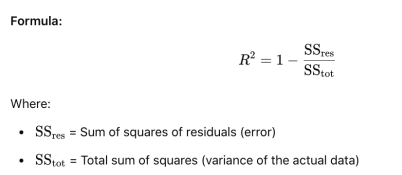
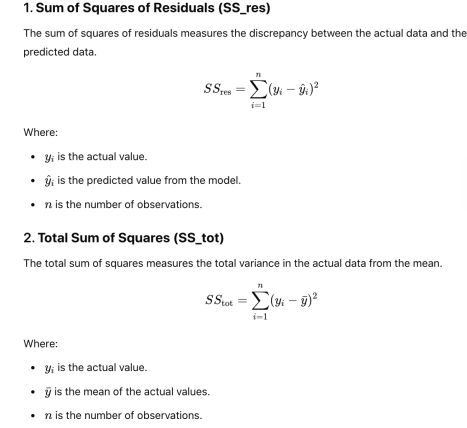
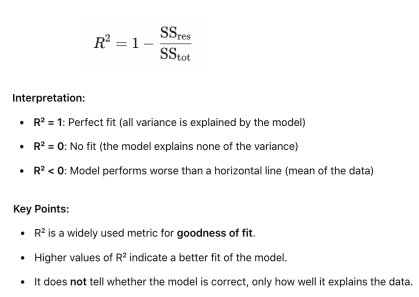


Image 15.Formula R2





*4.3 Steps in Regression*

1. **Data Collection:** Gathering labeled data that can be used for training the model.
2. **Data Preprocessing**:
   1. Handling Missing Values: Replace or remove missing data.
   2. Feature Scaling: Standardize or normalize features to bring them to a similar range (especially for algorithms like KNN).
   3. Encoding Categorical Variables: Use techniques like Label Encoding or One-Hot Encoding for categorical features.
3. **Splitting the Dataset**: Split data into training and test sets (commonly 70-30 or 80-20), and often use cross-validation.
4. **Model Selection**: Choose the classification algorithm(s) based on the problem requirements.
5. **Model Training**: Fit the selected model to the training data.
6. **Model Evaluation**: Use metrics to evaluate model performance on unseen test data.

*4.4 Implement*



Image 16. Libraries use for project

These libraries together offer a comprehensive toolkit for data analysis, visualization, preprocessing, and machine learning model development.

- Check unique data for each feature in the dataFrame:

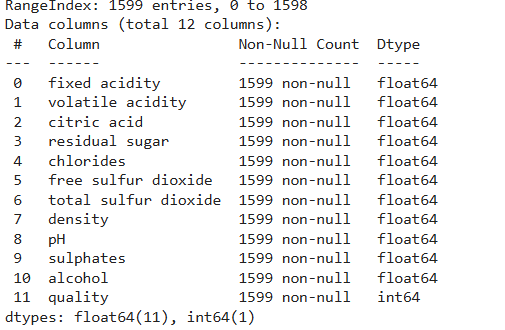


Image 17.All columns in data

**Observation:**

* We can see that all columns do not contain null values.

Quick summarize data:

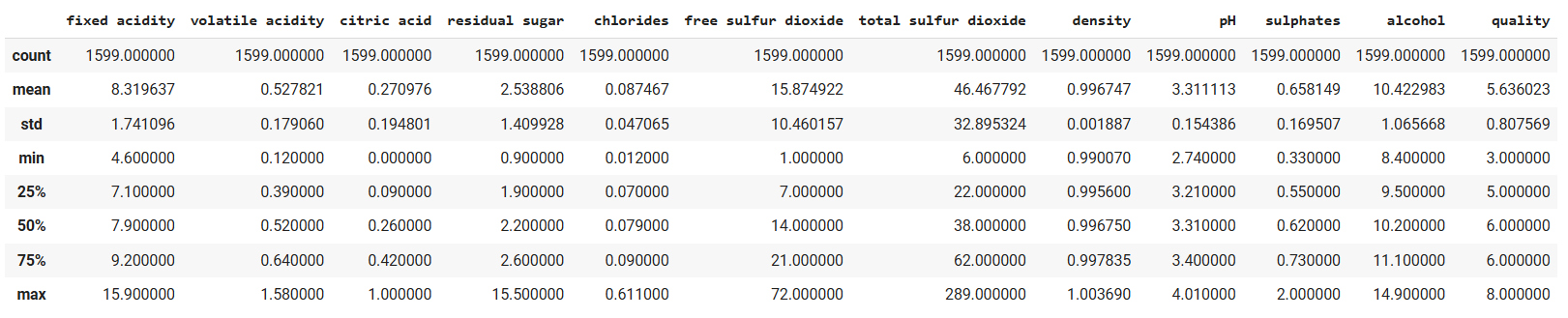


Image 18.Decribe data

Handling duplicate values :

df.drop\_duplicates(inplace= True)

df.duplicated().sum()

Handling missing values:

df.isnull().sum()

print(df.isnull().sum())

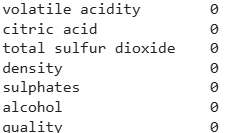


Image 19. Check null value

-This dataset has no null values.

Convert data type:

label\_encoder = LabelEncoder()

df['color'] = label\_encoder.fit\_transform(df['color'])

df.head(20)

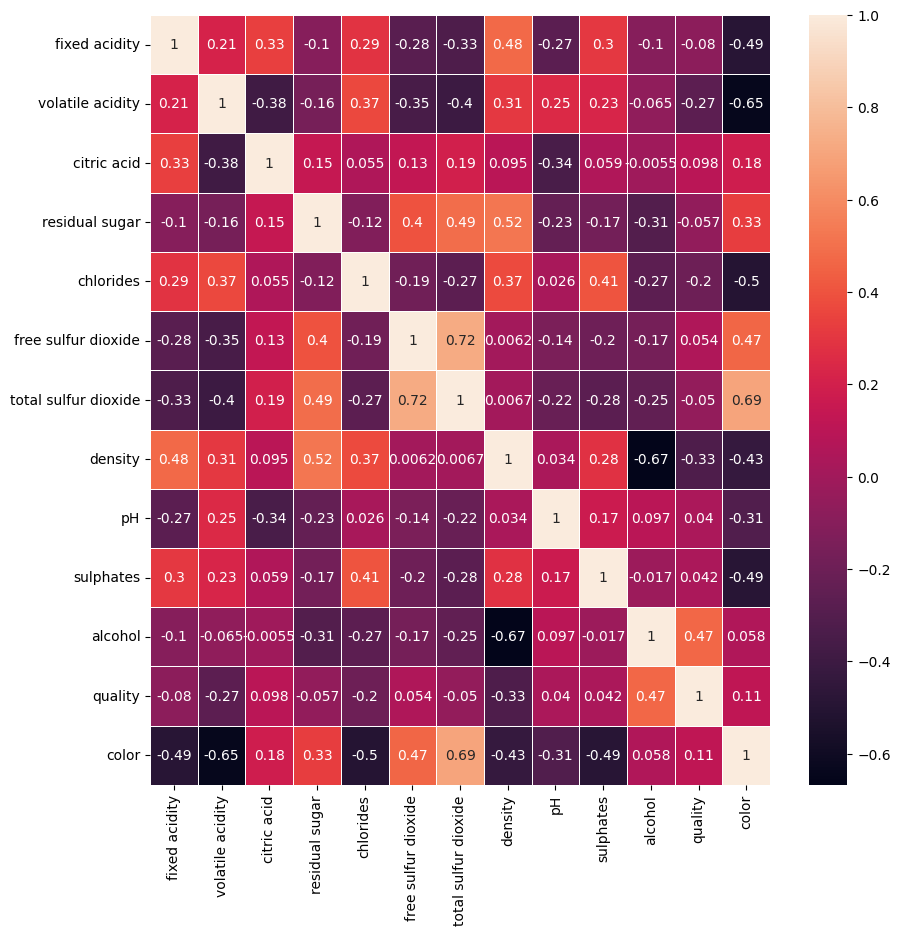
Correlation feature and drop:

corr\_matrix = df.corr()

fig, ax = plt.subplots(figsize=(10,10))

sns.heatmap(corr\_matrix,annot=True,linewidths=.5,ax=ax)

plt.show()



-Strong correlation between alcohol,volatile acidity and quality

-Let's drop features which have less than 0.15 correlation

df.drop(['free sulfur dioxide', 'residual sugar', 'pH', 'chlorides', 'fixed acidity'], axis=1, inplace=True)

Normalization data:

scaler = StandardScaler()

numerical\_cols = df.select\_dtypes(include=['float64', 'int64']).columns

numerical\_cols = numerical\_cols[numerical\_cols != 'quality']

df[numerical\_cols] = scaler.fit\_transform(df[numerical\_cols])

Split data into training and evaluation:

x\_train\_cls, x\_test\_cls, y\_train\_cls, y\_test\_cls = model\_selection.train\_test\_split(

x, y, test\_size = 0.2, random\_state = 42

)

x\_train\_cls.shape, x\_test\_cls.shape, y\_train\_cls.shape, y\_test\_cls.shape

x = df.drop(['quality', 'quality\_category'], axis=1)

y = df['quality\_category']

x\_train\_cls, x\_test\_cls, y\_train\_cls, y\_test\_cls = train\_test\_split(x, y, test\_size=0.2, random\_state=42)

y\_test\_cls = y\_test\_cls.astype(str)

y\_pred = y\_pred.astype(str)

-Splits the dataset (x and y) into training (80%) and testing sets (20%) to prepare for model training and evaluation.

*For this report we have chosen Support Vector Regression, Random Forest, Decision Tree and K-Nearest Neighbors to solve regression problem .*

-Create model for each algorithms:

model = SVR(kernel='rbf', C=3.0, epsilon=0.1)

decision\_tree = DecisionTreeRegressor()

knn = KNeighborsRegressor()

random\_forest = RandomForestRegressor()

-Training models:

model.fit(x\_train, y\_train)

decision\_tree.fit(x\_train, y\_train)

knn.fit(x\_train, y\_train)

random\_forest.fit(x\_train, y\_train)

-Regression evaluation for each model:

Support Vector Regression:

MSE --> 0.5153705796514213

RMSE --> 0.7178931533671437

MAE --> 0.5358938361848281

R2 --> 0.31454988158820374

Decision Tree:

Mean Squared Error (MSE): 1.018796992481203

Root Mean Squared Error (RMSE): 1.0093547406542474

Mean Absolute Error (MAE): 0.6973684210526315

R-squared (R2): -0.3550143269841901

Random Forest:

Mean Squared Error (MSE): 0.5131622180451128

Root Mean Squared Error (RMSE): 0.7163534169982808

Mean Absolute Error (MAE): 0.5408834586466166

R-squared (R2): 0.3174870335800074

K-Nearest Neighbors:

Mean Squared Error (MSE): 0.5639473684210526

Root Mean Squared Error (RMSE): 0.7509642923741798

Mean Absolute Error (MAE): 0.5708646616541354

1. squared (R2): 0.24994206940627917

-Compare all models with each other by MSE,RMSE,MAE,R2:

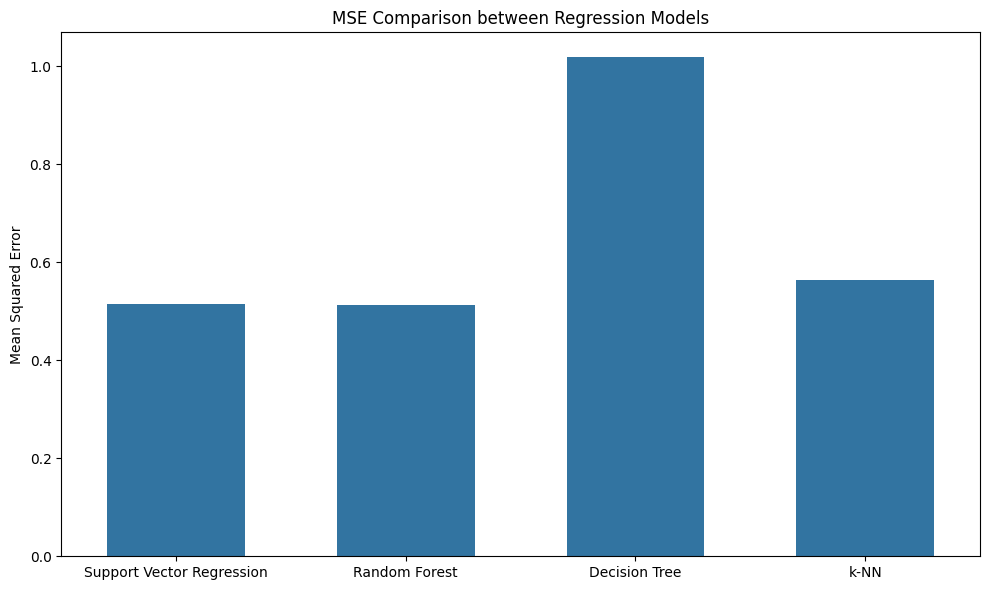


Image 20.MSE

**Observations:**

1. **Random Forest**: Has the lowest MSE, indicating it performs best among the models in terms of prediction accuracy.
2. **Decision Tree**: Exhibits the highest MSE, suggesting it has the least accurate predictions.
3. **Support Vector Regression and k-NN**: Have moderate MSE values, with SVR slightly better than k-NN.

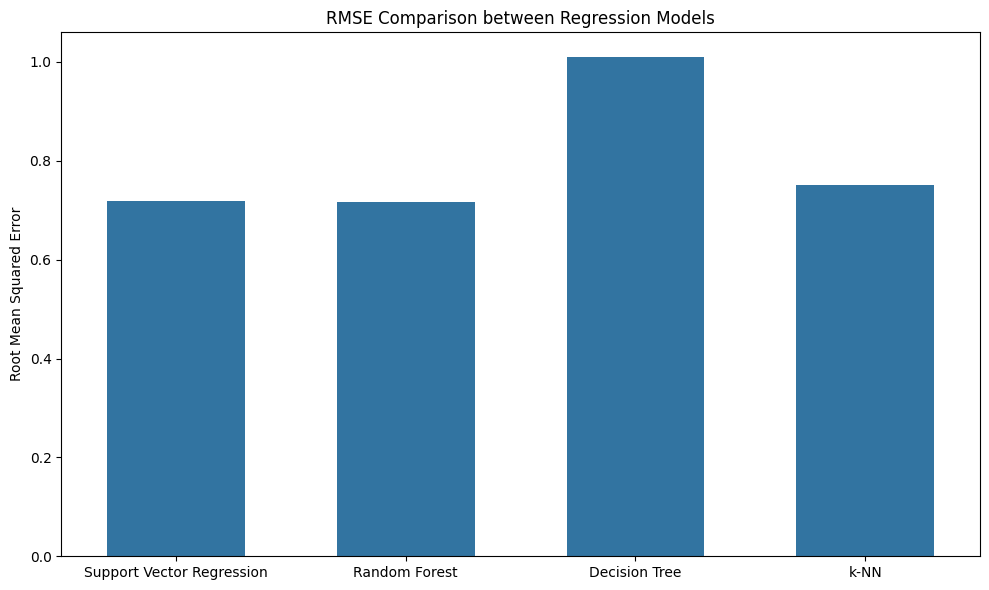


Image 21.RMSE

**Observations:**

1. **Random Forest**: Continues to have the lowest RMSE, indicating strong performance.
2. **Decision Tree**: Has the highest RMSE, consistent with higher error rates.
3. · **Support Vector Regression and k-NN**: Have similar RMSE values, with SVR slightly outperforming k-NN.

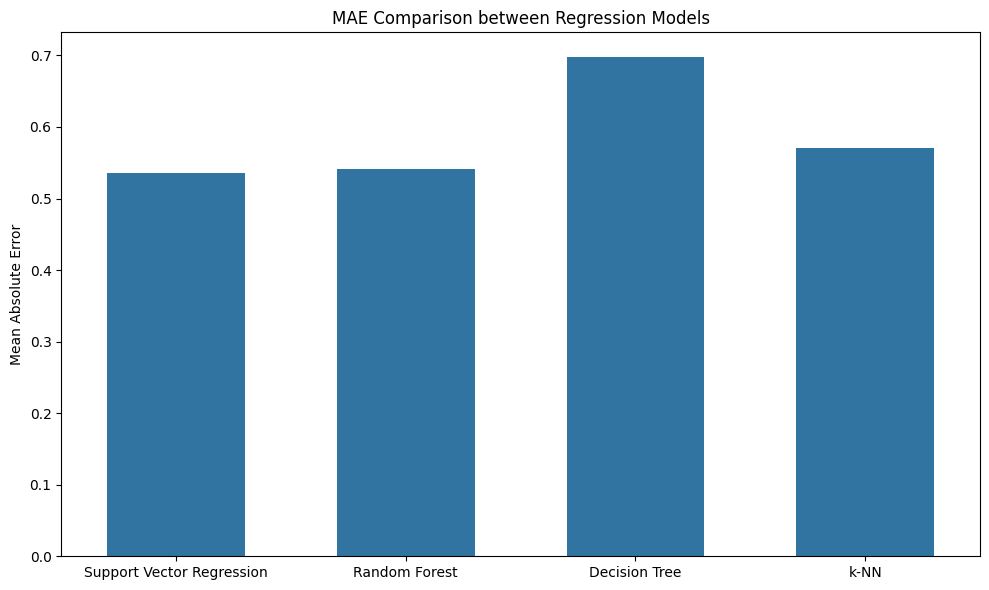


Image 22.MAE

**Observations:**

1. **Random Forest**: Maintains a low MAE, indicating it consistently performs well.
2. **Decision Tree**: Shows the highest MAE, indicating less accurate predictions.
3. **Support Vector Regression and k-NN**: Have moderate MAE values, with SVR performing slightly better.

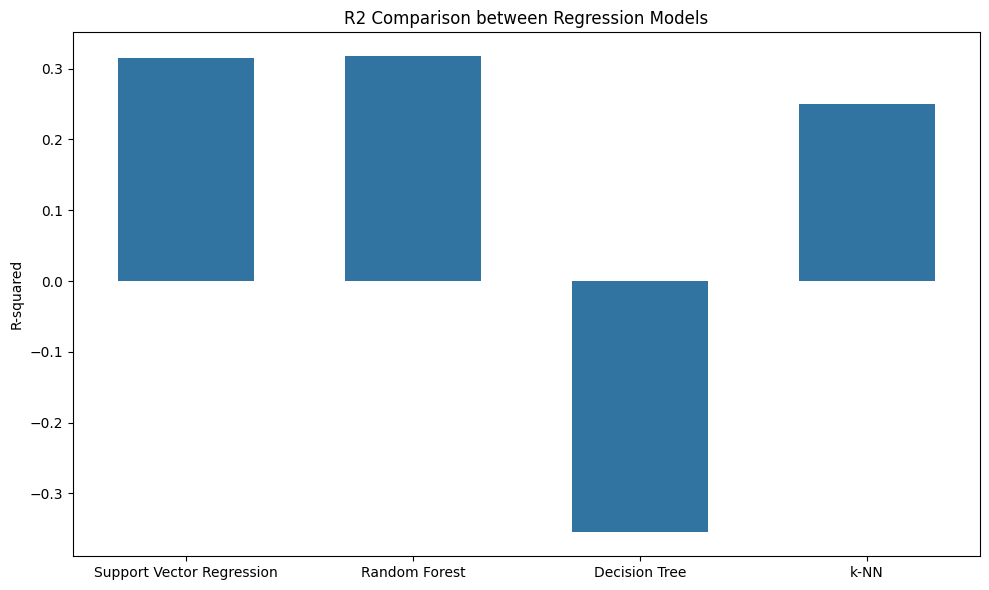


Image 23.R2

**Observations:**

1. **Random Forest**: Has the highest R², indicating it explains the most variance and is the best performing model.
2. **Support Vector Regression**: Also performs well, with a relatively high R².
3. **k-NN**: Shows moderate performance, with a decent R² value.
4. **Decision Tree**: Has a negative R², suggesting it performs worse than a horizontal mean line, indicating poor fit.

**Conclusion:**

The **Random Forest** is the optimal choice for solving regression problems based on RMSE,MSE,MAE,R2.

**QUESTION 2 – OVERFITTING**

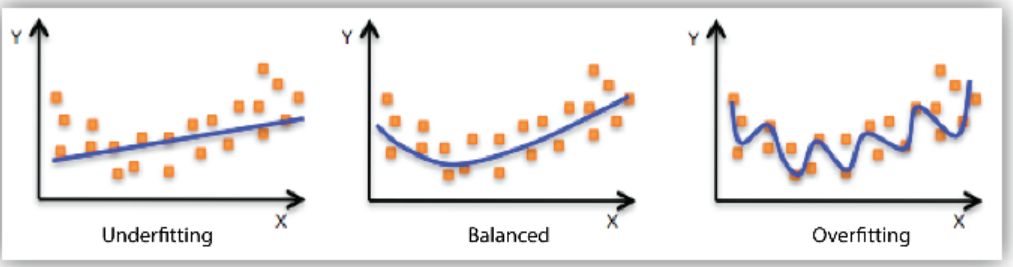
***Requirements:***

* Research and present the concept of overfitting in general, and solutions to address overfitting for the methods in Question 1 (note: you can choose one or more methods to solve overfitting).
* Implement code that includes the solution to overfitting into the methods in Question 1.
* Compare the results with and without overfitting.

1. ***Overfitting definition***

-Overfitting happens when a machine learning model learns not only the patterns in the training data but also the noise or random fluctuations within it. This makes the model perform exceptionally well on the training data but poorly on new, unseen data. Essentially, the model becomes too specialized for the training data and fails to generalize, meaning it doesn’t apply well to other datasets.

-There is also an opposite case of overfitting called underfitting where the model performs too simply on dataset result in the predict might not accuray.



***2.Method on solving overfitting***

1. Simplify the Model:
   * Reduce the model’s complexity by decreasing parameters (e.g., lowering the number of layers or nodes in neural networks, reducing tree depth in decision trees).
   * Choose a simpler model if the current model is too complex for the data.
2. Regularization:
   * Add L1 (Lasso) or L2 (Ridge) regularization to penalize large weights in the model, encouraging simplicity.
   * In neural networks, use Dropout to randomly deactivate neurons during training, which forces the model to generalize better.
3. Use More Training Data:
   * Collect more data if possible to give the model a broader base to learn from, reducing the chance it memorizes only the training data's specifics.
   * If collecting new data isn’t feasible, try data augmentation (for image or text data) to create new variations of the existing data.
4. Cross-Validation:
   * Use k-fold cross-validation to assess model performance more reliably. This technique helps ensure the model generalizes well across various splits of the data, revealing early signs of overfitting.
5. Early Stopping:
   * When training a model, especially in neural networks, monitor its performance on a validation set. Stop training when the validation performance starts to degrade, indicating overfitting.
6. Feature Selection:
   * Use methods like correlation analysis or recursive feature elimination to select only the most relevant features, reducing noise and focusing on key patterns.

***3.Problem solving approach***

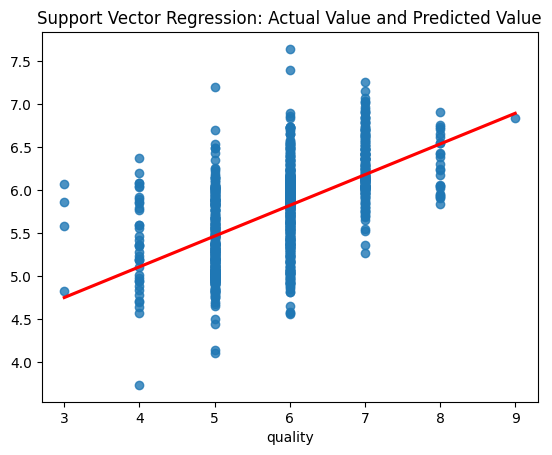
* Identify problem (overfit or underfit)
* Determine solving method for each case
* Apply to question 1 models
* Compare model before and after
* Conclusion

***4.Solving problem***

*4.1 Identify problem*

Observation on result in question 1:

Make a quick view in SVR model as example:



It appears that the model might be underfitting. This can be inferred because the predicted line does not capture the variability in the data points. Many points are scattered around the line, especially at each quality level (e.g., 5, 6, and 7), without the model accurately predicting these specific points.

*4.2 Determine solving method for each model and apply.*

To solve underfitting for these models are choosing hyperparameter tuning techniques, specifically aimed at **increasing model complexity** to reduce underfitting. Here are the specific names or types of each method:

1. **Support Vector Regressor (SVR)**:
   * **Increasing C parameter**: Often referred to as **penalty tuning** or **margin adjustment**.
   * **Reducing epsilon**: This is called **epsilon-tuning** and specifically modifies the margin of error within the support vector machine regression.

svr\_new = SVR(kernel='rbf', C=5, epsilon=0.1)

#Increase C to 5 and reduce epsilon to 0.1

1. **Decision Tree Regressor / Classifier**:
   * **Increasing max\_depth**: Known as **depth tuning** or **increasing tree depth**, allowing the tree to capture more details.
   * **Decreasing min\_samples\_split**: Called **node splitting tuning**, which controls the minimal conditions required for further splits.

decision\_tree\_new = DecisionTreeRegressor(max\_depth=5,min\_samples\_split=2)

#adjust max\_depth to 5 and min\_samples\_split to 2

1. **K-Nearest Neighbors (KNN)**:
   * **Decreasing n\_neighbors**: Known as **neighbor tuning** or **reducing neighborhood size** to make the model more sensitive to local variations.

knn\_new = KNeighborsRegressor(n\_neighbors=10)

#set n\_neighbors to 10

1. **Random Forest Regressor / Classifier**:
   * **Increasing max\_depth**: Similar to depth tuning in decision trees, sometimes called **deeper forest tuning**.
   * **Increasing n\_estimators**: Referred to as **ensemble size tuning**, where a larger ensemble increases the complexity and overall capacity of the forest.

random\_forest\_new = RandomForestRegressor(n\_estimators=500, max\_depth=15,max\_features='sqrt')

#set n\_estimators to 500 and max\_depth to 15

1. **Logistic Regression**:
   * **Increasing C parameter**: Called **regularization tuning**; lowering regularization reduces penalty on complexity, allowing a closer fit to the data.
   * **Using polynomial or interaction terms**: Known as **feature engineering** or **polynomial expansion**, allowing the model to capture non-linear patterns by adding derived features.

logistic\_regression\_clf\_new = LogisticRegression(C=10,max\_iter=1000, solver='saga')

#set C to 10 and max\_iter to 1000

*4.3 Compare model before and after*

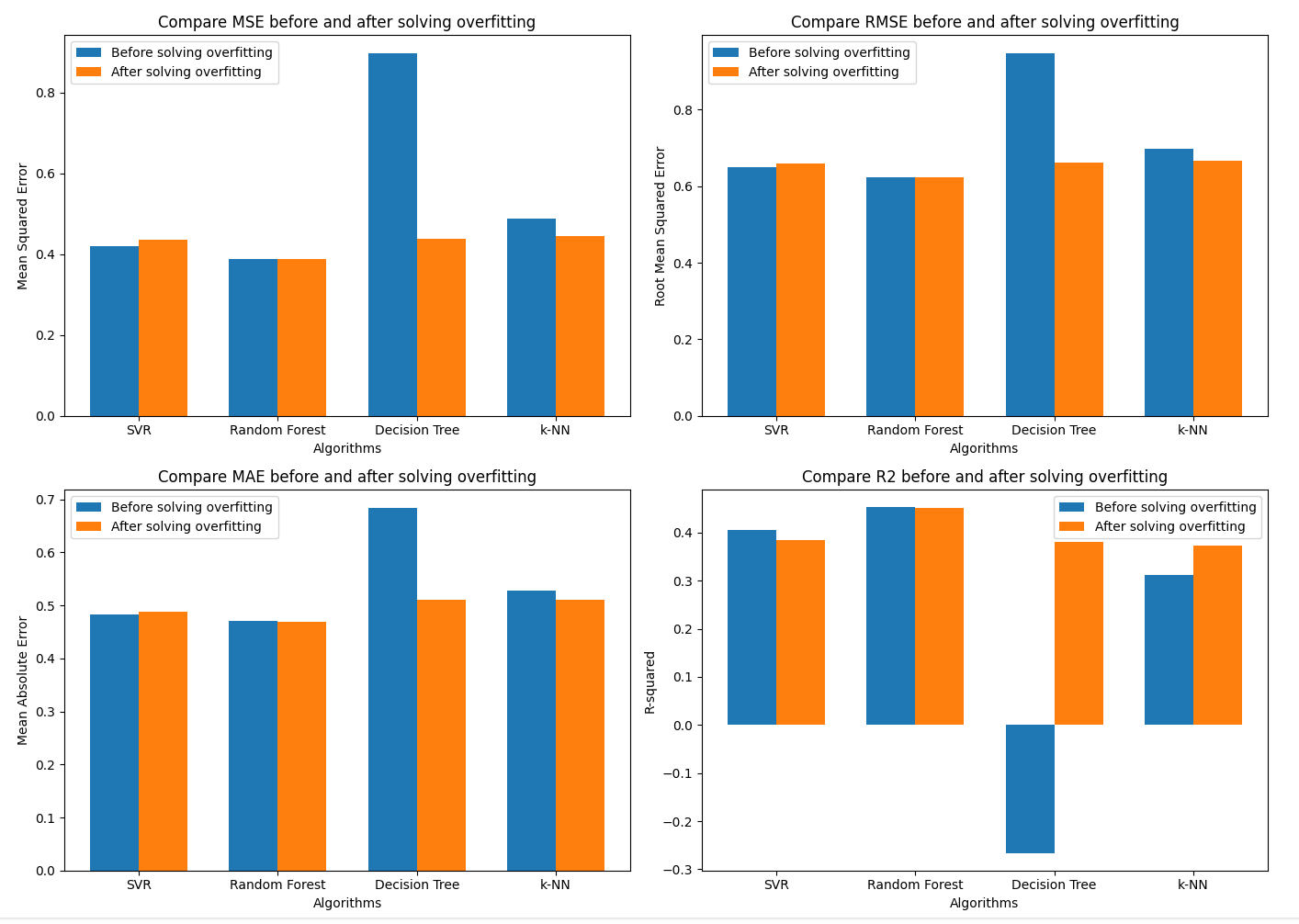
**

Image 24.Compare before and after solving overfitting except SVR

-Every model seems to improve slightly better but for the SVR this could be due to the problem that this model is facing overfitting instead.Let try lower the C parameter to see if it fix this problem:

svr\_new = SVR(kernel='rbf', C=1, epsilon=0.1)

#Increase C to 1

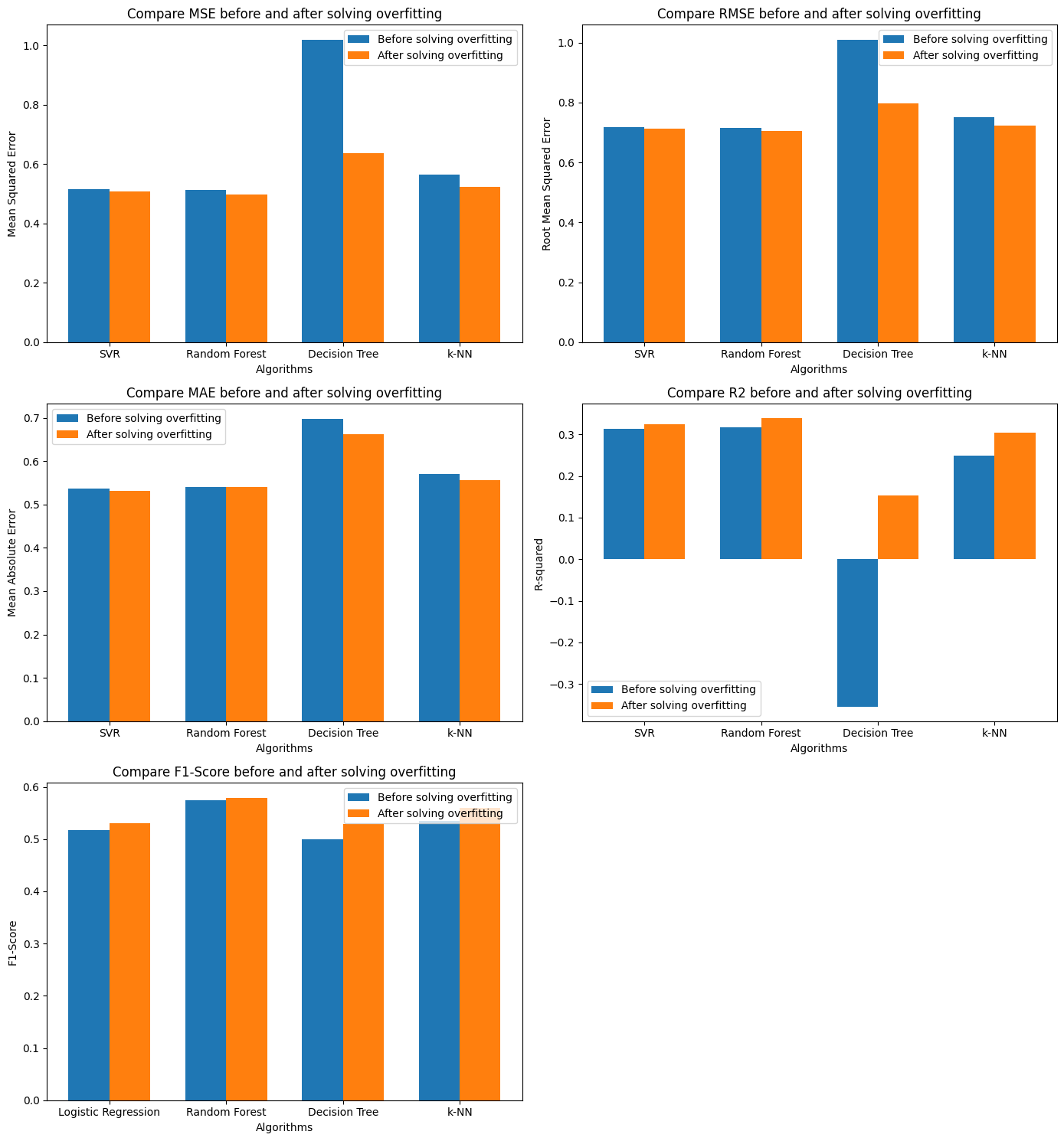


Image 25.Compare before and after solving overfitting

-This time the SVR performed better than the previous version.

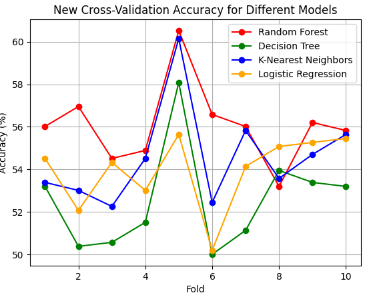


Image 26.New Cross-Validation Accuracy

-The performance for classification has also increased 1-6%. The highest growth rate is KNN around 6% (54% -> 60%)

**QUESTION 3 – CORRELATION ANALYSIS**

***Requirements:***

* Research and present feature selection using correlation analysis. Apply this to the problem for the methods in Question 1.

1. ***Correlation matrix definition.***

A correlation matrix is a statistical technique used to evaluate the relationship between two variables in a data set. The matrix is a table in which every cell contains a correlation coefficient, where 1 is considered a strong relationship between variables, 0 a neutral relationship and -1 a not strong relationship. It’s most commonly used in building regression models.

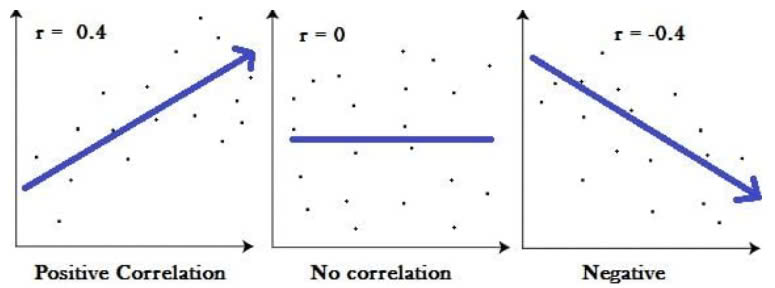


Image 27.Correlation matrix

Using a correlation matrix, we can evaluate the relationship between two variables:

If the relationship is 1, then the relationship is strong.

If the relationship is 0, then it means the relationship is neutral.

If the relationship is -1, then it means the relationship is negative or not strong.

The correlation matrix is a statistical technique that gives you the values between -1 to 1 which you can determine the relationship between variables.

Correlation Matrix Applications

We can summarize a large amount of data in which the goal is to identify patterns. In our example above, the observable pattern is that all the variables highly correlate with each other.

To input into other analyses. For example, people commonly use the correlation matrix as inputs for exploratory factor analysis, confirmatory factor analysis, structural equation models, and linear regression when excluding missing values pairwise.

As a diagnostic when checking other analyses For example, with linear regression, a high number of correlations suggests that the linear regression estimates will be unreliable.

1. **Apply into exercise 1**

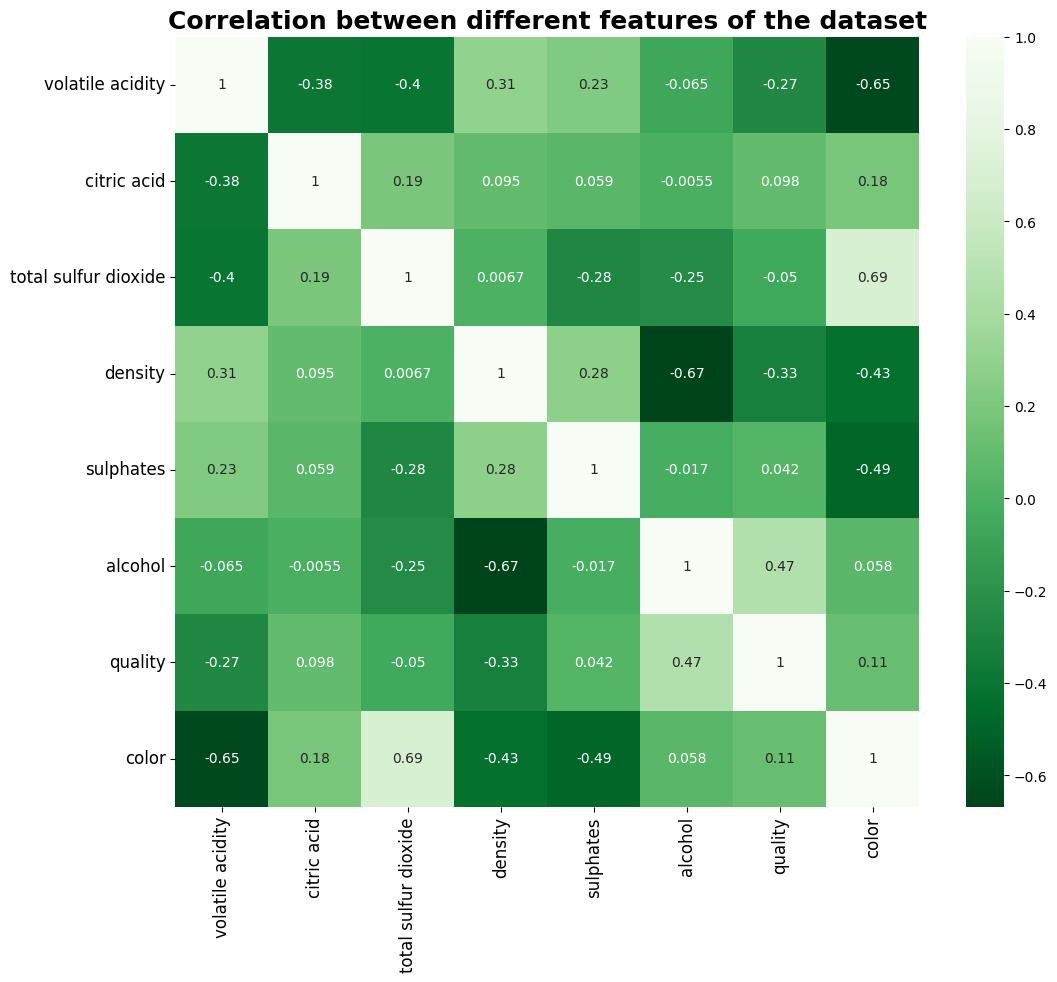
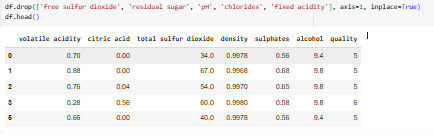


Image 28.Correlation features

**Observation:**

* Strong correlation between alcohol,volatile acidity and quality
* Let's drop features which have less than 0.15 correlation.

**Implement:**



**REFERENCES**

1. Ahmed Sho3ib. (n.d.). Adult income with logistic regression and KNN. Retrieved from [Kaggle](https://www.kaggle.com/code/ahmedsho3ib/adult-income-with-logistic-regression-and-knn" \t "_new): Preprocessing and train data.
2. Built In. (n.d.). Correlation Matrix. Retrieved from [Builtin.com](https://builtin.com/data-science/correlation-matrix" \t "_new).
3. Lecture 4. (n.d.). Evaluation: Regression and Classification.
4. Mulye, Aditi. (n.d.). Adult Income Dataset From Scratch. Retrieved from [Kaggle](https://www.kaggle.com/code/aditimulye/adult-income-dataset-from-scratch" \t "_new): Correlation analysis.