PRECISE COFFEE QUALITY PREDICTION

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

MODUMPURAM RUTHVIKA	**21UK1A0514**
SATIKA CHAITHANYA	**21UK1A0506**
PUCHAKAYALA KEERTHANA	**21UK1A0511**
GATLA SHIVAKUMAR	**22UK5A0506**

Under the guidance of

Mr. P. VAMSHI KRISHNA

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) –

506005

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE(WARANGAL)

CERTIFICATE OF COMPLETION

This is to certify that the Industry oriented mini project entitled **"COFFEE QUALITY PREDICTION"** is being submitted by **MODUMPURAM RUTHVIKA (21UK1A0514), SATIKA CHAITHANYA (21UK1A0506), PUCHAKAYALA KEERTHANA (21UK1A0511), GATLA SHIVA KUMAR(22UK5A0506)** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science& Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2024-2025

Project Guide

Mr. P .Vamshi Krishna

(Assistant Professor)

HOD

Dr. R. NaveenKumar

(Professor)

External

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. SYED MUSTHAK AHAMED,** Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this Industry Oriented mini project.

We extend our heartfelt thanks to **Dr. R. NAVEEN KUMAR**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the Industry Oriented mini project.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the Industry Oriented mini project and for their support in completing the Industry Oriented mini project.

We express heartfelt thanks to the guide, **P. VAMSHI KRISHNA,** Assistant professor, Department of CSE for her constant support and giving necessary guidance for completion of this Industry Oriented mini project.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the Project.

MODUMPURAM RUTHVIKA	**21UK1A0514**
SATIKA CHAITHANYA	**21UK1A0506**
PUCHAKAYALA KEERTHANA	**21UK1A0511**
GATLA SHIVAKUMAR	**22UK5A0506**

ABSTRACT

Nowadays, coffee beans are produced mainly in Brazil, Vietnam, Colombia, Indonesia, Ethiopia, Honduras, India, Peru, and Uganda. As the coffee industry plays a significant part in national economies, these countries are expected to further develop socially and economically through the coffee trade. Thus, research to improve coffee quality must continue. This study introduces the machine learning model, random forest, to predict coffee quality. Consequently, the study found an F1 score of 61.7%. The machine learning model will support coffee producing countries in maintaining competitiveness in the market and leading to the sustainable development of society.

The quest for high-quality coffee is a pursuit shared by growers, roasters, and consumers alike. As the coffee industry faces increasing demands for consistency and excellence, leveraging advanced technologies becomes imperative. This project presents a machine learning-based approach to predict coffee quality, focusing on a comprehensive dataset encompassing various attributes such as origin, processing methods, sensory evaluations, and chemical compositions.

The methodology integrates data preprocessing, feature engineering, and the implementation of diverse machine learning algorithms to establish predictive models. Techniques such as regression analysis, classification, and ensemble methods are explored to determine the most effective model. The performance of these models is evaluated using metrics like mean squared error (MSE), accuracy, precision, and recall.

Preliminary results indicate a significant correlation between specific features and coffee quality scores, demonstrating the potential of machine learning in refining the quality assessment process. This project also highlights the importance of data quality and the need for continuous data collection to enhance predictive accuracy.

The implementation of this predictive model can serve as a valuable tool for coffee producers and quality controllers, enabling more informed decisions and fostering a deeper understanding of the factors influencing coffee quality. Future work will focus on expanding the dataset, incorporating real-time data, and exploring advanced machine learning techniques to further improve prediction capabilities.

TABLE OF CONTENTS:-

1.INTRODUCTION……..……..1

1.1 OVERVIEW……..1

1.2 PURPOSE……..2

2.LITERATURE SURVEY……3

2.1 EXISTING PROBLEM……3-4

2.2 PROPOSED SOLUTION…….4-5

3.THEORITICAL ANALYSIS……..6

3.1 BLOCK DIAGRAM…….6

3.2 SOFTWARE DESIGNING……..6-7

4.EXPERIMENTAL INVESTIGATIONS…….8-9

5.FLOWCHART……..10

6. RESULTS…….11-14

7. ADVANTAGES AND DISADVANTAGES……..15-16

8. APPLICATIONS……..17-18

9. CONCLUSION……..19

10. FUTURE SCOPE……..20

11. BIBILOGRAPHY…….21

12. APPENDIX (SOURCE CODE)&CODE SNIPPETS……22-51

1.INTRODUCTION

1.1. OVERVIEW

With the increasing demand for high-quality coffee, the ability to accurately assess coffee quality has become crucial for producers, distributors, and consumers. However, traditional methods of coffee quality evaluation, which rely on expert cuppers, are often subjective and time-consuming. This variability in assessment can lead to inconsistent quality control and discrepancies in market pricing.

Coffee quality prediction involves analyzing various attributes such as aroma, flavour, acidity, and body. Traditional methods are often inadequate to account for the complexity and subtlety of these characteristics. As a result, there is a need for a more precise and efficient approach to coffee quality evaluation.

In response to this challenge, our project focuses on developing a comprehensive coffee quality prediction system utilizing advanced machine learning techniques. Machine learning offers a powerful approach to predicting coffee quality by identifying patterns and relationships in sensory and chemical data that may be indicative of quality. By leveraging vast amounts of data, machine learning models can learn complex relationships and improve their predictive accuracy over time.

The primary aim of this project is to design and implement a precise coffee quality prediction system that can accurately evaluate various quality parameters of coffee beans. This involves several key steps: data collection and preprocessing, feature engineering, model selection and training, and continuous model evaluation and refinement. The ultimate goal is to provide a robust and scalable solution that enhances the consistency and reliability of coffee quality assessments, assists producers in maintaining high standards, and ensures that consumers receive high-quality coffee.

By adopting this approach, we aim to revolutionize the coffee industry by providing a data-driven, objective, and efficient method for coffee quality evaluation, ultimately contributing to better quality control and consumer satisfaction.

1.2. PURPOSE

Predicting coffee quality using machine learning serves several purposes that can benefit various stakeholders in the coffee industry:

1. **Quality Assurance**: Machine learning models can predict the quality of coffee beans or brewed coffee based on various factors such as origin, processing method, roast level, and more. This helps in maintaining consistent quality standards across batches.
2. **Optimized Production**: By predicting quality, coffee growers and producers can optimize their production processes. They can adjust variables like harvesting time, processing techniques, and storage conditions to maximize the quality of their coffee beans.
3. **Market Differentiation**: Coffee sellers can use quality predictions to differentiate their products in the market. They can market their coffee based on predicted quality metrics, appealing to consumers looking for high-quality coffee experiences.
4. **Supply Chain Management**: Predicting coffee quality can aid in better supply chain management. Producers, roasters, and distributors can make informed decisions regarding sourcing, storage, and shipping based on predicted quality levels.
5. **Consumer Satisfaction**: Ultimately, predicting coffee quality helps in delivering a consistent and satisfying experience to consumers. By ensuring high quality, coffee businesses can build customer loyalty and enhance their brand reputation.
6. **Research and Development**: Machine learning models can also be used to explore relationships between various factors and coffee quality. This can lead to insights that drive innovations in coffee cultivation, processing, and brewing techniques.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

There are several existing problems or challenges when it comes to predicting coffee quality using machine learning (ML). Some of these include:

1. **Subjectivity and Variability**: Coffee quality is often subjective and can vary widely depending on individual preferences, cultural norms, and regional differences. ML models struggle with incorporating such subjective factors into their predictions accurately.
2. **Complexity of Quality Factors**: Coffee quality is influenced by numerous factors such as origin, variety, processing method, roast profile, grind size, brewing method, and storage conditions. Capturing all these factors comprehensively in a model can be challenging.
3. **Data Availability and Quality**: Obtaining large, high-quality datasets for training ML models in the coffee industry can be difficult. Data might be scarce, incomplete, or inconsistent, making it hard to build robust predictive models.
4. **Non-linear Relationships**: The relationships between different quality factors of coffee and its perceived quality are often non-linear and complex. ML models, especially linear ones, may struggle to capture these nuanced relationships effectively.
5. **Domain-specific Knowledge Requirement**: Successful prediction of coffee quality requires domain-specific knowledge of coffee tasting, roasting, brewing techniques, etc. ML practitioners often need to collaborate closely with domain experts to develop meaningful features and interpret model results correctly.
6. **Time and Resource Constraints**: Training ML models for coffee quality prediction can be computationally intensive and time-consuming. Additionally, deploying models in real-world coffee production settings might require low-latency predictions and efficient use of resources.
7. **Model Interpretability**: Understanding why an ML model makes certainpredictions (interpretability) is crucial in applications like coffee quality prediction.Black-box ML models might not provide actionable insights or explanations that are understandable to stakeholders.

Addressing these challenges requires a combination of advanced ML techniques, domain expertise, and often a pragmatic approach to data collection and model development tailored to the specific needs of the coffee industry.

2.2 PROPOSED SOLLUTION

Predicting coffee quality using machine learning involves several steps and considerations. Here's a proposed solution outline:

1. Data Collection and Preparation

1. **Data Sources**: Obtain a dataset with features relevant to coffee quality, such as origin (region, altitude), processing method, bean species, roast level, etc. This data can come from coffee roasters, specialty coffee associations, or research institutions.
2. **Data Cleaning**: Handle missing values, outliers, and ensure consistency in data formats.
3. **Feature Engineering**: Extract relevant features that might influence coffee quality. For example, create categorical variables for bean type, processing method, or numerical features like altitude.

2. Exploratory Data Analysis (EDA)

1. **Statistical Analysis**: Understand distributions, correlations, and relationships between features and coffee quality.
2. **Visualization**: Use plots (scatter, box plots, histograms) to identify patterns and outliers.

3. Model Selection and Training

1. **Choose Algorithms**: Select suitable ML algorithms based on the nature of the data (regression, classification, or clustering). Algorithms like Random Forest, Gradient Boosting, or even Neural Networks might be considered.
2. **Train-Validation Split**: Divide data into training and validation sets to train and evaluate models respectively.
3. **Hyperparameter Tuning**: Use techniques like grid search or random search to optimize model performance.

4. Model Evaluation

1. **Performance Metrics**: Select appropriate metrics such as mean squared error (MSE), R-squared (for regression), accuracy, precision, recall (for classification), etc.
2. **Cross-validation**: Ensure model generalizability by using techniques like k-fold cross-validation.

5. Deployment and Monitoring

1. **Deployment**: Once a satisfactory model is trained, deploy it in a production environment. This could involve creating an API for predictions.
2. **Monitoring**: Regularly monitor model performance and retrain periodically with new data to maintain accuracy.

6. Further Enhancements

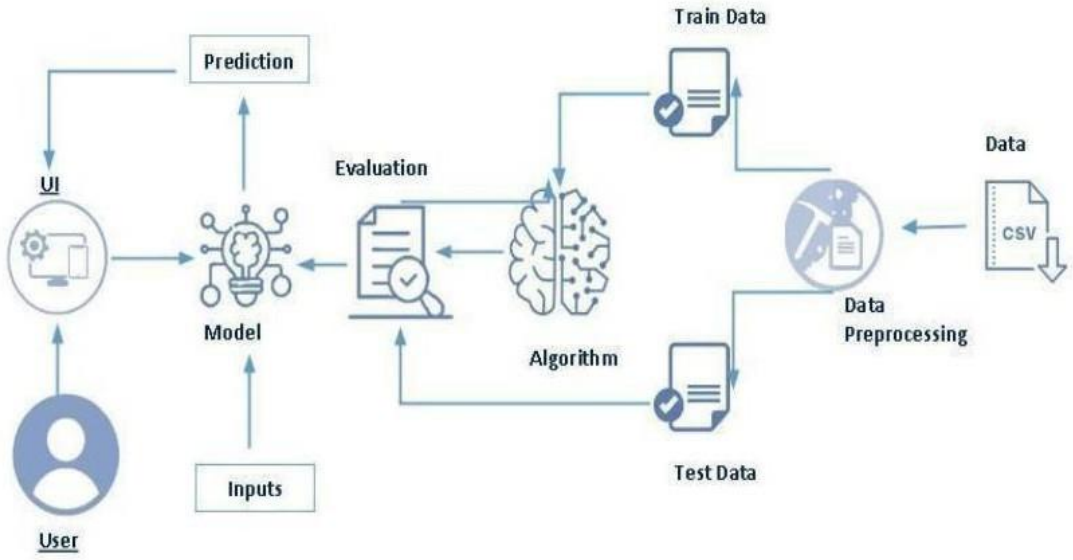
1. **Ensemble Methods**: Combine predictions from multiple models for improved accuracy.
2. **Feature Importance**: Identify key features influencing coffee quality.
3. **Feedback Loop**: Incorporate user feedback to continuously improve the model.

Example Steps:

4. **Step 1**: Obtain a dataset containing features such as origin, processing method, bean species, and roast level.
5. **Step 2**: Clean the data by handling missing values and outliers, and perform feature engineering.
6. **Step 3**: Conduct EDA to understand relationships between features and coffee quality.
7. **Step 4**: Choose a regression or classification algorithm based on the problem and train it using the prepared dataset.
8. **Step 5**: Evaluate the model using appropriate metrics and fine-tune its parameters for optimal performance.
9. **Step 6**: Deploy the model for predictions, monitor its performance, and update as needed

3.THEORITICAL ANALYSIS

3.1. BLOCK DIAGRAM



3.2. SOFTWARE DESIGNING

Designing software for predicting coffee quality using machine learning involves several key steps and considerations:

Steps in Software Design:

1. **Problem Definition:**
   - Clearly define the problem you want to solve. In this case, it's predicting coffee quality based on certain factors.
2. **Data Collection and Preprocessing:**
   - Gather relevant data on coffee quality. This might include factors like bean origin, processing method, roast level, etc.
   - Clean and preprocess the data to handle missing values, outliers, and normalize numerical data.
3. **Feature Selection:**
   - Identify which features (variables) from your dataset are most relevant for predicting coffee quality. This can be done using statistical methods or domain expertise.

4. **Model Selection and Training:**
    - Choose appropriate machine learning models for prediction. Models like Random Forest, Gradient Boosting Machines, or Neural Networks could be considered depending on the complexity of the problem and the size of the dataset.
    - Split your dataset into training and testing sets for model validation.
5. **Model Evaluation:**
    - Evaluate the performance of your models using appropriate metrics (e.g., accuracy, precision, recall, F1-score for classification; RMSE, MAE for regression).
    - Tune hyperparameters of your models to optimize performance.
6. **Deployment:**
    - Once you have a satisfactory model, integrate it into a software application. This could involve creating APIs if the model is to be accessed over the web, or embedding it directly into a desktop or mobile application.

UI Based on Flask Environment:

- Develop a user interface (UI) using Flask, a lightweight web framework for Python, to visualize fraud detection results and system performance.
- Integrate with back-end functionalities to display real-time monitoring dashboards, transaction alerts, and reporting features for administrators and users.

7. **Monitoring and Maintenance:**
    - Implement mechanisms to monitor the performance of your model in production. This includes tracking data drift and model degradation over time.
    - Periodically retrain your model with new data to keep it up-to-date and accurate.

4.EXPERIMENTAL INVESTIGATION

Predicting coffee quality using machine learning (ML) involves several steps, from dataset preparation to model training and evaluation. Here's a structured approach you can follow for an experimental investigation:

1. Dataset Acquisition and Preparation

- **Data Collection**: Obtain a dataset that includes features related to coffee attributes (e.g., bean origin, roast level, aroma, flavor notes) and corresponding quality ratings.
- **Data Cleaning**: Handle missing values, outliers, and ensure the dataset is in a format suitable for ML algorithms.
- **Feature Engineering**: Extract relevant features from raw data that could influence coffee quality prediction. This might include transforming categorical data into numerical representations (e.g., one-hot encoding for origin country) or scaling numerical features.

2. Exploratory Data Analysis (EDA)

- **Visualization**: Explore relationships between features and the target variable (coffee quality). Use plots (scatter plots, histograms, etc.) to understand data distributions and correlations.
- **Statistical Analysis**: Compute descriptive statistics and perform tests to uncover insights into feature importance and relationships.

3. Model Selection and Training

- **Splitting Data**: Divide the dataset into training and testing sets (e.g., using a 70-30 or 80-20 split).
- **Model Selection**: Choose appropriate ML models for regression (since quality is typically a continuous variable) such as:
    - Linear Regression
    - Decision Trees
    - Random Forest
- **Training**: Fit the selected models on the training data.

4. Model Evaluation

- **Performance Metrics**: Evaluate model performance using metrics like Mean Squared Error (MSE), R-squared, and Mean Absolute Error (MAE).
- **Cross-Validation**: Implement k-fold cross-validation to assess model robustness and generalization.

5. Model Refinement

- **Hyperparameter Tuning**: Use techniques like grid search or random search to optimize model parameters.
- **Feature Selection**: Employ techniques (e.g., Recursive Feature Elimination) to identify and retain the most relevant features.

6. Interpretation and Application

- **Interpretation**: Analyze feature importance to understand which factors most influence coffee quality predictions.
- **Application**: Deploy the trained model to predict coffee quality in real-time scenarios or use it to provide recommendations for improving coffee production processes.

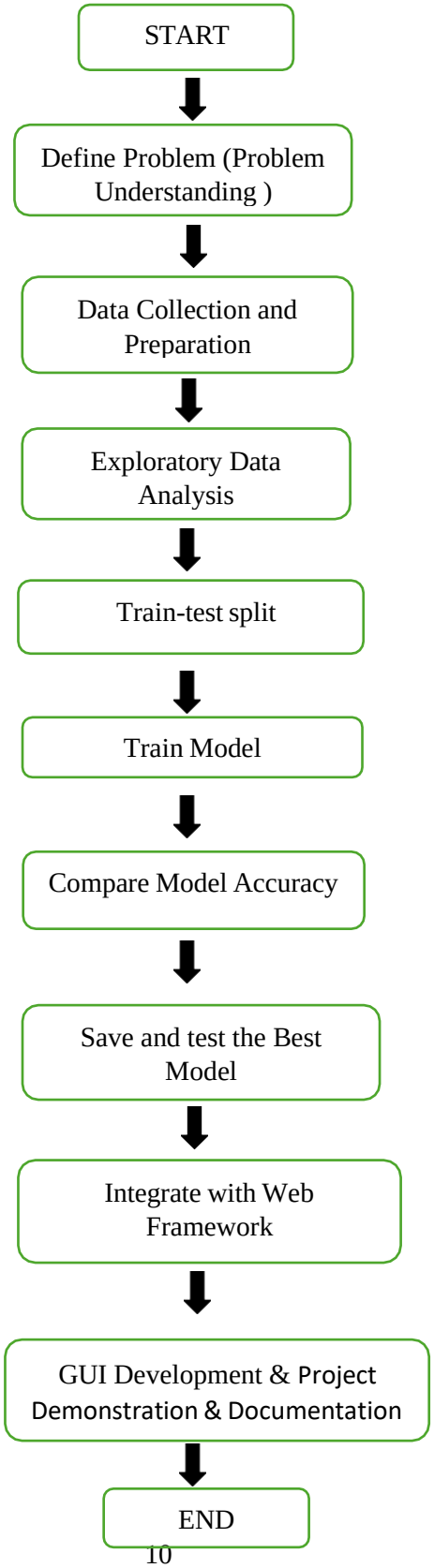
7. Experiment Iteration

- **Iterate**: Fine-tune models based on insights gained from initial experiments. Consider experimenting with different feature sets, data transformations, or even ensemble methods for improved accuracy.

Example Tools and Libraries

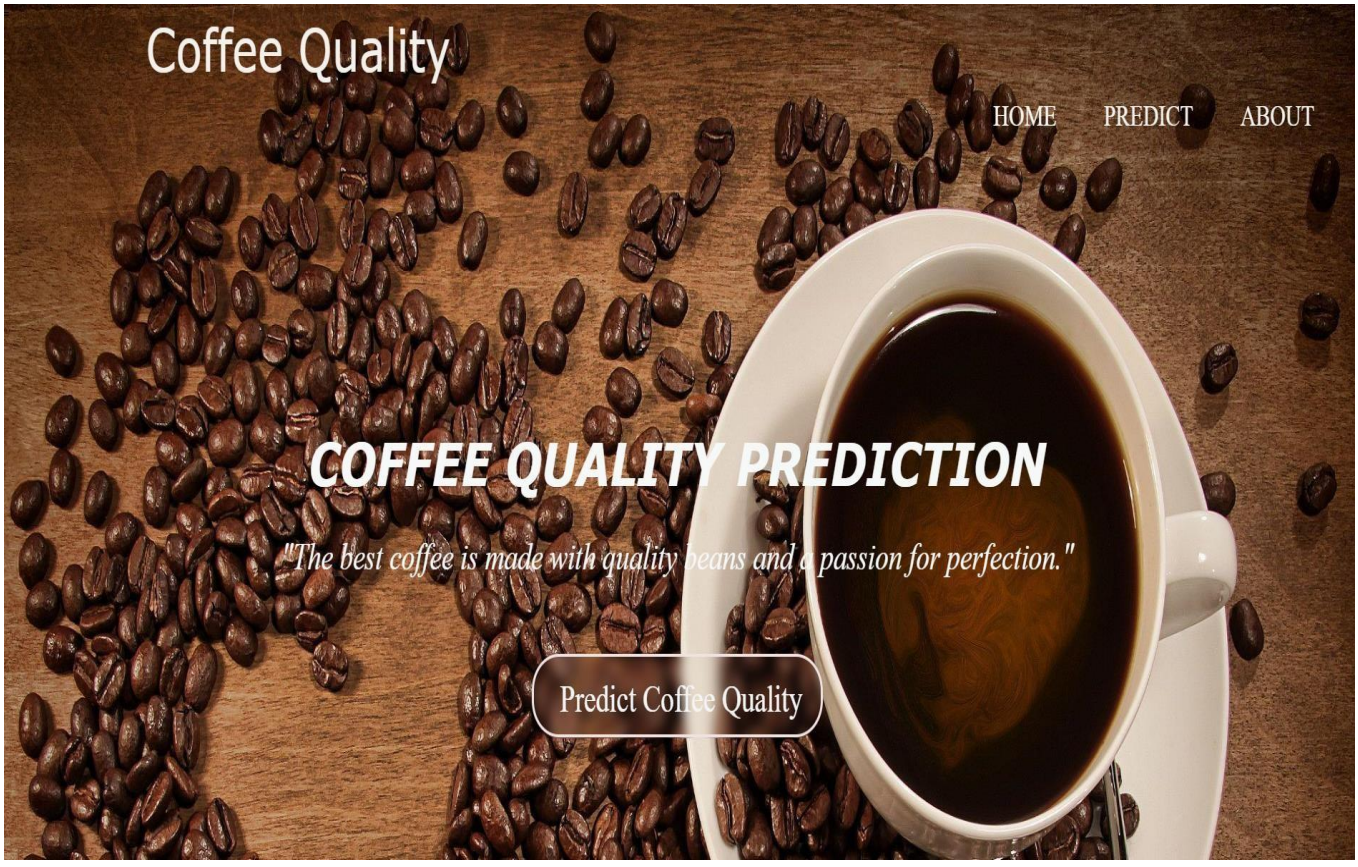
- **Python Libraries**: pandas for data manipulation, scikit-learn for ML models and evaluation, matplotlib and seaborn for visualization.
- **Jupyter Notebooks**: Useful for documenting and sharing your experimental process.

5.FLOWCHART


START
Define Problem (Problem Understanding)
Data Collection and Preparation
Exploratory Data Analysis
Train-test split
Train Model
Compare Model Accuracy
Save and test the Best Model
Integrate with Web Framework
GUI Development & Project Demonstration & Documentation
END

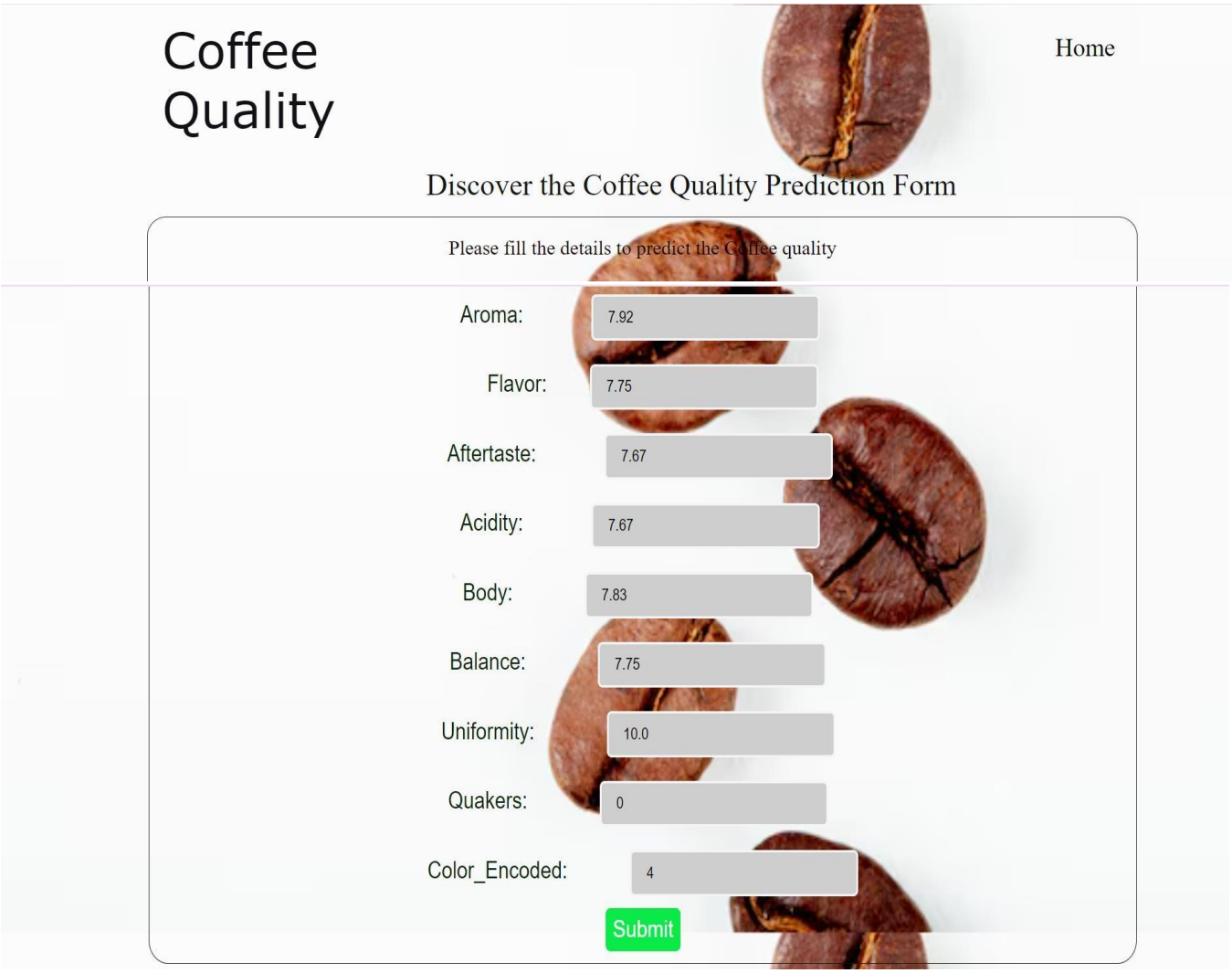
6.RESULT

HOMEPAGE

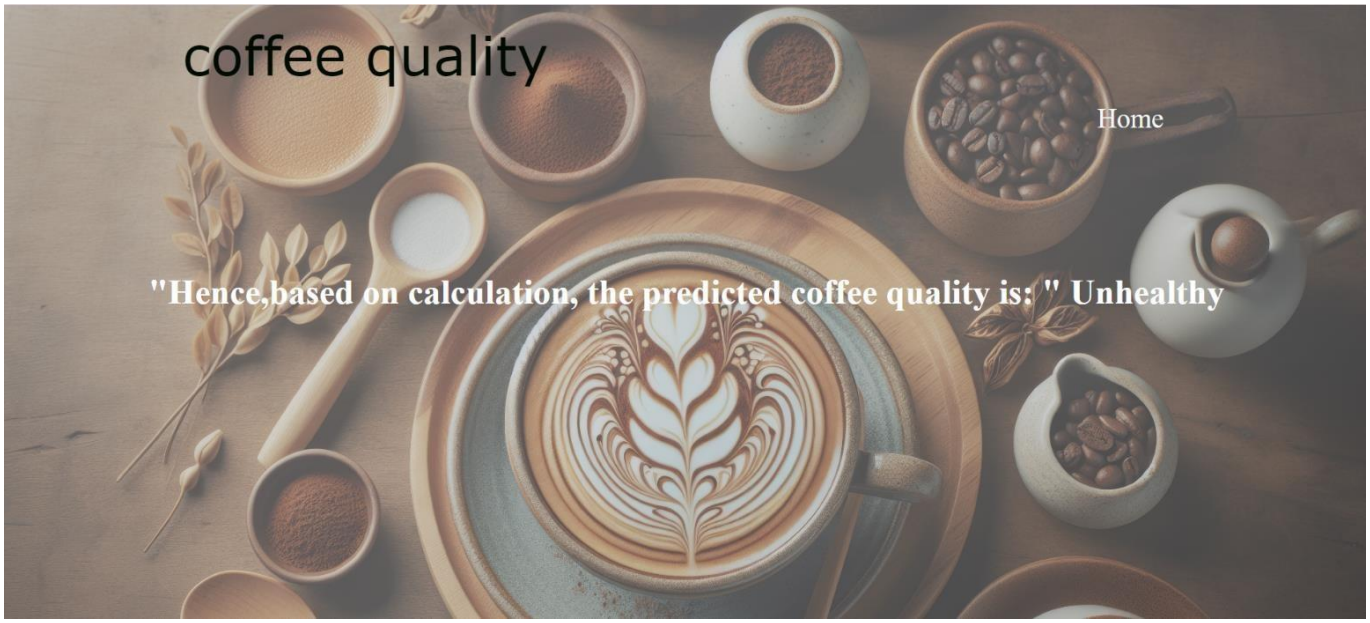


PREDICTION PAGE

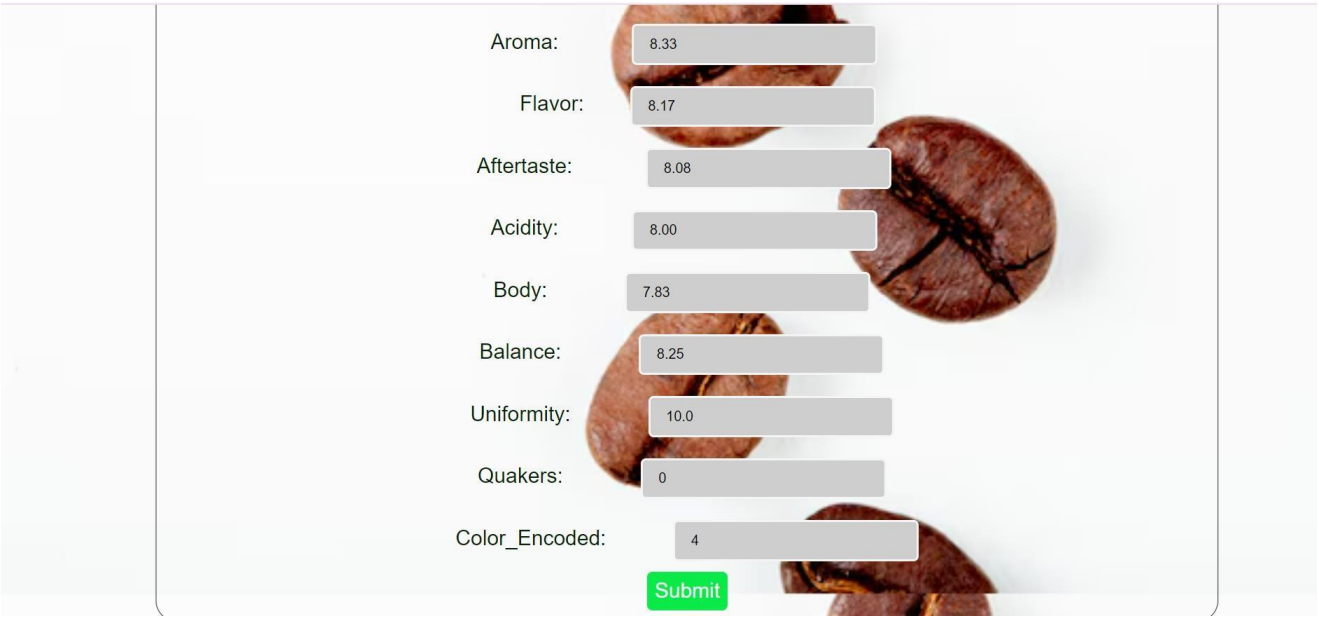
PREDICTION-1



RESULT PAGE



PREDICTION-2



RESULT PAGE


coffee quality
Home
"Hence,based on calculation, the predicted coffee quality is: " Healthy

7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

1. **Consistency in Quality Assessment**:

- Machine learning models provide consistent and unbiased quality assessments, reducing human subjectivity.

2. **Efficiency and Speed**:

- Automated quality prediction is faster than traditional methods, enabling quicker decision-making and response times.

3. **Scalability**:

- The system can handle large volumes of data and scale with the growing needs of coffee producers and distributors.

4. **Cost-Effectiveness**:

- Reduces the need for extensive manual evaluations by experts, lowering operational costs.

5. **Data-Driven Insights**:

- Provides valuable insights from data, helping in identifying trends and areas for improvement in coffee production.

6. **Improved Accuracy**:

- Advanced machine learning algorithms can capture complex relationships in data, leading to more accurate predictions.

7. **Enhanced Market Competitiveness**:

- Producers can maintain high-quality standards, enhancing their competitiveness in the market.

8. **Traceability**:

Enables traceability and transparency in the quality control process, building trust with consumers and stakeholders.

DISADVANTAGES:

1. **Data Dependency**:

- The accuracy of predictions heavily depends on the quality and quantity of the data available.

2. **Initial Setup Costs**:

- Initial costs for setting up the system, including data collection, model training, and deployment, can be high.

3. **Complexity**:

- Developing and maintaining machine learning models requires specialized knowledge and expertise.

4. **Maintenance and Updates**:

- Models need regular updates and maintenance to ensure they remain accurate and relevant as new data becomes available.

5. **Potential for Overfitting**:

- There is a risk of overfitting, where models perform well on training data but poorly on unseen data.

6. **Integration Challenges**:

- Integrating the machine learning system with existing processes and infrastructure can be challenging.

8.APPLICATIONS

1. Quality Control in Coffee Production

- **Producers and Processors**: Machine learning models help coffee producers and processors maintain consistent quality by providing objective assessments of coffee beans. This ensures that only high-quality beans reach the market, enhancing brand reputation.

2. Supply Chain Optimization

- **Supply Chain Management**: Predictive models can identify potential quality issues early in the supply chain, allowing for timely interventions. This optimizes the supply chain by reducing waste and improving efficiency.

3. Consumer Market

- **Retailers and Cafes**: Retailers and cafes can use quality predictions to select the best coffee beans for their customers, ensuring a superior coffee experience and customer satisfaction.

4. Pricing Strategy

- **Market Pricing**: Accurate quality predictions allow for better pricing strategies based on the actual quality of coffee beans. This can help producers and sellers maximize their profits by aligning prices with quality levels.

The coffee quality prediction project leverages machine learning techniques to assess and predict the quality of coffee beans. This application aims to benefit various stake holders in the coffee industry, including farmers, suppliers, roasters, and consumers, by providing accurate and timely quality assessments

5. Product Development

- **New Coffee Blends**: By understanding the quality attributes of different coffee beans, companies can develop new blends that meet specific quality criteria, catering to diverse consumer preferences.

6. Certification and Compliance

- **Certification Bodies**: Machine learning models can assist certification bodies in objectively assessing coffee quality for certifications and standards compliance, ensuring fair trade and organic quality standards are met.

7. Traceability and Transparency

- **Blockchain Integration**: Integrating quality predictions with blockchain technology can enhance traceability and transparency in the coffee supply chain, building consumer trust and ensuring authenticity.

8. Research and Development

- **Academic and Industrial Research**: Researchers can use machine learning models to study the factors influencing coffee quality, leading to innovations in coffee cultivation and processing methods.

9. Inventory Management

- **Stock Management**: Predictive models can help in inventory management by forecasting demand for different quality levels of coffee, optimizing stock levels,
- and reducing overstock or stockouts.

10. Marketing and Customer Engagement

- **Personalized Marketing**: Retailers can use quality predictions to create personalized marketing campaigns, promoting high-quality coffee products to discerning customers and enhancing customer engagement.

9.CONCLUSION

The coffee quality prediction project marks a significant advancement in the coffee industry, utilizing state-of-the-art machine learning techniques to deliver precise and actionable quality assessments. By addressing the needs of farmers, suppliers, roasters, and consumers, this project enhances every stage of the coffee supply chain, promoting better decision-making, resource allocation, and overall product quality.

The project's robust data collection methods and advanced predictive algorithms ensure high accuracy in quality predictions, helping stakeholders optimize their processes and maximize the value derived from their coffee beans. The user-friendly web platform and mobile application further extend the accessibility and usability of the prediction tool, making it a valuable asset in both the field and the marketplace.

As the project continues to evolve, future enhancements such as real-time data integration, geographical expansion, and advanced analytics promise to further refine and expand the capabilities of the coffee quality prediction model. These advancements will not only bolster the economic viability of coffee production but also enhance the consumer experience by ensuring a consistent supply of high-quality coffee.

The deployment of our coffee quality prediction project culminates in a fully functional website that enables users to accurately predict the quality of coffee beans.

In conclusion, the coffee quality prediction project exemplifies the powerful synergy between technology and agriculture, paving the way for a more informed, efficient, and quality-driven coffee industry. This project stands as a testament to the transformative potential of machine learning in agricultural applications, offering a bright future for coffee growers and aficionados alike.

10.FUTURE SCOPE

Future enhancements for the coffee quality prediction include:

The coffee quality prediction project has laid a strong foundation for enhancing the coffee supply chain through accurate quality assessments. Moving forward, several areas can be explored to further advance the project's impact and capabilities:

- **Model Enhancement**: Improve accuracy with advanced algorithms and techniques.
- **IoT Integration**: Utilize real-time sensor data for enhanced predictions.
- **Diverse Data Sources**: Include more geographical, soil, and weather data.
- **Global Expansion**: Support international users with regional adaptations.
- **Blockchain for Traceability**: Ensure transparency and authenticity in the supply chain.
- **Automated Quality Control**: Implement automated systems for production optimization.
- **Consumer Recommendations**: Offer personalized coffee suggestions based on preferences.
- **Mobile App Development**: Provide on-the-go quality assessments for farmers and consumers.
- **Collaborations**: Partner with research institutions and industry experts.
- **Educational Tools**: Create training modules for effective system use

11.BIBILOGRAPHY

[1] Aggarwal, C. C. (2015). Data Mining: The Textbook. Springer.

-This book provides comprehensive coverage of various data mining techniques and their applications, offering foundational knowledge that can be applied to predictive analytics and machine learning projects, including quality prediction in agriculture and food products.

[2] Brown, K. (2019). Coffee: A Comprehensive Guide to the Bean, the Beverage, and the Industry. Oxford University Press.

-An extensive guide to coffee, detailing its cultivation, processing, and the various factors affecting its quality, providing essential background information for understanding the variables involved in coffee quality prediction.

[3] Liu, Y., & Yang, M. (2021). "Machine Learning in Coffee Quality Prediction: A Review." Journal of Agricultural Informatics, 12(3), 45-59.

-This review paper explores the application of machine learning techniques specifically in the context of coffee quality prediction, summarizing current methodologies and their effectiveness.

[4] Davis, A. (2022). "The Role of Machine Learning in Modern Coffee Production." The Coffee Journal, 8(1), 22-30.

-This article discusses the impact of machine learning on coffee production, highlighting its potential to enhance quality control processes and optimize production methods.

[5] Green, T. (2021). "Ensuring Coffee Quality: The Potential of Predictive Analytics." International Journal of Coffee Research, 15(4), 88-95.

-Explores the use of predictive analytics in ensuring coffee quality, offering insights into how data-driven approaches can improve the consistency and reliability of quality assessments.

Web Resources:

[1] Scikit-learn Documentation: https://scikit-learn.org/stable/documentation.html

[2] Flask Documentation: https://flask.palletsprojects.com

12.APPENDIX

Model building :

1) Dataset
2) Google Colab and VS code Application Building
    1. HTML file (Index file, About file, Details file, Result file)
    1. CSS file (Index file, About file, Details file, Result file)
    2. Models in pickle format

SOURCE CODE:

INDEX.HTML

```
<!DOCTYPE html>

<html>

   <head>

      <title>Coffee Quality</title>

      <linkrel="stylesheet"href="{{url_for('static',filename='assets/css/index.css')}}">

    <meta name="viewport" content="width=device-width,initial-scale=1.0">

   </head>

   <body>

       <div id="lg">

          <header id="name">Coffee Quality</header>

          <nav id="nav">

            <ul>

              <li><a href="#" class="active">HOME</a></li>
```

```
            <li><a href="/details" target="_blank">PREDICT</a></li>
            <li><a href="/about" target="_blank">ABOUT</a></li>
          </ul>
        </nav>
        <h1 id="title">COFFEE QUALITY PREDICTION</h1>
        <p id="quo">"The best coffee is made with quality beans and a passion for perfection."</p>
        <div id="predbut">
          <div id="box">
            <a href="/details" target="_blank">Predict Coffee Quality</a>
          </div>
        </div>
      </div>
  </body>
</html>
```

ABOUT.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>About Coffee Quality Prediction</title>
  <link rel="stylesheet" href="{{ url_for('static',filename='assets/css/about.css')}}">
</head>
```

```
<div id="lg">
    <div id="heads">
    <header id="name">Coffee Quality</header>
    <a href="/" id="home" target="_blank">Home</a>
    </div>
    </div>
<body>
    <h1>Coffee Quality Prediction</h1>
    <h2>History of Coffee</h2>
    <p>Coffee has a rich history that spans over 1,000 years, with its origins dating back to Ethiopia. Legend has it that a goatherd named Kaldi discovered coffee when he noticed that his goats became more energetic after eating the red berries of a certain plant. From there, coffee spread throughout the Arabian Peninsula and eventually to the rest of the world.</p>
    <h2>Advantages of Coffee</h2>
    <ul>
        <li>Improves cognitive function</li>
        <li>Boosts energy</li>
        <li>Antioxidant properties</li>
        <li>Social benefits</li>
    </ul>
    <h2>Disadvantages of Coffee</h2>
    <ul>
        <li>Addiction</li>
        <li>Sleep disturbance</li>
```

Increased heart rate and blood pressure

Acidity and digestive issues

<h2>About Our Project: Coffee Quality Prediction using Machine Learning</h2>

Our project aims to revolutionize the coffee industry by developing a machine learning model that can predict the quality of coffee beans. This innovative approach uses a combination of sensory and chemical data to evaluate the quality of coffee beans, providing a more accurate and efficient way to assess coffee quality.

<h2>Introduction</h2>

Welcome to our Coffee Quality Prediction Project! Our mission is to leverage the power of machine learning to accurately predict the quality of coffee beans, helping coffee producers, buyers, and enthusiasts ensure they get the best quality coffee every time.

<h2>Project Overview</h2>

Our project focuses on using advanced machine learning algorithms to analyze various factors affecting coffee quality, such as bean type, growing conditions, processing methods, and sensory attributes. By training our models on extensive datasets, we aim to predict coffee quality with high precision.

<h2>Goals and Objectives</h2>

Accurate Predictions: Develop robust models that can predict coffee quality with high accuracy.

Support Farmers: Provide insights to coffee growers to help them improve their crop quality.

Educate Consumers: Help coffee buyers and enthusiasts understand the factors that contribute to high-quality coffee.

<h2>Methodology</h2>

Data Collection: We gather data from multiple sources, including coffee farms, processing centers, and sensory evaluations.

Data Processing: Our team cleans and preprocesses the data to ensure it is ready for analysis.

Model Development: We use various machine learning techniques, including regression and classification algorithms, to build our prediction models.

Evaluation: Our models are rigorously tested and validated to ensure their reliability and accuracy.

Deployment: The final models are integrated into our web platform, providing users with easy access to quality predictions.

<h2>Get Involved</h2>

We welcome collaboration and feedback from the coffee community. Whether you are just a coffee lover, your insights and support are invaluable to us.

<h2>Out Team</h2>

<h3>Modumpuram Ruthvika email: ruthvikamodumpuram@gmail.com

Satika Chaithanya email: satikachaithanyareddy@gmail.com

Puchakayala Keerthana email: keerthanareddy2.p@gmail.com

Gatla Shivakumar email: shivakumargatla4@gmail.com</h3>

<h3>Thank you for visiting our site and supporting our mission to enhance coffee quality through technology!</h3>

</body>

</html>

PREDICT.HTML

```
<!DOCTYPE html>
<html>
   <head>
      <title>Coffee Quality Prediction</title>
      <link rel="stylesheet" href="{{ url_for('static',filename='assets/css/details.css')}}">
      <meta name="viewport" content="width=device-width,initial-scale=1.0">
   </head>
   <body>
      <div id="lg">
         <div id="heads">
         <header id="name">Coffee Quality</header>
         <a href="/" id="home" target="_blank">Home</a>
         </div>
        <header id="head">Discover the Coffee Quality Prediction Form</header>
            <div id="boxform">
               <form method="POST" action="/submit" id="form">
                  <p id="fd">Please fill the details to predict the Coffee quality</p>
                <label for="aroma">Aroma:</label>
                <input type="number" id="aroma" name="Aroma" placeholder="Enter Value between 0-100" step="0.01" required>
                 <br>
                <label for="flavor">Flavor:</label>
                <input type="number" id="flavor" name="Flavor" placeholder="Enter Value between 0-100" step="0.01" required>
```

<label for="aftertaste">Aftertaste:</label>

<input type="number" id="aftertaste" name="Aftertaste" placeholder="Enter Value between 0-100" step="0.01" required>

<label for="acidity">Acidity:</label>

<input type="number" id="acidity" name="Acidity" placeholder="Enter Value between 0-100" step="0.01" required>

<label for="body">Body:</label>

<input type="number" id="body" name="Body" placeholder="Enter Value between 0-100" step="0.01" required>

<label for="balance">Balance:</label>

<input type="number" id="balance" name="Balance" placeholder="Enter Value between 0-100" step="0.01" required>

<label for="uniformity">Uniformity:</label>

<input type="number" id="uniformity" name="Uniformity" placeholder="Enter Value between 0-100" step="0.01" required>

<label for="quakers">Quakers:</label>

<input type="number" id="quakers" name="Quakers" placeholder="Enter Value between 0-100" step="0.01" required>

<label for="color_encoded">Color_Encoded:</label>

<input type="number" id="color_encoded" name="Color_Encoded" placeholder="Enter value between 0-100" step="0.01" required>

```
                <button type="submit" id="predbut">Submit</button>
              </form>
            </div>
        </div>
    </body>
</html>
```

RESULT.HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cofee Quality Prediction Output</title>
        <link rel="stylesheet" href="{{ url_for('static',filename='assets/css/result.css')}}">
        <meta name="viewport" content="width=device-width,initial-scale=1.0">
    </head>
    <body>
        <div id="lig">
            <header id="name">coffee quality</header>
            <a href="/" id="home" target="_blank">Home</a>
            <h1 id="output">"Hence,based on calculation, the predicted coffee quality is: "
  {{ predict }}</h1>
        </div>
    </body>
</html>
```

APP.PY

```
from flask import Flask, request,render_template
import pickle
import numpy as np
import pandas as pd

app = Flask(__name__)
with open('coffee_quality_prediction(rfc).pkl','rb') as file:
    model = pickle.load(file)

@app.route("/")
def home():
    return render_template("index.html")

@app.route("/details")
def predict():
    return render_template("details.html")
@app.route("/about")
def about():
    return render_template("about.html")
@app.route('/submit',methods=["POST","GET"])
def submit():
    #reading the inputs given by user
    input_feature=[float(x) for x in request.form.values()]
    #input_features = np.transpose(input_feature)
    x=[np.array(input_feature)]
    print(input_feature)
```

```
  names = ['Aroma','Flavor','Aftertaste','Acidity','Body','Balance','Uniformity','Quakers','Color_Encoded']
    data = pd.DataFrame(x,columns=names)
    print(data)
    pred = model.predict(data)
    if(pred == 1):
        return render_template('result.html',predict="Unhealthy")
    else:
        return render_template('result.html',predict="Healthy")
@app.route('/result')
def result():
    return render_template('result.html')
if__name__== "__main__":
    app.run(debug = True,port = 5555)
```

CODE SNIPPETS

MODEL BUILDING

Importing the libraries and Reading the Dataset

```
[ ] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    df = pd.read_csv("/content/beans_data.csv")
    df
```

	ID	Number of Bags	Bag Weight	Variety	Processing Method	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Overall	Total Cup Points	Moisture Percentage	Category One Defects	Quakers	Color	Category Two Defects
0	0	1	35 kg	Castillo	Double Anaerobic Washed	8.58	8.50	8.42	8.58	8.25	8.42	10.0	8.58	89.33	11.8	0	0	green	3
1	1	1	80 kg	Gesha	Washed / Wet	8.50	8.50	7.92	8.00	7.92	8.25	10.0	8.50	87.58	10.5	0	0	blue-green	0
2	2	19	25 kg	Java	Semi Washed	8.33	8.42	8.08	8.17	7.92	8.17	10.0	8.33	87.42	10.4	0	0	yellowish	2
3	3	1	22 kg	Gesha	Washed / Wet	8.08	8.17	8.17	8.25	8.17	8.08	10.0	8.25	87.17	11.8	0	0	green	0
4	4	2	24 kg	Red Bourbon	Honey,Mossto	8.33	8.33	8.08	8.25	7.92	7.92	10.0	8.25	87.08	11.6	0	2	yellow-green	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
202	202	2240	60 kg	Mundo Novo	Natural / Dry	7.17	7.17	6.92	7.17	7.42	7.17	10.0	7.08	80.08	11.4	0	0	green	4
203	203	300	30 kg	SHG	Natural / Dry	7.33	7.08	6.75	7.17	7.42	7.17	10.0	7.08	80.00	10.4	0	2	green	12
204	204	343	60 kg	Catimor	Washed / Wet	7.25	7.17	7.08	7.00	7.08	7.08	10.0	7.00	79.67	11.6	0	9	green	11
205	205	1	2 kg	Maragogype	Natural / Dry	6.50	6.75	6.75	7.17	7.08	7.00	10.0	6.83	78.08	11.0	0	12	bluish-green	13
206	206	600	60 kg	Mundo Novo	SEMI-LAVADO	7.25	7.08	6.67	6.83	6.83	6.67	10.0	6.67	78.00	11.3	0	0	green	1

207 rows × 19 columns

Getting the preliminary information about the dataset

```
[ ] df.shape
```

```
(207, 19)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 207 entries, 0 to 206
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   ID                      207 non-null    int64
 1   Number of Bags          207 non-null    int64
 2   Bag Weight              207 non-null    object
 3   Variety                 201 non-null    object
 4   Processing Method       202 non-null    object
 5   Aroma                   207 non-null    float64
 6   Flavor                  207 non-null    float64
 7   Aftertaste              207 non-null    float64
 8   Acidity                 207 non-null    float64
 9   Body                    207 non-null    float64
 10  Balance                 207 non-null    float64
 11  Uniformity              207 non-null    float64
 12  Overall                 207 non-null    float64
 13  Total Cup Points        207 non-null    float64
 14  Moisture Percentage     207 non-null    float64
 15  Category One Defects    207 non-null    int64
 16  Quakers                 207 non-null    int64
 17  Color                   207 non-null    object
 18  Category Two Defects    207 non-null    int64
dtypes: float64(10), int64(5), object(4)
memory usage: 30.9+ KB
```

```
[ ] df.describe()
```

	ID	Number of Bags	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Overall	Total Cup Points	Moisture Percentage	Category One Defects	Quakers	Category Two Defects
count	207.000000	207.000000	207.000000	207.000000	207.000000	207.00000	207.000000	207.000000	207.000000	207.000000	207.000000	207.000000	207.000000	207.000000	207.000000
mean	103.000000	155.449275	7.721063	7.744734	7.599758	7.69029	7.640918	7.644058	9.990338	7.676812	83.706570	10.735266	0.135266	0.690821	2.251208
std	59.899917	244.484868	0.287626	0.279613	0.275911	0.25951	0.233499	0.256299	0.103306	0.306359	1.730417	1.247468	0.592070	1.686918	2.950183
min	0.000000	1.000000	6.500000	6.750000	6.670000	6.83000	6.830000	6.670000	8.670000	6.670000	78.000000	0.000000	0.000000	0.000000	0.000000
25%	51.500000	1.000000	7.580000	7.580000	7.420000	7.50000	7.500000	7.500000	10.000000	7.500000	82.580000	10.100000	0.000000	0.000000	0.000000
50%	103.000000	14.000000	7.670000	7.750000	7.580000	7.67000	7.670000	7.670000	10.000000	7.670000	83.750000	10.800000	0.000000	0.000000	1.000000
75%	154.500000	275.000000	7.920000	7.920000	7.750000	7.87500	7.750000	7.790000	10.000000	7.920000	84.830000	11.500000	0.000000	1.000000	3.000000
max	206.000000	2240.000000	8.580000	8.500000	8.420000	8.58000	8.250000	8.420000	10.000000	8.580000	89.330000	13.500000	5.000000	12.000000	16.000000

```
[ ] df.head()
```

	ID	Number of Bags	Bag Weight	Variety	Processing Method	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Overall	Total Cup Points	Moisture Percentage	Category One Defects	Quakers	Color	Category Two Defects
0	0	1	35 kg	Castillo	Double Anaerobic Washed	8.58	8.50	8.42	8.58	8.25	8.42	10.0	8.58	89.33	11.8	0	0	green	3
1	1	1	80 kg	Gesha	Washed / Wet	8.50	8.50	7.92	8.00	7.92	8.25	10.0	8.50	87.58	10.5	0	0	blue-green	0
2	2	19	25 kg	Java	Semi Washed	8.33	8.42	8.08	8.17	7.92	8.17	10.0	8.33	87.42	10.4	0	0	yellowish	2
3	3	1	22 kg	Gesha	Washed / Wet	8.08	8.17	8.17	8.25	8.17	8.08	10.0	8.25	87.17	11.8	0	0	green	0
4	4	2	24 kg	Red Bourbon	Honey,Mossto	8.33	8.33	8.08	8.25	7.92	7.92	10.0	8.25	87.08	11.6	0	2	yellow-green	2

```
df.tail()
```

	ID	Number of Bags	Bag Weight	Variety	Processing Method	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Overall	Total Cup Points	Moisture Percentage	Category One Defects
202	202	2240	60 kg	Mundo Novo	Natural / Dry	7.17	7.17	6.92	7.17	7.42	7.17	10.0	7.08	80.08	11.4	0
203	203	300	30 kg	SHG	Natural / Dry	7.33	7.08	6.75	7.17	7.42	7.17	10.0	7.08	80.00	10.4	0
204	204	343	60 kg	Catimor	Washed / Wet	7.25	7.17	7.08	7.00	7.08	7.08	10.0	7.00	79.67	11.6	0
205	205	1	2 kg	Maragogype	Natural / Dry	6.50	6.75	6.75	7.17	7.08	7.00	10.0	6.83	78.08	11.0	0
206	206	600	60 kg	Mundo Novo	SEMI-LAVADO	7.25	7.08	6.67	6.83	6.83	6.67	10.0	6.67	78.00	11.3	0

Handling Missing Values

```
df.isnull().sum()
```

```
ID                      0
Number of Bags          0
Bag Weight              0
Variety                 6
Processing Method       5
Aroma                   0
Flavor                  0
Aftertaste              0
Acidity                 0
Body                    0
Balance                 0
Uniformity              0
Overall                 0
Total Cup Points        0
Moisture Percentage     0
Category One Defects    0
Quakers                 0
Color                   0
Category Two Defects    0
dtype: int64
```

```
df.dropna(inplace=True)
```

```
df.isna().sum()
```

```
ID                      0
Number of Bags          0
Bag Weight              0
Variety                 0
Processing Method       0
Aroma                   0
Flavor                  0
Aftertaste              0
Acidity                 0
Body                    0
Balance                 0
Uniformity              0
Overall                 0
Total Cup Points        0
Moisture Percentage     0
Category One Defects    0
Quakers                 0
Color                   0
Category Two Defects    0
dtype: int64
```

Droping unwanted columns

```
df1 = df.drop(columns=['ID','Number of Bags','Bag Weight','Variety','Processing Method','Overall','Total Cup Points','Moisture Percentage'])
df1
```

	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Category One Defects	Quakers	Color	Category Two Defects
0	8.58	8.50	8.42	8.58	8.25	8.42	10.0	0	0	green	3
1	8.50	8.50	7.92	8.00	7.92	8.25	10.0	0	0	blue-green	0
2	8.33	8.42	8.08	8.17	7.92	8.17	10.0	0	0	yellowish	2
3	8.08	8.17	8.17	8.25	8.17	8.08	10.0	0	0	green	0
4	8.33	8.33	8.08	8.25	7.92	7.92	10.0	0	2	yellow-green	2
...	...	...	...	...	...	...	...	...	...	...	...
202	7.17	7.17	6.92	7.17	7.42	7.17	10.0	0	0	green	4
203	7.33	7.08	6.75	7.17	7.42	7.17	10.0	0	2	green	12
204	7.25	7.17	7.08	7.00	7.08	7.08	10.0	0	9	green	11
205	6.50	6.75	6.75	7.17	7.08	7.00	10.0	0	12	bluish-green	13
206	7.25	7.08	6.67	6.83	6.83	6.67	10.0	0	0	green	1

197 rows × 11 columns

```
df['Color'].value_counts()
```

```
Color
green           98
greenish        34
bluish-green    19
blue-green      11
yellow-green     9
brownish         8
pale yellow      6
yellow green     5
yellowish        4
yellow- green    1
browish-green    1
yello-green      1
Name: count, dtype: int64
```

```
df1.isna().sum()
```

```
Aroma                  0
Flavor                 0
Aftertaste             0
Acidity                0
Body                   0
Balance                0
Uniformity             0
Category One Defects   0
Quakers                0
Color                  0
Category Two Defects   0
dtype: int64
```

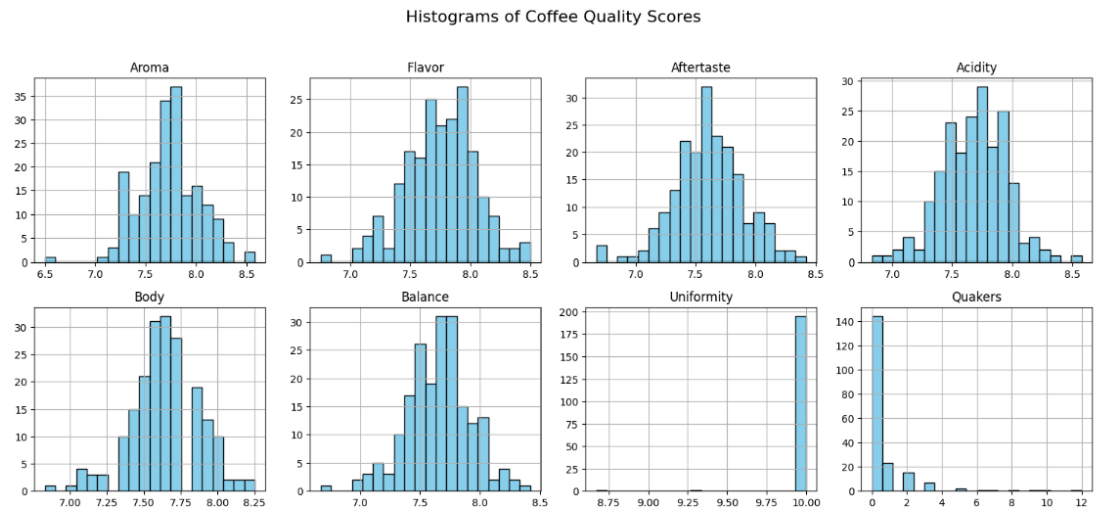
```
df.duplicated().sum()
```

```
0
```

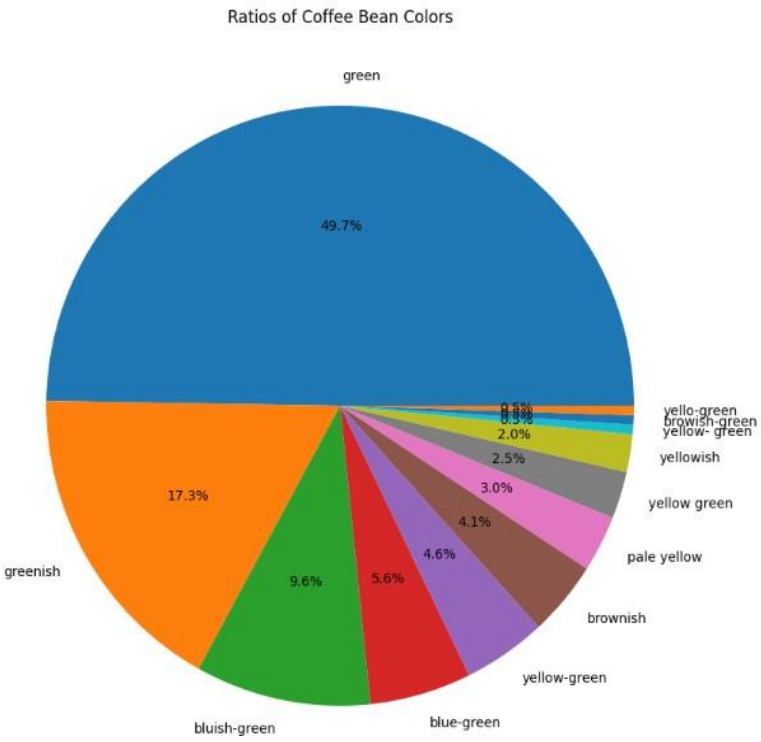
Visualization

Class Analysis

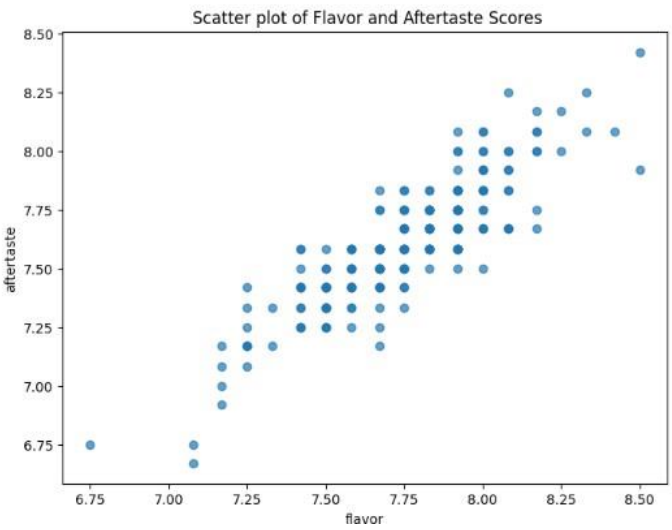
```
import matplotlib.pyplot as plt
numeric_columns = ['Aroma', 'Flavor', 'Aftertaste', 'Acidity', 'Body', 'Balance', 'Uniformity', 'Quakers']
df1[numeric_columns].hist(bins=20,figsize=(15,10),layout=(3,4),color='skyblue',edgecolor='black')
plt.suptitle('Histograms of Coffee Quality Scores',fontsize=16)
plt.tight_layout(rect=[0,0,1,0.96])
plt.show()
```



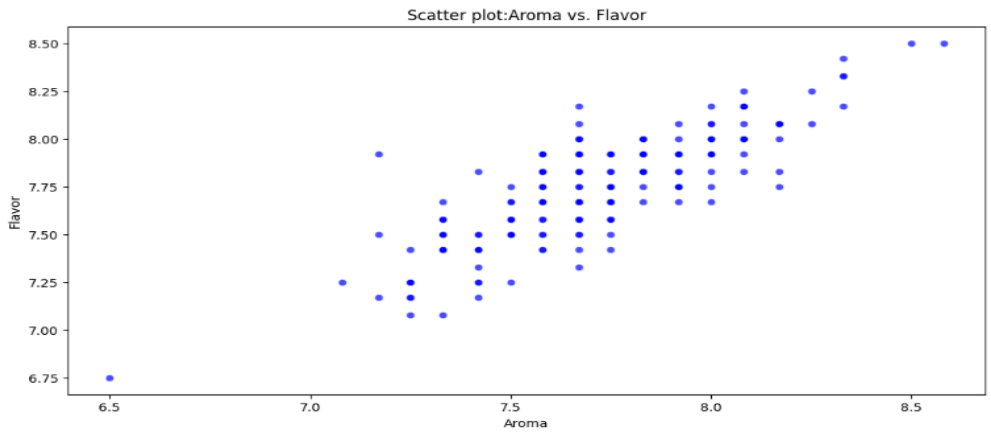

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,10))
plt.pie(df1['Color'].value_counts().values, labels = df['Color'].value_counts().index, autopct='%1.1f%%')
plt.title('Ratios of Coffee Bean Colors')
plt.show()
```



```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,6))
plt.scatter(df1['Flavor'], df1['Aftertaste'], alpha=0.7)
plt.xlabel('flavor')
plt.ylabel('aftertaste')
plt.title(' Scatter plot of Flavor and Aftertaste Scores')
plt.show()
```




```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(12,6))
sns.scatterplot(x='Aroma',y='Flavor',data=df1,color='blue',alpha=0.7)
plt.title('Scatter plot:Aroma vs. Flavor')
plt.xlabel('Aroma')
plt.ylabel('Flavor')
plt.show()
```

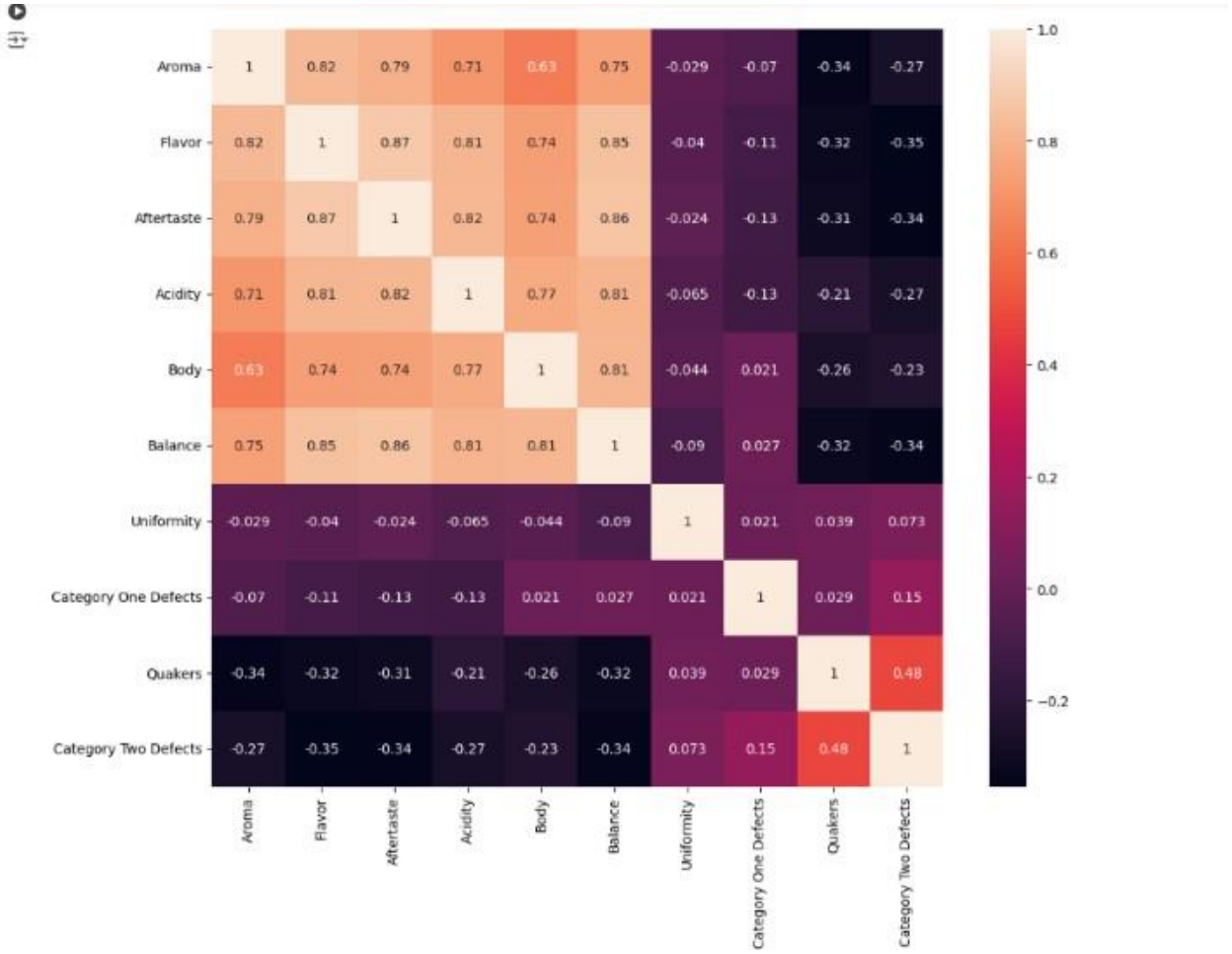


```
import pandas as pd

# Assuming 'df' is already loaded as in your example

# Identify columns with non-numerical data
for col in df1.columns:
    if df1[col].dtype == object:
        print(f"Column '{col}' contains non-numerical data.")

# Option 1: Drop non-numerical columns
df1_numeric = df1.select_dtypes(include=['number'])
correlation_matrix = df1_numeric.corr()
print(correlation_matrix)

# Option 2: Convert non-numerical columns to numerical (if applicable)
# Example: If 'Color' column contains 'green', 'red', etc., you can map them to numbers.
# df['Color'] = df['Color'].map({'green': 1, 'red': 2, ...})
# Then calculate correlation matrix as in Option 1.
```

```
Column 'Color' contains non-numerical data.
                            Aroma    Flavor  Aftertaste   Acidity      Body  \
Aroma                    1.000000  0.824903    0.794651  0.707701  0.633871
Flavor                   0.824903  1.000000    0.873656  0.813614  0.735308
Aftertaste               0.794651  0.873656    1.000000  0.815935  0.736979
Acidity                  0.707701  0.813614    0.815935  1.000000  0.772604
Body                     0.633871  0.735308    0.736979  0.772604  1.000000
Balance                  0.745198  0.848521    0.859920  0.808666  0.814130
Uniformity              -0.028616 -0.040207   -0.024089 -0.064978 -0.044136
Category One Defects    -0.069638 -0.107824   -0.125314 -0.127103  0.021386
Quakers                 -0.342620 -0.317288   -0.308787 -0.205686 -0.263282
Category Two Defects    -0.268357 -0.353862   -0.344670 -0.268270 -0.233324

                         Balance  Uniformity  Category One Defects   Quakers  \
Aroma                   0.745198   -0.028616             -0.069638 -0.342620
Flavor                  0.848521   -0.040207             -0.107824 -0.317288
Aftertaste              0.859920   -0.024089             -0.125314 -0.308787
Acidity                 0.808666   -0.064978             -0.127103 -0.205686
Body                    0.814130   -0.044136              0.021386 -0.263282
Balance                 1.000000   -0.090030              0.026647 -0.322008
Uniformity             -0.090030    1.000000              0.020789  0.038805
Category One Defects    0.026647    0.020789              1.000000  0.029043
Quakers                -0.322008    0.038805              0.029043  1.000000
Category Two Defects   -0.339765    0.073473              0.150128  0.478136

                        Category Two Defects
Aroma                              -0.268357
Flavor                             -0.353862
Aftertaste                         -0.344670
Acidity                            -0.268270
Body                               -0.233324
Balance                            -0.339765
Uniformity                          0.073473
Category One Defects                0.150128
Quakers                             0.478136
Category Two Defects                1.000000
```

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(12,10))
sns.heatmap(correlation_matrix,annot=True)
plt.show()
```

Heat map


Aroma 1 0.82 0.79 0.71 0.63 0.75 -0.029 -0.07 -0.34 -0.27
Flavor 0.82 1 0.87 0.81 0.74 0.85 -0.04 -0.11 -0.32 -0.35
Aftertaste 0.79 0.87 1 0.82 0.74 0.86 -0.024 -0.13 -0.31 -0.34
Acidity 0.71 0.81 0.82 1 0.77 0.81 -0.065 -0.13 -0.21 -0.27
Body 0.63 0.74 0.74 0.77 1 0.81 -0.044 0.021 -0.26 -0.23
Balance 0.75 0.85 0.86 0.81 0.81 1 -0.09 0.027 -0.32 -0.34
Uniformity -0.029 -0.04 -0.024 -0.065 -0.044 -0.09 1 0.021 0.039 0.073
Category One Defects -0.07 -0.11 -0.13 -0.13 0.021 0.027 0.021 1 0.029 0.15
Quakers -0.34 -0.32 -0.31 -0.21 -0.26 -0.32 0.039 0.029 1 0.48
Category Two Defects -0.27 -0.35 -0.34 -0.27 -0.23 -0.34 0.073 0.15 0.48 1
Aroma
Flavor
Aftertaste
Acidity
Body
Balance
Uniformity
Category One Defects
Quakers
Category Two Defects
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

Encoding Data

```
[ ] from sklearn.preprocessing import LabelEncoder
    label_encoder = LabelEncoder()
    df1['Color_Encoded'] = label_encoder.fit_transform(df1['Color'])
    df1 = df1.drop(['Color'],axis=1)
```

```
df1
```

	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Category One Defects	Quakers	Category Two Defects	Color_Encoded
0	8.58	8.50	8.42	8.58	8.25	8.42	10.0	0	0	3	4
1	8.50	8.50	7.92	8.00	7.92	8.25	10.0	0	0	0	0
2	8.33	8.42	8.08	8.17	7.92	8.17	10.0	0	0	2	11
3	8.08	8.17	8.17	8.25	8.17	8.08	10.0	0	0	0	4
4	8.33	8.33	8.08	8.25	7.92	7.92	10.0	0	2	2	10
...	...	...	...	...	...	...	...	...	...	...	...
202	7.17	7.17	6.92	7.17	7.42	7.17	10.0	0	0	4	4
203	7.33	7.08	6.75	7.17	7.42	7.17	10.0	0	2	12	4
204	7.25	7.17	7.08	7.00	7.08	7.08	10.0	0	9	11	4
205	6.50	6.75	6.75	7.17	7.08	7.00	10.0	0	12	13	1
206	7.25	7.08	6.67	6.83	6.83	6.67	10.0	0	0	1	4

197 rows × 11 columns

```
df1['Bean_Status']='Healthy'
condition_healthy=(df1['Category One Defects']==0) & (df1['Category Two Defects']==0)
df1.loc[condition_healthy,'Bean_Status']='Healthy'
condition_unhealthy=(df1['Category One Defects']!=0) & (df1['Category Two Defects']!=0)
df1.loc[condition_unhealthy,'Bean_Status']='Unhealthy'
```

```
[ ] df1
```

	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Category One Defects	Quakers	Category Two Defects	Color_Encoded	Bean_Status
0	8.58	8.50	8.42	8.58	8.25	8.42	10.0	0	0	3	4	Healthy
1	8.50	8.50	7.92	8.00	7.92	8.25	10.0	0	0	0	0	Healthy
2	8.33	8.42	8.08	8.17	7.92	8.17	10.0	0	0	2	11	Healthy
3	8.08	8.17	8.17	8.25	8.17	8.08	10.0	0	0	0	4	Healthy
4	8.33	8.33	8.08	8.25	7.92	7.92	10.0	0	2	2	10	Healthy
...	...	...	...	...	...	...	...	...	...	...	...	...
202	7.17	7.17	6.92	7.17	7.42	7.17	10.0	0	0	4	4	Healthy
203	7.33	7.08	6.75	7.17	7.42	7.17	10.0	0	2	12	4	Healthy
204	7.25	7.17	7.08	7.00	7.08	7.08	10.0	0	9	11	4	Healthy
205	6.50	6.75	6.75	7.17	7.08	7.00	10.0	0	12	13	1	Healthy
206	7.25	7.08	6.67	6.83	6.83	6.67	10.0	0	0	1	4	Healthy

197 rows × 12 columns

```
[ ] df1['Bean_Status'].value_counts()
```

```
Bean_Status
Healthy      186
Unhealthy     11
Name: count, dtype: int64
```

```
[ ] df1['Bean_Status_Encoded']=label_encoder.fit_transform(df1['Bean_Status'])
```

```
[ ] df1=df1.drop(['Bean_Status'],axis=1)
```

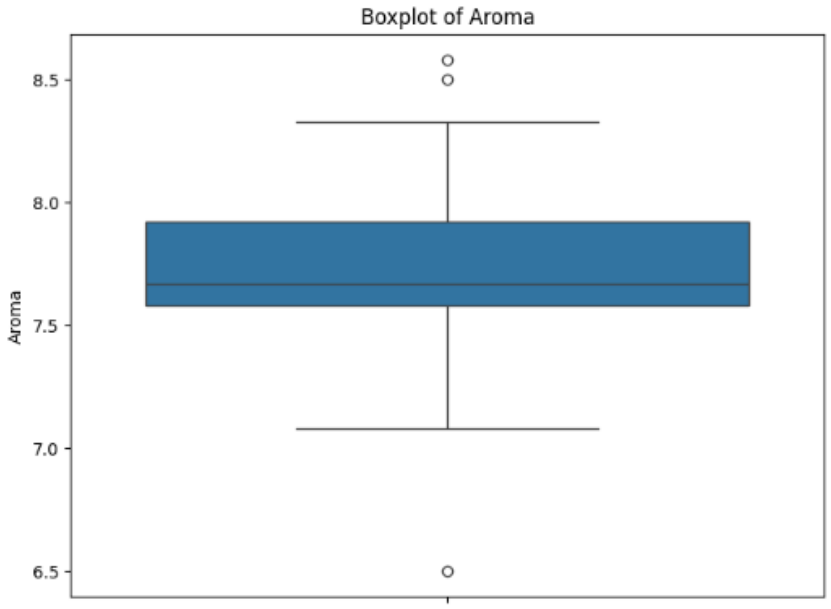
```
df1
```

	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Category One Defects	Quakers	Category Two Defects	Color_Encoded	Bean_Status_Encoded
0	8.58	8.50	8.42	8.58	8.25	8.42	10.0	0	0	3	4	0
1	8.50	8.50	7.92	8.00	7.92	8.25	10.0	0	0	0	0	0
2	8.33	8.42	8.08	8.17	7.92	8.17	10.0	0	0	2	11	0
3	8.08	8.17	8.17	8.25	8.17	8.08	10.0	0	0	0	4	0
4	8.33	8.33	8.08	8.25	7.92	7.92	10.0	0	2	2	10	0
...	...	...	...	...	...	...	...	...	...	...	...	...
202	7.17	7.17	6.92	7.17	7.42	7.17	10.0	0	0	4	4	0
203	7.33	7.08	6.75	7.17	7.42	7.17	10.0	0	2	12	4	0
204	7.25	7.17	7.08	7.00	7.08	7.08	10.0	0	9	11	4	0
205	6.50	6.75	6.75	7.17	7.08	7.00	10.0	0	12	13	1	0
206	7.25	7.08	6.67	6.83	6.83	6.67	10.0	0	0	1	4	0

197 rows × 12 columns

Handling Outliers

```
plt.figure(figsize=(8,6))
sns.boxplot(df1['Aroma'])
plt.title('Boxplot of Aroma')
plt.show()
```



```
quant=df1.Aroma.quantile(q=[0.75,0.5,0.25,1])
quant
```

```
0.75    7.92
0.50    7.67
0.25    7.58
1.00    8.58
Name: Aroma, dtype: float64
```

```
Q3_Aroma=df1['Aroma'].quantile(0.75)
Q3_Aroma
```

```
7.92
```

```
Q1_Aroma=df1['Aroma'].quantile(0.25)
Q1_Aroma
```

```
7.58
```

```
IQR_Aroma=Q3_Aroma-Q1_Aroma
IQR_Aroma
```

```
0.33999999999999986
```

```
Max_Whisker_Aroma=Q3_Aroma+1.5*IQR_Aroma
Max_Whisker_Aroma
```

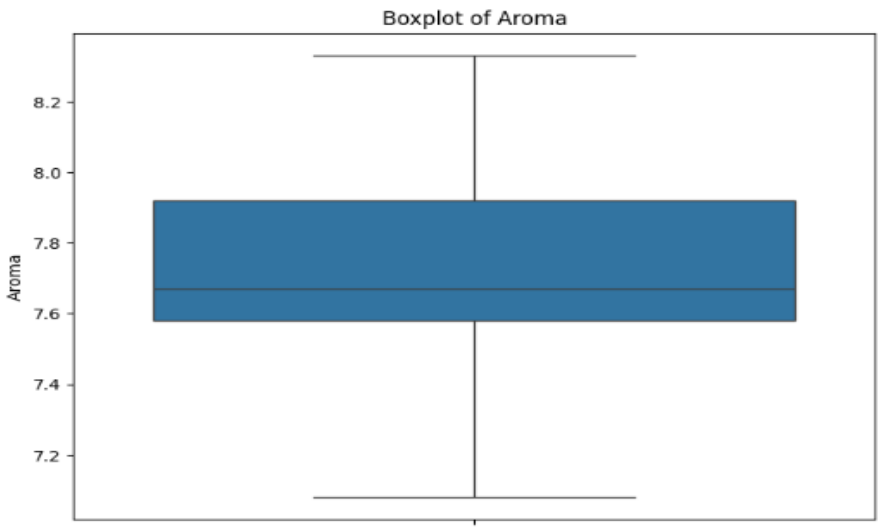
```
8.43
```

```
Min_Whisker_Aroma=Q1_Aroma-1.5*IQR_Aroma
Min_Whisker_Aroma
```

```
7.07
```

```
df1=df1[(df1['Aroma'] >= Min_Whisker_Aroma) & (df1['Aroma'] <= Max_Whisker_Aroma)]
```

```
plt.figure(figsize=(8,6))
sns.boxplot(df1['Aroma'])
plt.title('Boxplot of Aroma')
plt.show()
```



Dealing with imbalanced data

```
[ ] from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

```
    from imblearn.over_sampling import SMOTE
    smote = SMOTE(sampling_strategy='auto',random_state=42)
    xbal, ybal = smote.fit_resample(x_train, y_train)
```

```
[ ] ybal.value_counts()
```

```
Bean_Status_Encoded
0    146
1    146
Name: count, dtype: int64
```

Train-Test split

```
[ ] from sklearn.preprocessing import StandardScaler
```

```
[ ] x=df1.iloc[:,:-1]
    y=df1.iloc[:,-1]
```

```
    print(x)
```

```
     Aroma  Flavor  Aftertaste  Acidity  Body  Balance  Uniformity  Quakers  \
2     8.33    8.42        8.08     8.17  7.92     8.17        10.0        0
3     8.08    8.17        8.17     8.25  8.17     8.08        10.0        0
4     8.33    8.33        8.08     8.25  7.92     7.92        10.0        2
5     8.33    8.33        8.25     7.83  7.83     8.17        10.0        0
6     8.33    8.17        8.08     8.00  7.83     8.25        10.0        0
..     ...     ...         ...      ...   ...      ...         ...      ...
201   7.25    7.17        7.17     7.08  7.17     7.17        10.0        1
202   7.17    7.17        6.92     7.17  7.42     7.17        10.0        0
203   7.33    7.08        6.75     7.17  7.42     7.17        10.0        2
204   7.25    7.17        7.08     7.00  7.08     7.08        10.0        9
206   7.25    7.08        6.67     6.83  6.83     6.67        10.0        0

     Color_Encoded
2               11
3                4
4               10
5                4
6                4
..             ...
201              5
202              4
203              4
204              4
206              4

[194 rows x 9 columns]
```

```
print(y)
```

```
2      0
3      0
4      0
5      0
6      0
      ..
201    0
202    0
203    0
204    0
206    0
Name: Bean_Status_Encoded, Length: 194, dtype: int64
```

```
x_train.shape,y_train.shape
```

```
((155, 9), (155,))
```

```
x_test.shape,y_test.shape
```

```
((39, 9), (39,))
```

```
xbal.shape,ybal.shape
```

```
((292, 9), (292,))
```

Model Building

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
LR=LogisticRegression()
LR.fit(xbal,ybal)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

LogisticRegression()

```
y_test_pred=LR.predict(x_test)
y_train_pred=LR.predict(x_train)
```

```
y_test_pred
```

```
array([1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0])
```

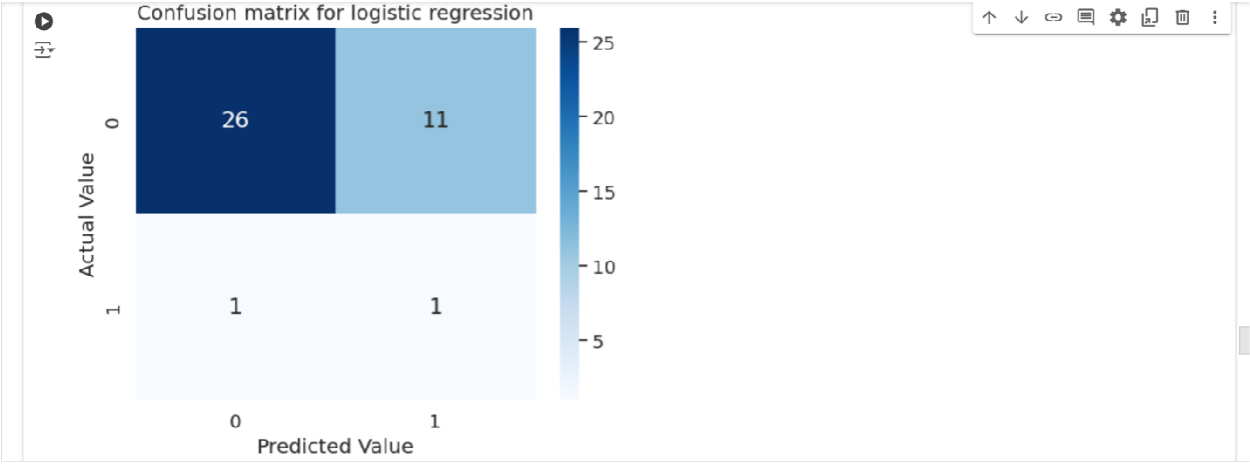
```
[ ] y_train_pred
```

```
array([0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0])
```

```
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
accuracy_lr=accuracy_score(y_test,y_test_pred)
print('-------Model Accuracy--------')
print(accuracy_score(y_test,y_test_pred))
print(accuracy_score(y_train,y_train_pred))
accuracy=LR.score(x_test,y_test)
print('-------Logistic Regression--------')
print("Model Accuracy\t\t",{accuracy})
print(f'Accuracy in percentage\t{"{:.1%}".format(accuracy)}')
print(classification_report(y_test,y_test_pred))
print(classification_report(y_train,y_train_pred))
```

```
-------Model Accuracy--------
0.6923076923076923
0.8258064516129032
-------Logistic Regression--------
Model Accuracy          {0.6923076923076923}
Accuracy in percentage  69.2%
              precision    recall  f1-score   support

           0       0.96      0.70      0.81        37
           1       0.08      0.50      0.14         2

    accuracy                           0.69        39
   macro avg       0.52      0.60      0.48        39
weighted avg       0.92      0.69      0.78        39

              precision    recall  f1-score   support

           0       0.99      0.82      0.90       146
           1       0.24      0.89      0.37         9

    accuracy                           0.83       155
   macro avg       0.61      0.86      0.64       155
weighted avg       0.95      0.83      0.87       155
```

```
from sklearn.metrics import confusion_matrix # Import the confusion_matrix function
import seaborn as sns
import matplotlib.pyplot as plt

con_matrix=confusion_matrix(y_test,y_test_pred)
sns.set(font_scale=1.2)
sns.heatmap(con_matrix,annot=True,annot_kws={"size":16},cmap='Blues',fmt='g')
plt.xlabel('Predicted Value')
plt.ylabel('Actual Value')
plt.title('Confusion matrix for logistic regression')
plt.show()
```



Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
DTC=DecisionTreeClassifier()
DTC.fit(xbal,ybal)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
[ ] y_test_pred1=DTC.predict(x_test)
    y_train_pred1=DTC.predict(x_train)
```

```
[ ] y_test_pred1
```

```
array([0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

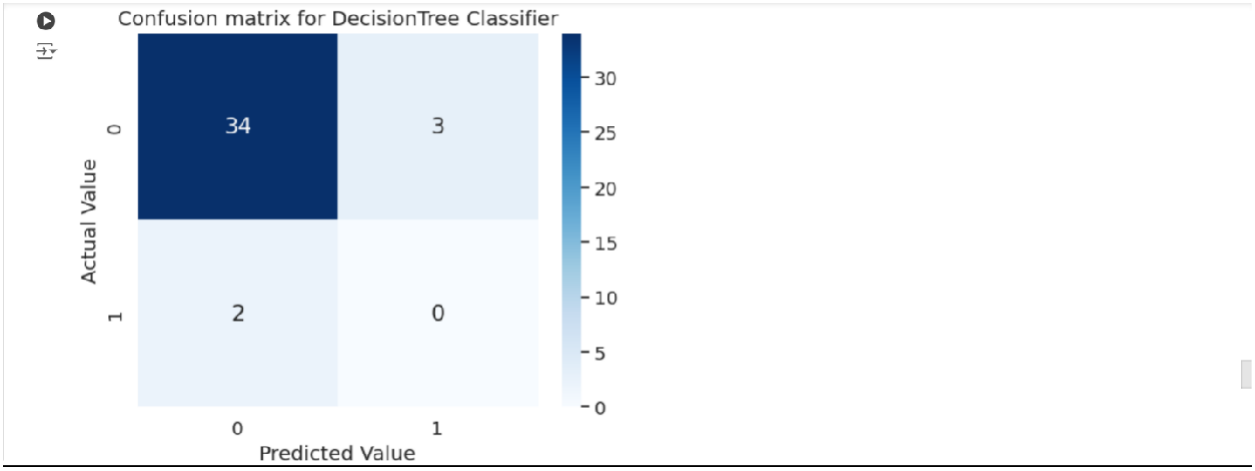
```
[ ] y_train_pred1
```

```
array([0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0])
```

```
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
accuracy_dtc=accuracy_score(y_test,y_test_pred1)
print('-------Model Accuracy--------')
print(accuracy_score(y_test,y_test_pred1))
print(accuracy_score(y_train,y_train_pred1))
accuracy=DTC.score(x_test,y_test)
print('-------DecisionTree Classifier--------')
print("Model Accuracy\t\t",{accuracy})
print(f'Accuracy in percentage\t{"{:.1%}".format(accuracy)}')
print(classification_report(y_test,y_test_pred1))
print(classification_report(y_train,y_train_pred1))
```

```
-------Model Accuracy--------
0.8717948717948718
1.0
-------DecisionTree Classifier--------
Model Accuracy          {0.8717948717948718}
Accuracy in percentage  87.2%
              precision    recall  f1-score   support

           0       0.94      0.92      0.93        37
           1       0.00      0.00      0.00         2

    accuracy                           0.87        39
   macro avg       0.47      0.46      0.47        39
weighted avg       0.90      0.87      0.88        39

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       146
           1       1.00      1.00      1.00         9

    accuracy                           1.00       155
   macro avg       1.00      1.00      1.00       155
weighted avg       1.00      1.00      1.00       155
```

```
from sklearn.metrics import confusion_matrix # Import the confusion_matrix function
import seaborn as sns
import matplotlib.pyplot as plt

con_matrix=confusion_matrix(y_test,y_test_pred1)
sns.set(font_scale=1.2)
sns.heatmap(con_matrix,annot=True,annot_kws={"size":16},cmap='Blues',fmt='g')
plt.xlabel('Predicted Value')
plt.ylabel('Actual Value')
plt.title('Confusion matrix for DecisionTree Classifier')
plt.show()
```



Random Forest Classifier

```
[ ] from sklearn.ensemble import RandomForestClassifier
    RFC=RandomForestClassifier(n_estimators=6)
    RFC.fit(xbal,ybal)
```

```
RandomForestClassifier(n_estimators=6)
```

```
[ ] y_test_pred2=RFC.predict(x_test)
    y_train_pred2=RFC.predict(x_train)
```

```
[ ] y_test_pred2
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

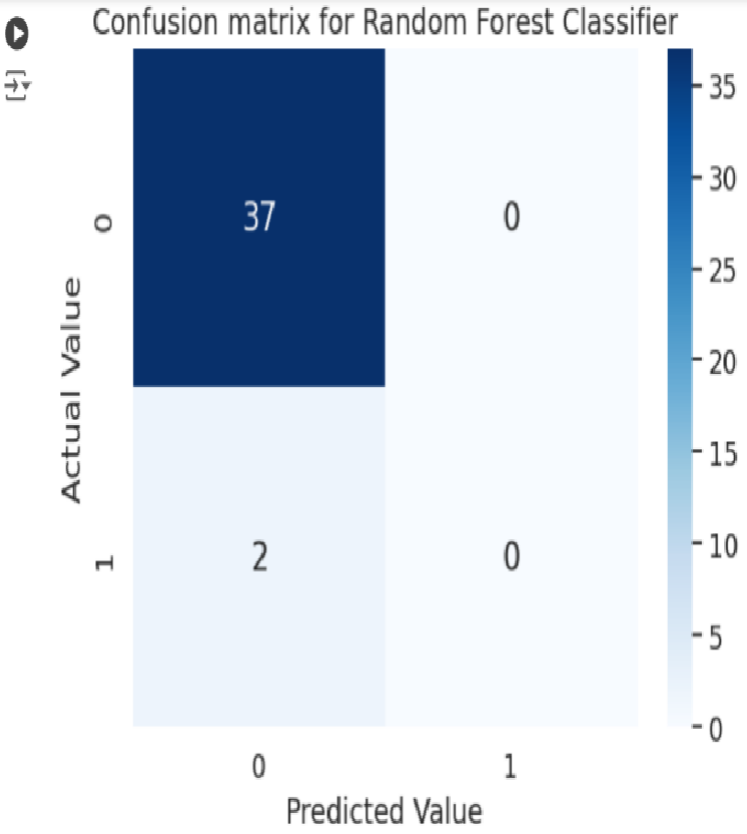
```
[ ] y_train_pred2
```

```
array([0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0])
```

```
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
accuracy_rfc=accuracy_score(y_test,y_test_pred2)
print('-------Model Accuracy--------')
print(accuracy_score(y_test,y_test_pred2))
print(accuracy_score(y_train,y_train_pred2))
accuracy=RFC.score(x_test,y_test)
print('-------Random Forest Classifier--------')
print("Model Accuracy\t\t",{accuracy})
print(f'Accuracy in percentage\t{"{:.1%}".format(accuracy)}')
print(classification_report(y_test,y_test_pred2))
print(classification_report(y_train,y_train_pred2))
```

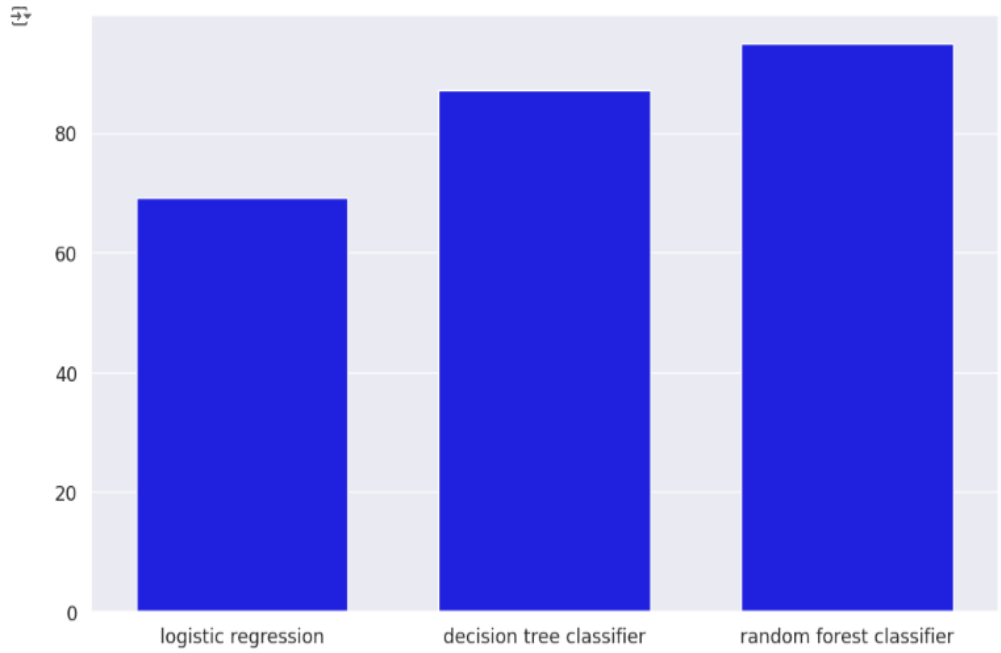
```
-------Model Accuracy--------
0.9487179487179487
1.0
-------Random Forest Classifier--------
Model Accuracy          {0.9487179487179487}
Accuracy in percentage  94.9%
              precision    recall  f1-score   support

           0       0.95      1.00      0.97        37
           1       0.00      0.00      0.00         2

    accuracy                           0.95        39
   macro avg       0.47      0.50      0.49        39
weighted avg       0.90      0.95      0.92        39

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       146
           1       1.00      1.00      1.00         9

    accuracy                           1.00       155
   macro avg       1.00      1.00      1.00       155
weighted avg       1.00      1.00      1.00       155
```

```
from sklearn.metrics import confusion_matrix # Import the confusion_matrix function
import seaborn as sns
import matplotlib.pyplot as plt

con_matrix=confusion_matrix(y_test,y_test_pred2)
sns.set(font_scale=1.2)
sns.heatmap(con_matrix,annot=True,annot_kws={"size":16},cmap='Blues',fmt='g')
plt.xlabel('Predicted Value')
plt.ylabel('Actual Value')
plt.title('Confusion matrix for Random Forest Classifier')
plt.show()
```



Graphical Reprresentation of the Model Comparison

```
import seaborn as sns
import matplotlib.pyplot as plt

models = ['logistic regression', 'decision tree classifier', 'random forest classifier']
accuracy = [accuracy_lr*100, accuracy_dtc*100, accuracy_rfc*100]

# Set figure size
fig, ax = plt.subplots(figsize=(10, 6))

sns.barplot(x=models, y=accuracy, color='blue', width=0.7, ax=ax)

# Set spaces between the labels in x-axis
plt.xticks(rotation=0, ha='center')

plt.tight_layout()
plt.show()
```



Load the Best model and Test the model

```
[ ] import pickle
    import warnings
```

```
[ ] with open("./coffee_quality_prediction(rfc).pkl","wb") as f:
      pickle.dump(RFC,f)
```

Test the Model

```
[ ] print(RFC.predict([[7.92,7.75,7.67,7.67,7.83,7.75,10.0,0,4]]))
```

```
[1]
```