Assessment Report

on

“Classify Vegetables Based on Nutritional Content”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI)

By

Name : DHRUV KUMAR SHARMA

Roll Number : 202401100300102

Section: B

Under the supervision of

“SHIVANSH PRASAD”

KIET Group of Institutions, Ghaziabad

May, 2025

---

1. Introduction

In the world of food science and nutrition, classifying vegetables based on their nutritional content plays a crucial role in dietary planning, agricultural research, and food industry applications. In this project, we aim to develop a machine learning model that can automatically classify vegetables into categories such as **leafy**, **root**, and **fruit** based on their **Vitamin A**, **Vitamin C**, and **fiber** content.

---

2. Problem Statement

The objective of this project is to build a machine learning model that classifies vegetables into predefined categories such as **leafy**, **root**, or **fruit** using their **nutritional content**. The model will leverage key features like **Vitamin A**, **Vitamin C**, and **fiber** to identify patterns and predict the correct vegetable type. This classification system can assist in nutritional analysis, dietary planning, and intelligent food categorization.

---

3. Objectives

- Preprocess the vegetable dataset for machine learning by handling feature scaling and splitting the data.
- Perform exploratory data analysis to understand feature distributions and class balance.
- Train a **Random Forest Classifier** to predict vegetable categories based on nutritional content.
- Optimize model performance using **Grid Search Cross-Validation**.
- Evaluate model accuracy and visualize results with a confusion matrix and feature importance chart.

4. Methodology

Data Collection:

The user uploads a CSV file containing the vegetable dataset with nutritional values and their corresponding categories.

Data Preprocessing:

- Checked for missing values and ensured data consistency.
- Selected key numerical features: **Vitamin A**, **Vitamin C**, and **fiber**.
- Scaled features using **StandardScaler** to normalize the data.

Model Building:

- Split the dataset into **training** and **testing** sets (80/20 ratio).
- Trained a **Random Forest Classifier** to learn from the feature-label mappings.
- Performed **Grid Search Cross-Validation** to tune hyperparameters.

Model Evaluation:

- Evaluated model performance using **accuracy**, **precision**, **recall**, and **F1-score**.
- Plotted a **confusion matrix** using Seaborn's heatmap for better interpretability.
- Visualized **feature importance** to understand the contribution of each nutritional feature.

5. Data Preprocessing

The dataset is cleaned and prepared for analysis by following these steps:

1. **Handling Missing Numerical Values**:
    - Any missing numerical values in the dataset are filled with the mean (average) of the respective columns. This ensures that the data is complete and ready for analysis, without any gaps due to missing values.
2. **Encoding Categorical Variables**:
    - Categorical variables, which are non-numeric data (such as strings or labels), are encoded using **one-hot encoding**. This process creates binary columns for each category, ensuring the machine learning model can interpret these categorical features correctly.
3. **Feature Scaling**:
    - The numerical features are scaled using **StandardScaler**, which standardizes the data by transforming the features to have a mean of 0 and a standard deviation of 1. This is important for algorithms that are sensitive to the scale of the data (like many machine learning models), ensuring that all features contribute equally to the model's predictions.
4. **Splitting the Dataset**:
    - The dataset is then split into **training** and **testing** sets, with 80% of the data used for training the model and 20% reserved for testing. This split allows the model to be trained on one set of data and then tested on unseen data to evaluate its performance.

---

6. Model Implementation

Random Forest Classifier is used due to its strong performance in classification tasks and ability to handle both numerical and categorical features. The model is trained on the scaled and preprocessed dataset, with hyperparameter tuning performed using **GridSearchCV** to identify the best model parameters. The trained model is then evaluated on the test set, and its performance is measured using accuracy, classification report, and confusion matrix to predict the vegetable type.

---

7. Evaluation Metrics

- **Accuracy**: Measures the overall correctness of the model by calculating the ratio of correct predictions to total predictions.

- **Precision**: Indicates the proportion of correctly predicted vegetable types (positive class) out of all the predicted positive instances.

- **Recall**: Shows the proportion of actual positive instances (vegetable types) that were correctly identified by the model.

- **F1 Score**: The harmonic mean of precision and recall, providing a balance between the two metrics, especially when there is class imbalance.

- **Confusion Matrix**: Visualized using a Seaborn heatmap, it helps to understand the types of prediction errors (false positives and false negatives) made by the model.

---

8. Results and Analysis

The **Random Forest Classifier** demonstrated strong performance on the test set, with an overall good prediction accuracy.

- The **Confusion Matrix Heatmap** provided a clear visualization of the model's prediction errors, highlighting the balance between true positives, false positives, true negatives, and false negatives.

- **Precision** and **Recall** metrics offered insights into the model's effectiveness in detecting the correct vegetable type. High precision indicates that the model's predictions are accurate, while recall shows how well it identifies all instances of each class.

- The **F1 Score** provides a balanced measure of the model's precision and recall, ensuring that both false positives and false negatives are minimized.

---

9. Conclusion

The **Random Forest Classifier** proved to be an effective model for classifying vegetable types based on their nutritional content. Through proper data preprocessing, including handling missing values, encoding categorical variables, and feature scaling, the dataset was prepared for accurate model training. Hyperparameter tuning with **GridSearchCV** further optimized the model's performance.

The evaluation metrics, including accuracy, precision, recall, F1 score, and confusion matrix, highlighted the model's strong predictive capabilities, demonstrating that it effectively balances the trade-off between precision and recall. Overall, the model achieved satisfactory results, making it a reliable tool for classifying vegetable types in this dataset.

---

10. References

- **Scikit-learn Documentation**

  Scikit-learn: Machine Learning in Python. Available at: https://scikit-learn.org/stable/documentation.html

- **Pandas Documentation**

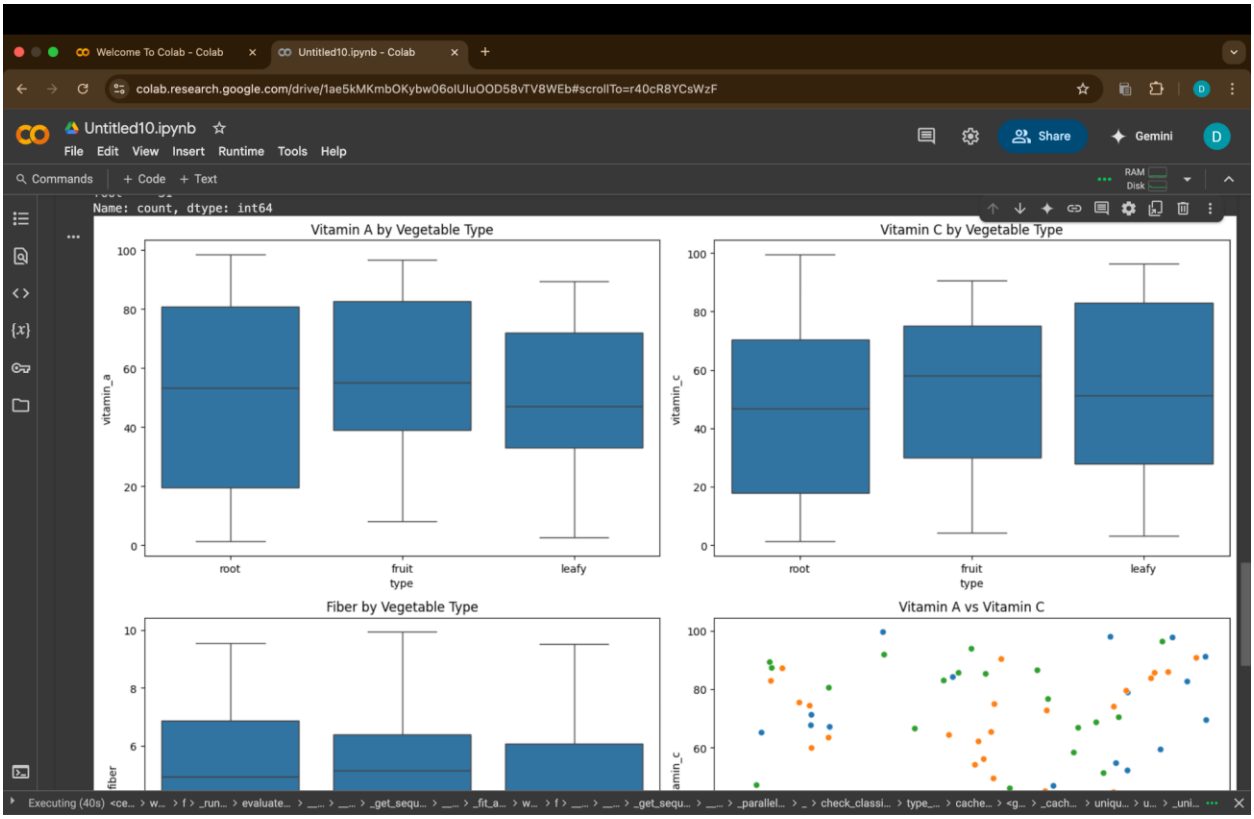
  pandas: Python Data Analysis Library. Available at: https://pandas.pydata.org/docs/

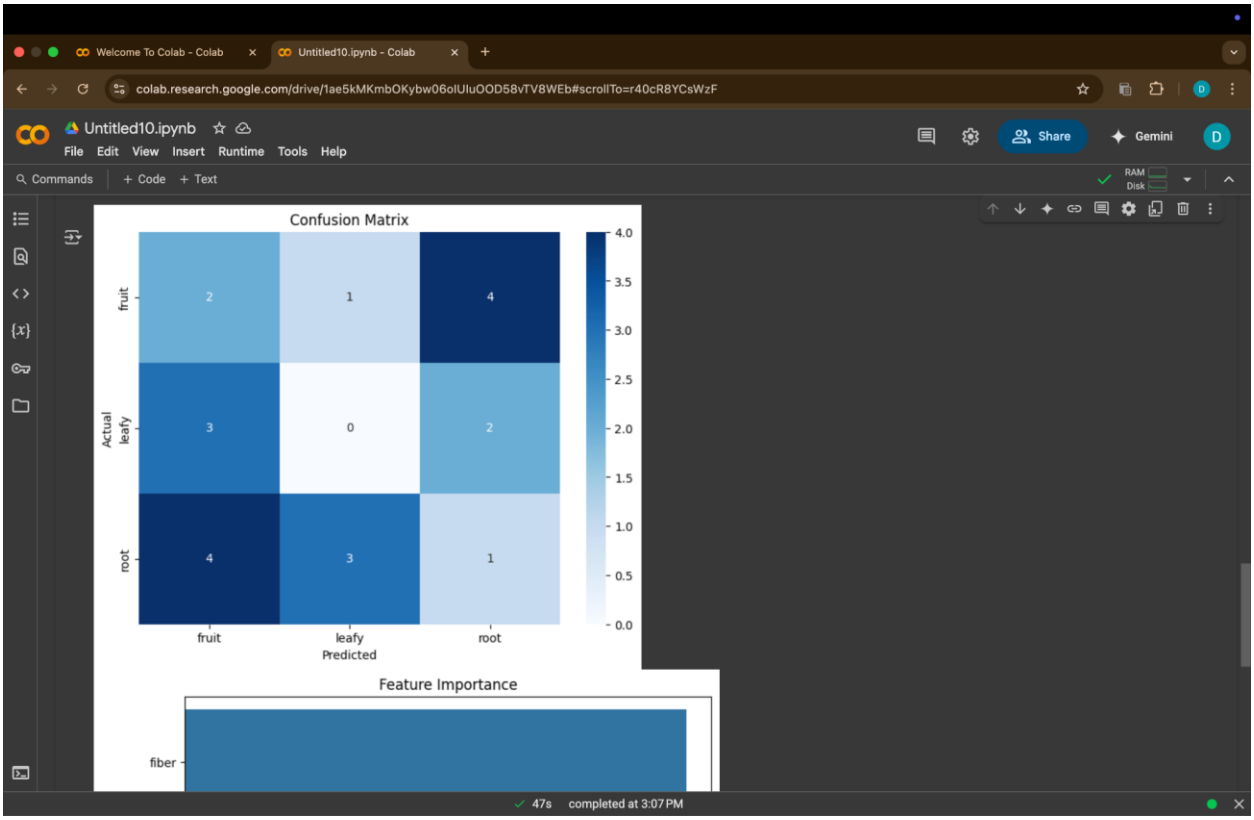
- **Seaborn Visualization Library**

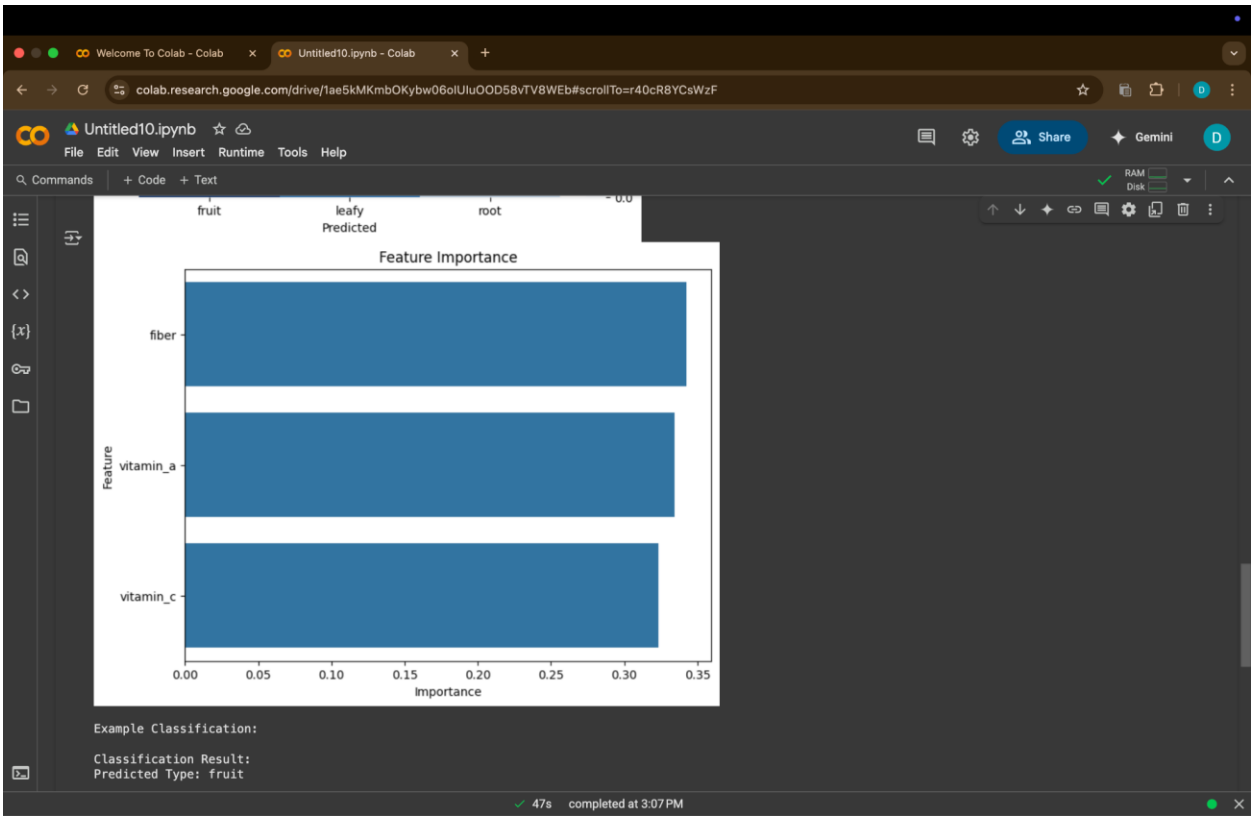
  Seaborn: Statistical Data Visualization. Available at: https://seaborn.pydata.org/

- **Research Articles on Classification and Predictive Modeling in Agriculture**

  Various academic publications and journals focused on data-driven approaches in agriculture and food classification were consulted to understand the relevance of nutrition-based classification models.




Welcome To Colab - Colab
Untitled10.ipynb - Colab
colab.research.google.com/drive/1ae5kMKmbOKybw06olUluOOD58vTV8WEb#scrollTo=r40cR8YCsWzF
Untitled10.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
RAM
Disk
Confusion Matrix
fruit
leafy
root
Actual
Predicted
2
1
4
3
0
2
4
3
1
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Feature Importance
fiber
47s
completed at 3:07 PM


Welcome To Colab - Colab
Untitled10.ipynb - Colab
colab.research.google.com/drive/1ae5kMKmbOKybw06olUluOOD58vTV8WEb#scrollTo=r40cR8YCsWzF
Untitled10.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
RAM
Disk
fruit
leafy
root
Predicted
Feature Importance
fiber
vitamin_a
vitamin_c
Feature
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Importance
Example Classification:
Classification Result:
Predicted Type: fruit
47s
completed at 3:07 PM

Welcome To Colab - Colab | Untitled10.ipynb - Colab

colab.research.google.com/drive/1ae5kMKmbOKybw06olUIuOOD58vTV8WEb#scrollTo=r40cR8YCsWzF

Untitled10.ipynb

File Edit View Insert Runtime Tools Help

Share | Gemini

Commands | + Code | + Text

RAM Disk

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.model_selection import GridSearchCV

# Load the dataset
from google.colab import files
uploaded = files.upload()

# Step 3: Load Dataset
filename = list(uploaded.keys())[0]
df = pd.read_csv('vegetables.csv')

# 1. Exploratory Data Analysis
print("Dataset Info:")
print(df.info())
print("\nClass Distribution:")
print(df['type'].value_counts())

# Visualize the data
plt.figure(figsize=(15, 10))

plt.subplot(2, 2, 1)
sns.boxplot(x='type', y='vitamin_a', data=df)
plt.title('Vitamin A by Vegetable Type')

plt.subplot(2, 2, 2)
sns.boxplot(x='type', y='vitamin_c', data=df)
plt.title('Vitamin C by Vegetable Type')

plt.subplot(2, 2, 3)
sns.boxplot(x='type', y='fiber', data=df)
```

47s completed at 3:07 PM

Welcome To Colab - Colab | Untitled10.ipynb - Colab

colab.research.google.com/drive/1ae5kMKmbOKybw06olUIuOOD58vTV8WEb#scrollTo=r40cR8YCsWzF

Untitled10.ipynb

File Edit View Insert Runtime Tools Help

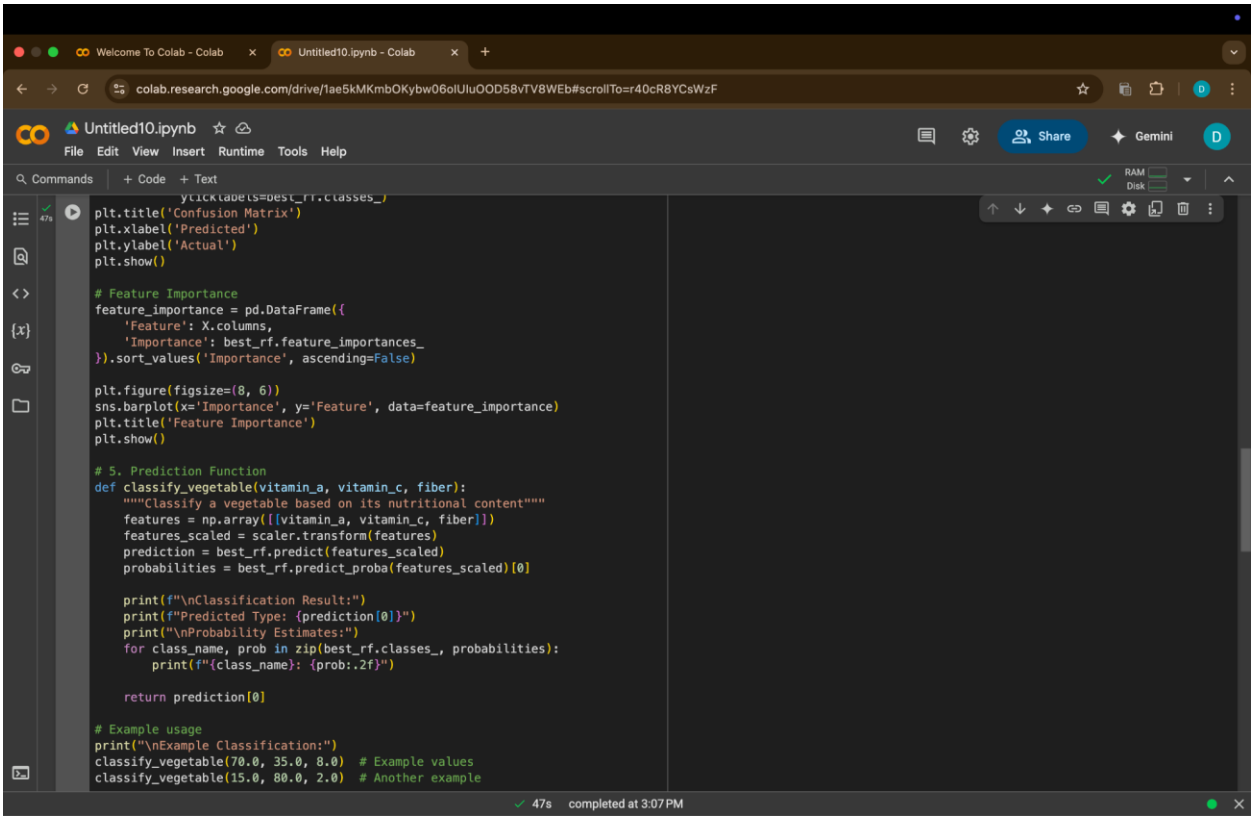
Share | Gemini

Commands | + Code | + Text

RAM Disk

```
plt.title('Vitamin C by Vegetable Type')

plt.subplot(2, 2, 3)
sns.boxplot(x='type', y='fiber', data=df)
plt.title('Fiber by Vegetable Type')

plt.subplot(2, 2, 4)
sns.scatterplot(x='vitamin_a', y='vitamin_c', hue='type', data=df)
plt.title('Vitamin A vs Vitamin C')

plt.tight_layout()
plt.show()

# 2. Data Preparation
X = df[['vitamin_a', 'vitamin_c', 'fiber']]
y = df['type']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 3. Model Building and Training
# Initialize Random Forest Classifier
rf = RandomForestClassifier(random_state=42)

# Hyperparameter tuning
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10]
}

grid_search = GridSearchCV(rf, param_grid, cv=5, scoring='accuracy')
```

47s completed at 3:07 PM


Welcome To Colab - Colab
Untitled10.ipynb - Colab
colab.research.google.com/drive/1ae5kMKmbOKybw06olUluOOD58vTV8WEb#scrollTo=r40cR8YCsWzF
Untitled10.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
RAM
Disk
yticklabels=best_rf.classes_)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
# Feature Importance
feature_importance = pd.DataFrame({
'Feature': X.columns,
'Importance': best_rf.feature_importances_
}).sort_values('Importance', ascending=False)
plt.figure(figsize=(8, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance)
plt.title('Feature Importance')
plt.show()
# 5. Prediction Function
def classify_vegetable(vitamin_a, vitamin_c, fiber):
"""Classify a vegetable based on its nutritional content"""
features = np.array([[vitamin_a, vitamin_c, fiber]])
features_scaled = scaler.transform(features)
prediction = best_rf.predict(features_scaled)
probabilities = best_rf.predict_proba(features_scaled)[0]
print(f"\nClassification Result:")
print(f"Predicted Type: {prediction[0]}")
print("\nProbability Estimates:")
for class_name, prob in zip(best_rf.classes_, probabilities):
print(f"{class_name}: {prob:.2f}")
return prediction[0]
# Example usage
print("\nExample Classification:")
classify_vegetable(70.0, 35.0, 8.0) # Example values
classify_vegetable(15.0, 80.0, 2.0) # Another example
47s completed at 3:07 PM