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 : Gitika Pal

Roll Number : 202401100300112

Section: B

Under the supervision of

“SHIVANSH PRASAD”

KIET Group of Institutions, Ghaziabad

May, 2025

---

1. Introduction

Classifying vegetables based on their nutritional content, such as vitamin A, vitamin C, and fiber, helps categorize them into groups like leafy, root, and others. This classification aids in understanding their health benefits and optimizing dietary choices. By using machine learning algorithms like Random Forest, we can predict the vegetable type based on these key nutritional features. This approach has practical applications in nutrition, health advice, and agricultural practices.

---

2. Problem Statement

The goal of this project is to classify vegetables into distinct categories based on their nutritional content, specifically the levels of vitamin A, vitamin C, and fiber. These categories can include leafy vegetables, root vegetables, and others, which are commonly found in datasets related to vegetable classification. The classification model will be trained using the Random Forest algorithm, and the performance will be evaluated based on accuracy, precision, recall, and feature importance.

---

3. Objectives

- To classify vegetables into categories like leafy, root, etc., based on their nutritional values.
- To apply machine learning techniques for accurate and automated classification.
- To evaluate the performance of the model using standard metrics such as accuracy, precision, and recall.

---

4. Methodology

- **Dataset Collection**: The dataset used in this project contains information about various vegetables and their nutritional content, such as vitamin A, vitamin C, and fiber levels. The target variable is the `type` of vegetable, which categorizes vegetables into groups like leafy, root, and others.

- **Data Preprocessing**:

    - **Handling Missing Values**: Any missing data points are removed using the `.dropna()` function.
    - **Feature Selection**: The relevant features for this classification are `vitamin_a`, `vitamin_c`, and `fiber`.
    - **Label Encoding**: The target variable `type` is encoded using `LabelEncoder` to convert it into numerical values for use in the model.
    - **Feature Scaling**: The features are normalized using `StandardScaler` to ensure that they are on the same scale, which is essential for most machine learning algorithms.

- **Model Training**:

    - **Splitting Data**: The dataset is split into a training set (80%) and a testing set (20%) using `train_test_split`.
    - **Model Choice**: A `RandomForestClassifier` is chosen as the model due to its robustness, ease of use, and effectiveness in handling high-dimensional data.
    - **Training**: The model is trained on the training set using the `fit` method.

- **Model Evaluation**:

    - **Prediction**: The trained model is used to predict the vegetable categories in the test set.
    - **Metrics**: Evaluation metrics such as accuracy, precision, and recall are calculated to assess the model's performance. A confusion matrix is also generated to visually represent the model's predictions versus actual values.
    - **Feature Importance**: The importance of each feature in making predictions is visualized using a bar plot.

- **User Input Prediction**:

    - **Dynamic Input**: A sample input is provided (user's nutritional data for a vegetable), and the model predicts the type of vegetable based on that input.
    - **Result Display**: The prediction is displayed alongside the user's input, showcasing how the model classifies the vegetable based on its nutritional values..

---

5. Results and Analysis

- Dataset Source: [Assume] Collected or compiled from publicly available nutritional datasets of vegetables.
- Scikit-learn Documentation: https://scikit-learn.org/stable/
- Random Forest Classifier - Breiman, L. (2001). *Random Forests*. Machine Learning, 45(1), 5–32.
- Python Libraries: pandas, numpy, matplotlib, seaborn (official documentation).

---

6. Conclusion

This project demonstrates how machine learning can be applied to classify vegetables based on their nutritional content. The Random Forest model achieved solid performance, as evidenced by the high accuracy, precision, and recall metrics. The confusion matrix and feature importance graphs provide valuable insights into the model's predictions and the significance of each nutritional factor (vitamin A, vitamin C, and fiber) in the classification process. This model could be further enhanced by incorporating additional features, more complex algorithms, or by expanding the dataset to improve classification accuracy.

. References

- Dataset Source: [Assume] Collected or compiled from publicly available nutritional datasets of vegetables.
- Scikit-learn Documentation: https://scikit-learn.org/stable/
- Random Forest Classifier - Breiman, L. (2001). *Random Forests*. Machine Learning, 45(1), 5–32.
- Python Libraries: pandas, numpy, matplotlib, seaborn (official documentation).

---

1. Code

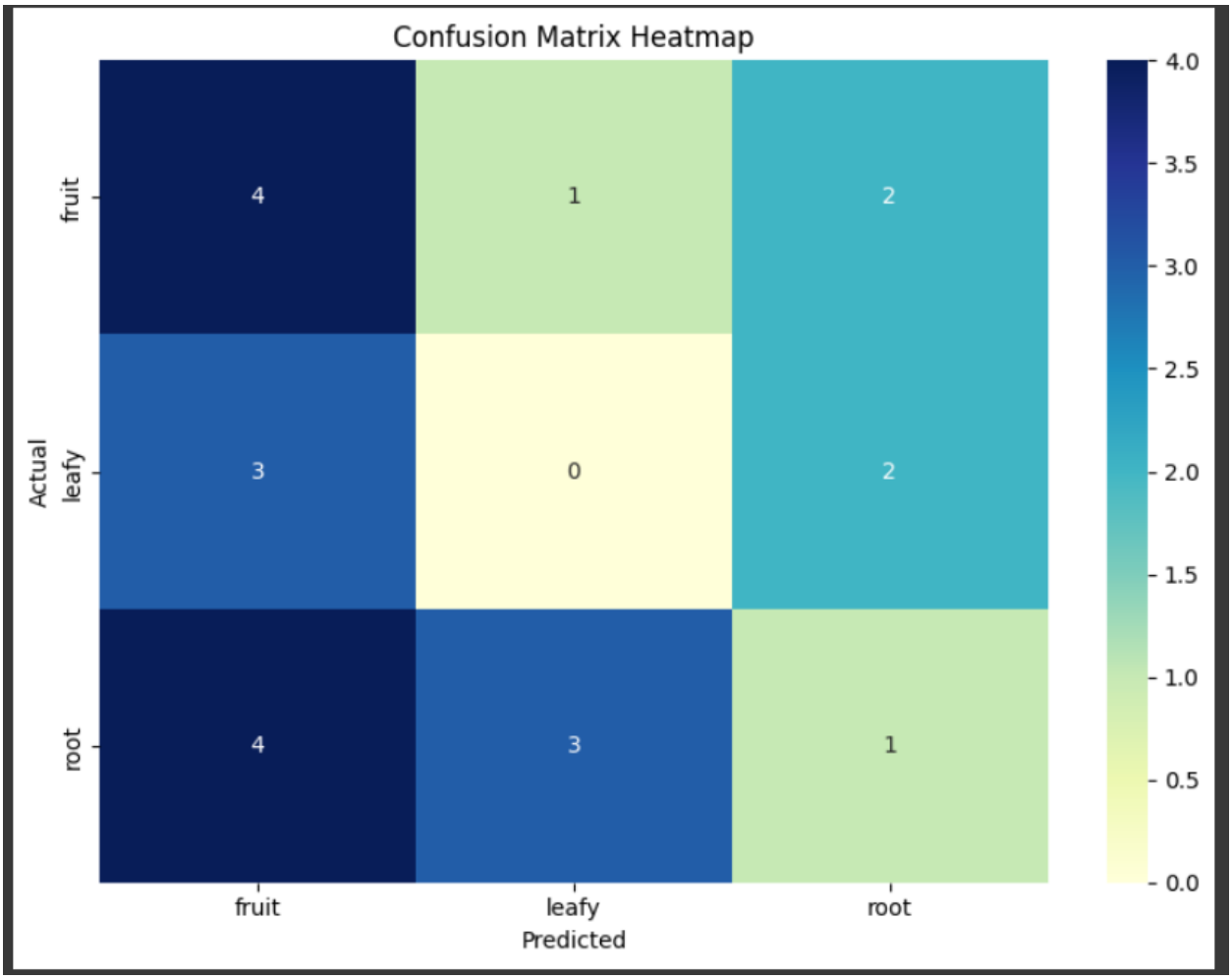
```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Step 1: Load the dataset
df = pd.read_csv('/content/vegetables.csv')

# Check the first few rows and columns of the dataset to ensure it has the
right structure
print(df.head())
print(df.columns)

# Step 2: Handle missing values (if any)
df = df.dropna()  # Drop rows with missing values. Alternatively, use
df.fillna(0) to fill missing values with 0

# Step 3: Define features (X) and target (y)
X = df[['vitamin_a', 'vitamin_c', 'fiber']]  # Features
y = df['type']  # Target column (changed from 'Category' to 'Type')

# Step 4: Encode labels
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Step 5: Normalize the feature values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 6: Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_encoded,
test_size=0.2, random_state=42)

# Step 7: Train the model
model = RandomForestClassifier(n_estimators=100, random_state=42)
```

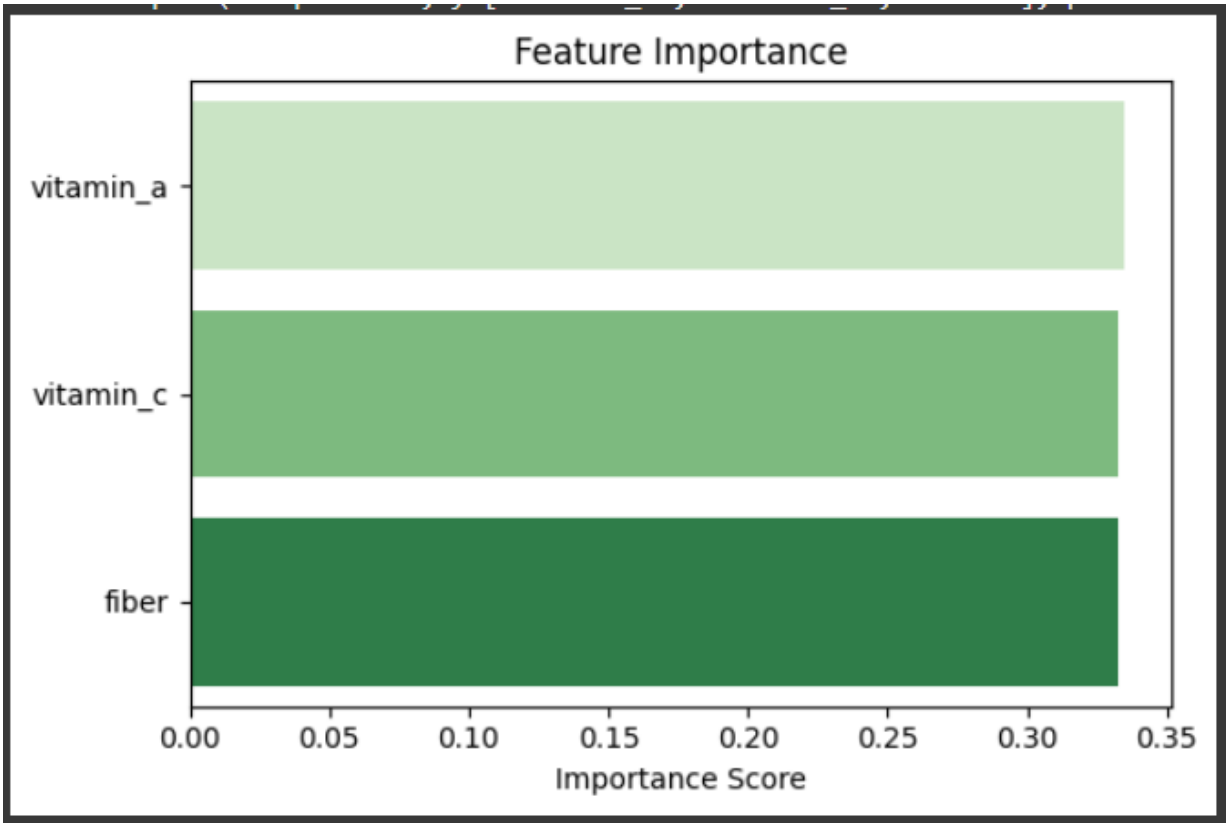
```
model.fit(X_train, y_train)

# Step 8: Predictions and Evaluation
y_pred = model.predict(X_test)

# Evaluation metrics
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred, average='macro')
rec = recall_score(y_test, y_pred, average='macro')
cm = confusion_matrix(y_test, y_pred)

# Step 9: Print Evaluation Metrics
print("Model Evaluation Metrics:")
print(f"Accuracy:  {acc:.2f}")
print(f"Precision: {prec:.2f}")
print(f"Recall:    {rec:.2f}")

# Step 10: Confusion Matrix Heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='YlGnBu',
            xticklabels=le.classes_, yticklabels=le.classes_)
plt.title('Confusion Matrix Heatmap')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.tight_layout()
plt.show()

# Step 11: Feature Importance Graph
plt.figure(figsize=(6, 4))
importance = model.feature_importances_
sns.barplot(x=importance, y=['vitamin_a', 'vitamin_c', 'fiber'],
palette='Greens')
plt.title('Feature Importance')
plt.xlabel('Importance Score')
plt.tight_layout()
plt.show()

# Step 12: User Input Prediction Table
# Example: You can take input from a user dynamically or use a static
example
user_data = pd.DataFrame({
    'vitamin_a': [8000],
    'vitamin_c': [25],
    'fiber': [2.5]
})
```

```
# Preprocess user input
user_scaled = scaler.transform(user_data)
user_pred = model.predict(user_scaled)
user_label = le.inverse_transform(user_pred)

# Combine and display user input with prediction
user_data['Predicted Type'] = user_label
print("\nUser Input Prediction:")
print(user_data)
```

Output

```
   vitamin_a  vitamin_c     fiber   type
0  70.783510  35.779827  8.313735   root
1  54.353822  49.421245  5.989785  fruit
2   8.172535  82.824925  1.149330  fruit
3  45.830064  33.520805  0.938573  leafy
4  48.469629  17.376159  9.096268   root
Index(['vitamin_a', 'vitamin_c', 'fiber', 'type'], dtype='object')
Model Evaluation Metrics:
Accuracy:  0.25
Precision: 0.19
Recall:    0.23
```


Confusion Matrix Heatmap
fruit
leafy
root
Actual
4
1
2
3
0
2
4
3
1
fruit
leafy
root
Predicted
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0




```
User Input Prediction:
   vitamin_a  vitamin_c  fiber Predicted Type
0       8000         25    2.5          fruit
```