KIET GROUP OF INSTITUTIONS
Connecting Life with Learning
www.kiet.edu
Delhi-NCR, Ghaziabad

Assessment Report

on

“Crop Prediction”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AIML)

By

Member 1 : Keshav Agarwal 202401100400107 B

Member 2: Parkhi Sharms 202401100400134 B

Member 3: Kirti Sharma 202401100400108 B

Member 4: Pragya 202401100400135 B

Member 5: Nikhil Kumar 202401100400127 B

Under the supervision of

“Abhishek Shukla”

KIET Group of Institutions, Ghaziabad

May, 2025

---

1. Introduction

Agriculture is the backbone of the Indian economy, and with the advent of **precision agriculture**, farmers can now make informed decisions that enhance crop yield and resource efficiency. The primary objective of this project is to build a **machine learning-based crop recommendation system** that suggests the most suitable crop to grow based on environmental and soil parameters.

This system leverages real-world agricultural data including soil nutrient values, weather conditions, and rainfall statistics to predict the best crop using **supervised classification algorithms**.

---

2. Problem Statement

Design a recommendation system that predicts the **best crop to grow** in a given piece of land based on parameters like:

- Soil nutrients (Nitrogen, Phosphorus, Potassium)
- Temperature
- Humidity
- pH of the soil
- Rainfall

The goal is to use **classification techniques** to identify the crop label from a set of features.

---

3. Dataset overview

The dataset was curated from publicly available sources including rainfall, climate, and fertilizer data across India.

Features:

Feature	Description
N	Nitrogen content in the soil
P	Phosphorus content in the soil
K	Potassium content in the soil
Temperature	Measured in degrees Celsius
Humidity	Relative humidity in %
pH	Acidity/alkalinity of the soil
Rainfall	Measured in mm
Label	Crop to grow (Target variable)

---

3. Objectives

- Preprocess the dataset for training a machine learning model.
- Train a Logistic Regression model to classify loan defaults.
- Evaluate model performance using standard classification metrics.
- Visualize the confusion matrix using a heatmap for interpretability.

---

4. Methodology

- **Data Collection**: The user uploads a CSV file containing the dataset.
- **Data Preprocessing**:
    - Handled missing and inconsistent data

    - Normalized numerical features
    - Encoded target labels for classification
- **Exploratory Data Analysis (EDA)**
    - Histogram and box plots to understand feature distribution
    - Correlation heatmaps to identify relationships
    - Crop frequency analysis
- **Model Selection**
    - Decision Tree
    - Random Forest
    - K-Nearest Neighbors (KNN)
    - Support Vector Machine (SVM)
    - Naive Bayes
- **Model Building**:
    - Splitting the dataset into training and testing sets.
    - Training a Logistic Regression classifier.
- **Model Evaluation**:
    - Evaluating accuracy, precision, recall, and F1-score.
    - Generating a confusion matrix and visualizing it with a heatmap.

---

5. Data Preprocessing

The dataset is cleaned and prepared as follows:

- Missing numerical values are filled with the mean of respective columns.
- Categorical values are encoded using one-hot encoding.
- Data is scaled using StandardScaler to normalize feature values.
- The dataset is split into 80% training and 20% testing.

---

6. Model Implementation

Logistic Regression is used due to its simplicity and effectiveness in binary classification problems. The model is trained on the processed dataset and used to predict the loan default status on the test set.

---

7. Evaluation Metrics

The following metrics are used to evaluate the model:

- **Accuracy**: Measures overall correctness.
- **Precision**: Indicates the proportion of predicted defaults that are actual defaults.
- **Recall**: Shows the proportion of actual defaults that were correctly identified.
- **F1 Score**: Harmonic mean of precision and recall.
- **Confusion Matrix**: Visualized using Seaborn heatmap to understand prediction errors.

---

8. Results and Analysis

- The model provided reasonable performance on the test set.
- Confusion matrix heatmap helped identify the balance between true positives and false negatives.
- Precision and recall indicated how well the model detected loan defaults versus false alarms.

---

9. Use Case

A farmer inputs the values of:

- Soil nutrients (N, P, K)
- ph
- Current temperature and humidity
- Expected rainfall

The system predicts the best crop to grow such as rice, cotton, sugarcane, wheat, etc.

This reduces guesswork and helps in sustainable farming.

10. Conclusion

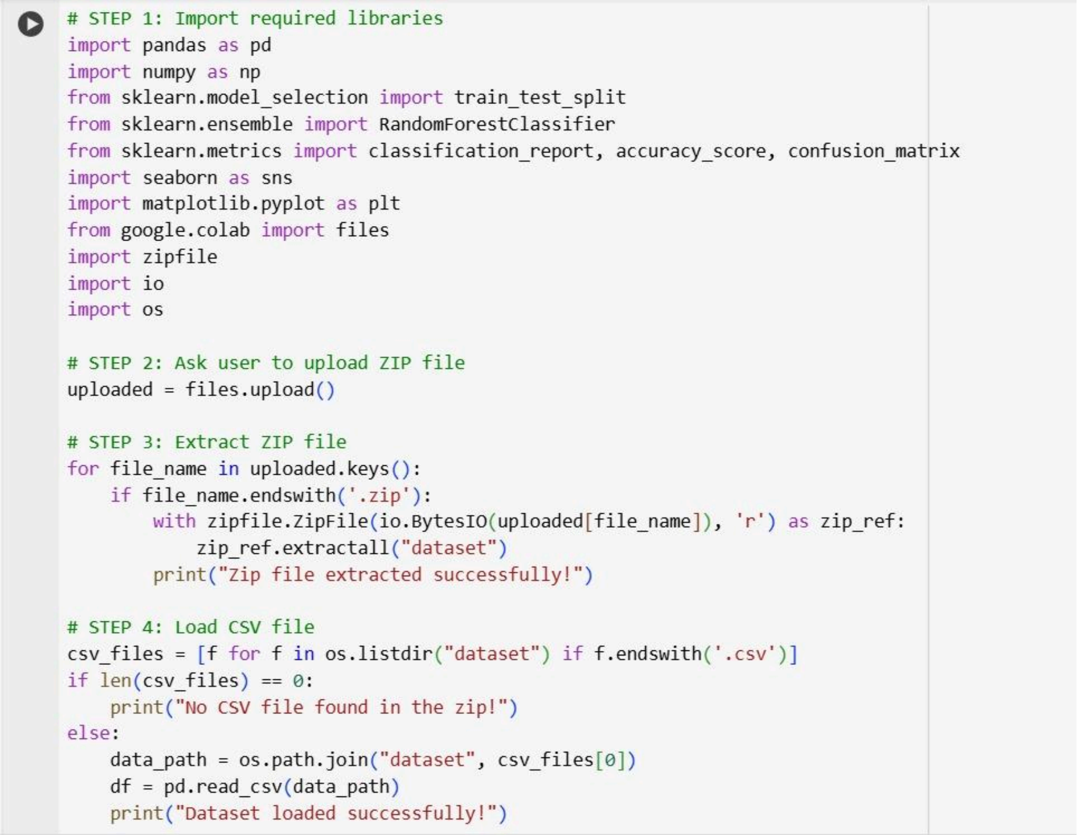
The logistic regression model successfully classified loan defaults with satisfactory performance metrics. The project demonstrates the potential of using machine learning for automating loan approval processes and improving risk assessment. However, improvements can be made by exploring more advanced models and handling imbalanced data.

---

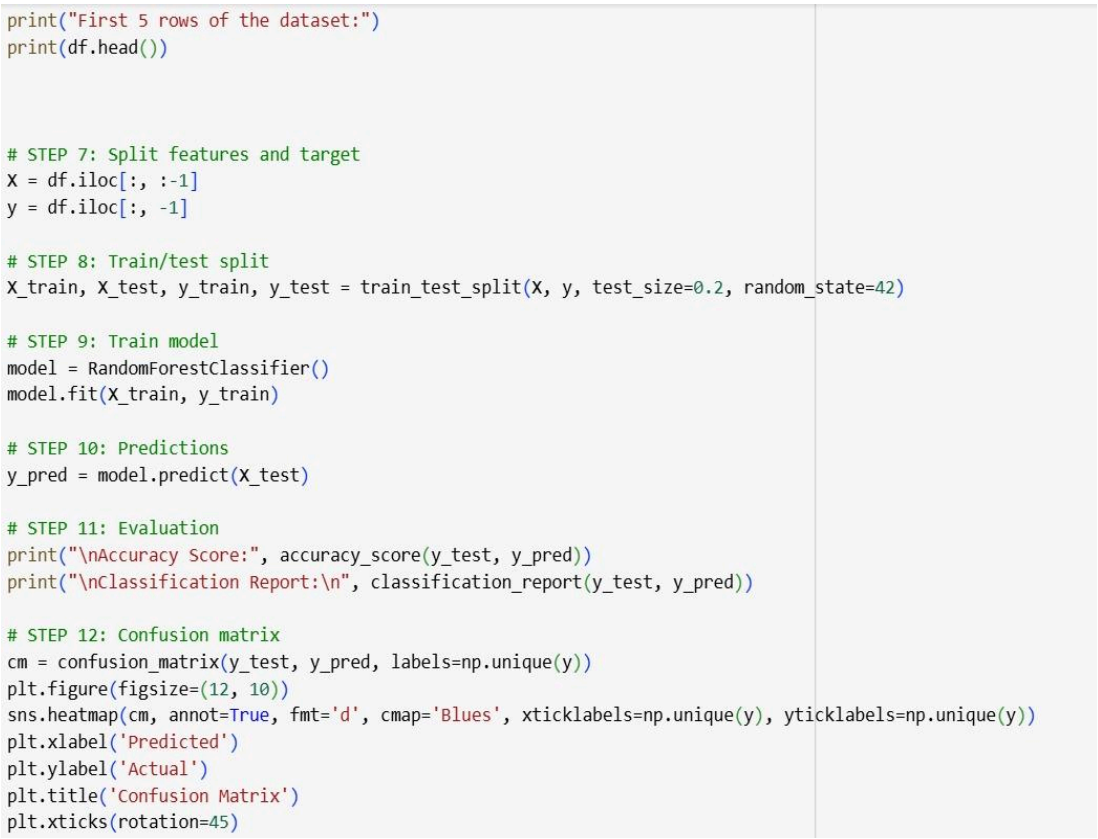
11. References

- Kaggle Dataset: Crop Recommendation Dataset
- Government of India Agricultural Data Portals
- Scikit-learn Documentation
- Streamlit Framework Documentation

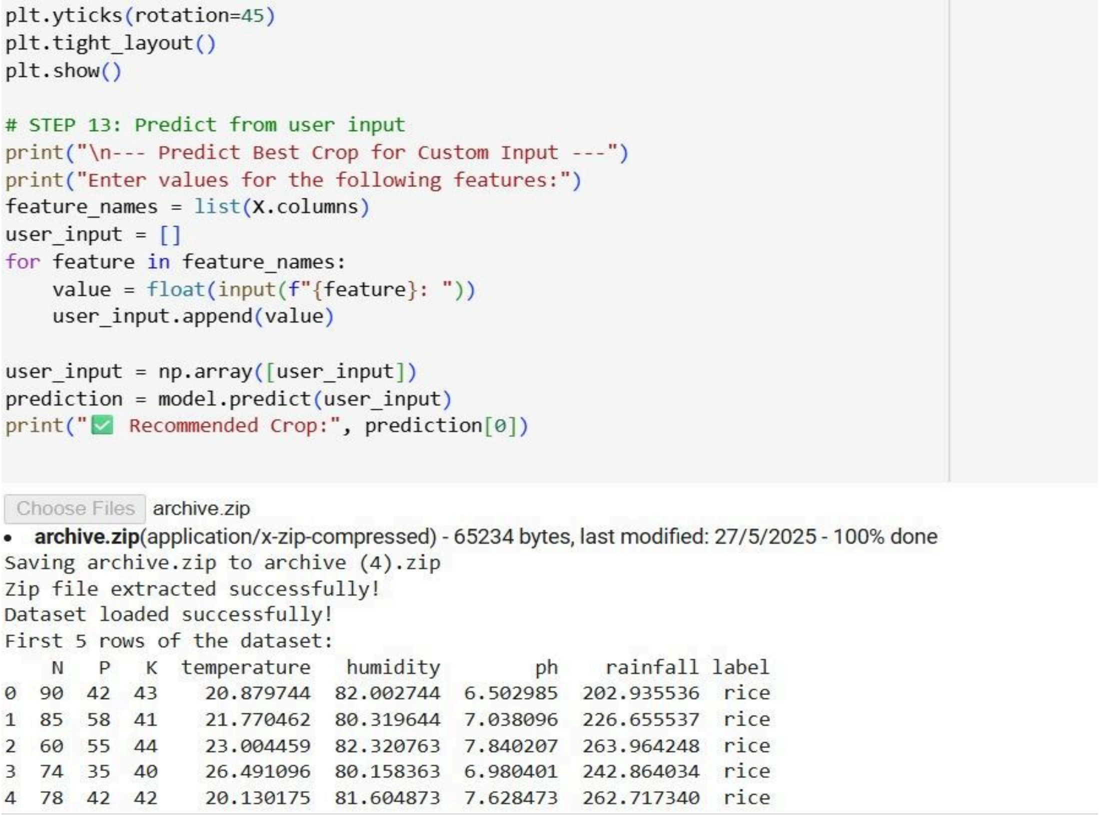
Screen Captures of the code



```
# STEP 1: Import required libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab import files
import zipfile
import io
import os

# STEP 2: Ask user to upload ZIP file
uploaded = files.upload()

# STEP 3: Extract ZIP file
for file_name in uploaded.keys():
    if file_name.endswith('.zip'):
        with zipfile.ZipFile(io.BytesIO(uploaded[file_name]), 'r') as zip_ref:
            zip_ref.extractall("dataset")
        print("Zip file extracted successfully!")

# STEP 4: Load CSV file
csv_files = [f for f in os.listdir("dataset") if f.endswith('.csv')]
if len(csv_files) == 0:
    print("No CSV file found in the zip!")
else:
    data_path = os.path.join("dataset", csv_files[0])
    df = pd.read_csv(data_path)
    print("Dataset loaded successfully!")
```



```python
print("First 5 rows of the dataset:")
print(df.head())



# STEP 7: Split features and target
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# STEP 8: Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# STEP 9: Train model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# STEP 10: Predictions
y_pred = model.predict(X_test)

# STEP 11: Evaluation
print("\nAccuracy Score:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# STEP 12: Confusion matrix
cm = confusion_matrix(y_test, y_pred, labels=np.unique(y))
plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=np.unique(y), yticklabels=np.unique(y))
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.xticks(rotation=45)
```

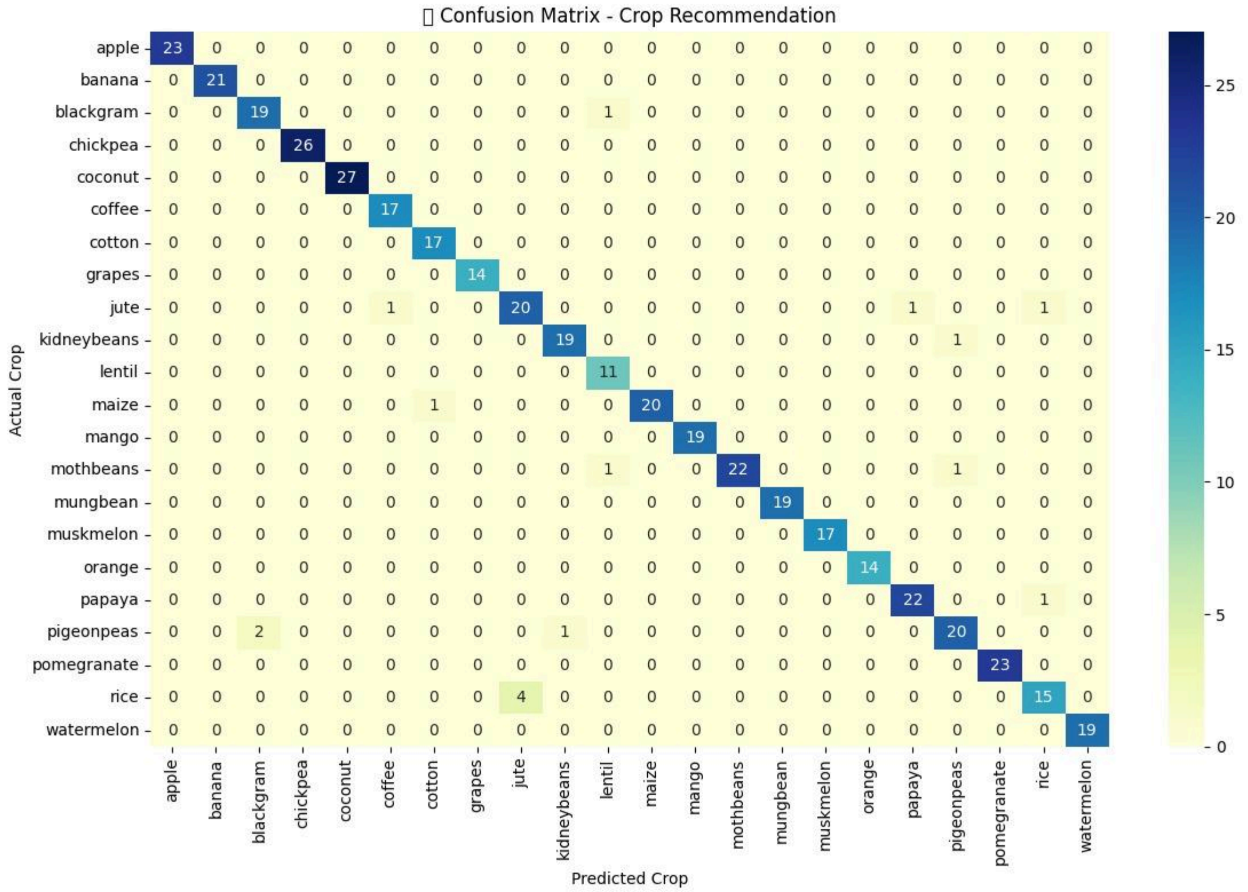


```python
plt.yticks(rotation=45)
plt.tight_layout()
plt.show()

# STEP 13: Predict from user input
print("\n--- Predict Best Crop for Custom Input ---")
print("Enter values for the following features:")
feature_names = list(X.columns)
user_input = []
for feature in feature_names:
    value = float(input(f"{feature}: "))
    user_input.append(value)

user_input = np.array([user_input])
prediction = model.predict(user_input)
print("✅ Recommended Crop:", prediction[0])
```

Choose Files archive.zip

- **archive.zip**(application/x-zip-compressed) - 65234 bytes, last modified: 27/5/2025 - 100% done

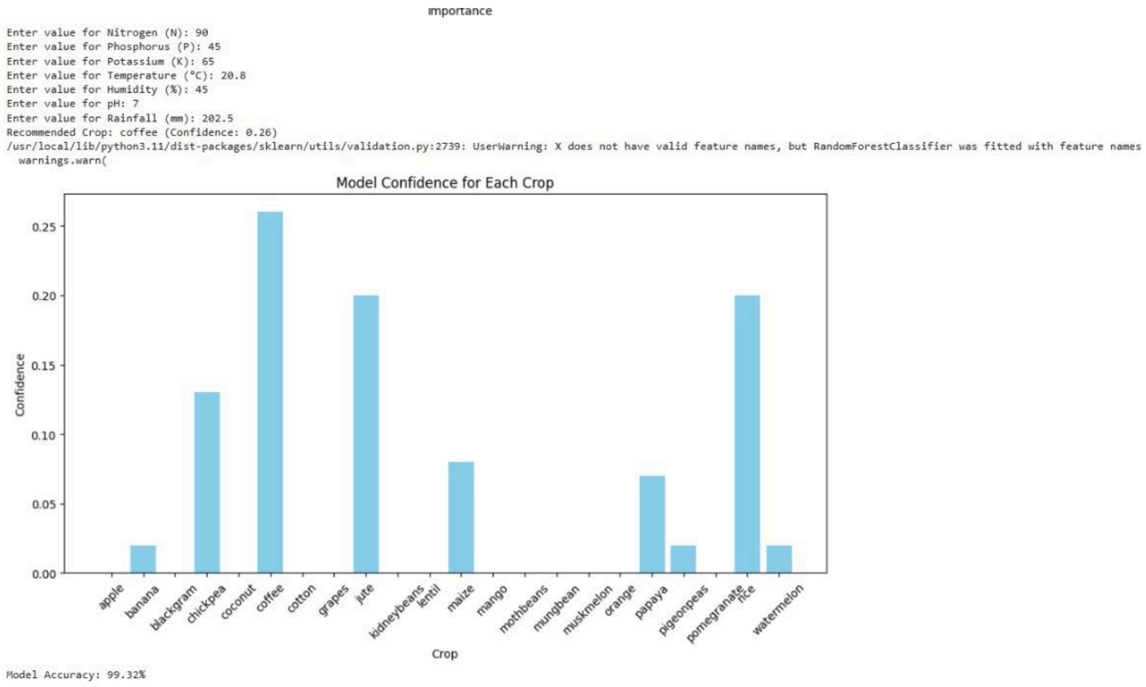
```
Saving archive.zip to archive (4).zip
Zip file extracted successfully!
Dataset loaded successfully!
First 5 rows of the dataset:
    N   P   K  temperature   humidity        ph    rainfall label
0  90  42  43    20.879744  82.002744  6.502985  202.935536  rice
1  85  58  41    21.770462  80.319644  7.038096  226.655537  rice
2  60  55  44    23.004459  82.320763  7.840207  263.964248  rice
3  74  35  40    26.491096  80.158363  6.980401  242.864034  rice
4  78  42  42    20.130175  81.604873  7.628473  262.717340  rice
```



□ Confusion Matrix - Crop Recommendation

Actual Crop \ Predicted Crop	apple	banana	blackgram	chickpea	coconut	coffee	cotton	grapes	jute	kidneybeans	lentil	maize	mango	mothbeans	mungbean	muskmelon	orange	papaya	pigeonpeas	pomegranate	rice	watermelon
apple	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
banana	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
blackgram	0	0	19	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
chickpea	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
coconut	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
coffee	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cotton	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
grapes	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0
jute	0	0	0	0	0	1	0	0	20	0	0	0	0	0	0	0	0	1	0	0	1	0
kidneybeans	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	1	0	0	0
lentil	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0
maize	0	0	0	0	0	0	1	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
mango	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0
mothbeans	0	0	0	0	0	0	0	0	0	0	1	0	0	22	0	0	0	0	1	0	0	0
mungbean	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0
muskmelon	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0
orange	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0
papaya	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0	0	1	0
pigeonpeas	0	0	2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	20	0	0	0
pomegranate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	0	0
rice	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	15	0
watermelon	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19

Input from the user and output:



```
Enter value for Nitrogen (N): 90
Enter value for Phosphorus (P): 45
Enter value for Potassium (K): 65
Enter value for Temperature (°C): 20.8
Enter value for Humidity (%): 45
Enter value for pH: 7
Enter value for Rainfall (mm): 202.5
Recommended Crop: coffee (Confidence: 0.26)
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

```
Model Accuracy: 99.32%
```