**ARMANDO MALUANA, MSISM**
GILBERT, AZ
(480) 745-4762 ▪ armmaluana@gmail.com ▪ LinkedIn

---

## Agricultural Production Optimization

**Problem Statement:** Construct a Predictive Machine Learning Model to enhance crop selection based on available climate and soil conditions.

**Goal:** Enhance Precision Farming practices by optimizing agricultural processes.

This Project Focuses on Advancing Precision Farming:

1- Maximizing Productivity
2- Deepening Understanding of Climate and Soil Requirements for Crops
3- Assisting Users in Dealing with Unpredictable Weather Patterns

**Tools:** The Machine Learning model was developed using Jupyter Notebook in Python. The toolkit includes essential libraries such as Pandas, NumPy, Matplotlib, Seaborn, and Interact.

The dataset encompasses a diverse range of crops, each with specific climate and soil preferences. I crafted feature vectors and divided the data into 80% for training and 20% for testing purposes. With the model now well-trained, users can input values for Nitrogen (N), Phosphorus (P), and Potassium (K), along with Temperature, Humidity, pH, and Rainfall.

Based on the input data, the model furnishes recommendations for the most suitable crops, accounting for the prevailing soil and climatic conditions.

**Final Outcome**:

```
In [189…
# Here input values to generate predictions
prediction = model.predict((np.array([[20,
                                       30,
                                       10,
                                       15,
                                       90,
                                       7.5,
                                       100]])))
print("The Sugested Crop for Given Climatic Condition is :", prediction)

The Sugested Crop for Given Climatic Condition is : ['orange']
```

**See Code Below:**

jupyter Agricultural Production Optimization Last Checkpoint: 25 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted

Python 3 (ipykernel)

Run Code

## Import Libraries

In [12]:

```
# for data manipulation
import numpy as np
import pandas as pd

#for data visualizations
import matplotlib.pyplot as plt
import seaborn as sns

#for interactivity
from ipywidgets import interact
```

In [15]:

```
#lets read the dataset
data = pd.read_csv("data.csv")
```

In [16]:

```
#Lets Check the shape of the dataset
print("shape of the Dataset:", data.shape)
```

```
shape of the Dataset: (2200, 8)
```

In [17]:

```
#Lets check the content of the dataset
data.head()
```

Out[17]:

| | 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 |

In [18]:

```
#lets check if there is any missing value in the dataset
#if there missing values you can use the fillin function. you can use:
  #Mean-numerical values,
  #Median - If the calues are the dataset contains large numbers of utliers.
  #Mode-If the values are categorrical
data.isnull().sum()
```

Out[18]:

```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

In [21]:

```
#Lets check the Crops present in the Dataset
data["label"].value_counts()
```

Out[21]:

```
rice           100
maize          100
jute           100
cotton         100
coconut        100
papaya         100
orange         100
apple          100
muskmelon      100
watermelon     100
grapes         100
mango          100
banana         100
pomegranate    100
lentil         100
blackgram      100
mungbean       100
mothbeans      100
pigeonpeas     100
```

In [35]:

```python
#Lets check the sumary for all the Crops
print("Average Ratio of Nitrogen in the Soil: {0:.2f}".format(data["N"].mean()))
print("Average Ratio of Phosphorous in the Soil: {0:.2f}".format(data["P"].mean()))
print("Average Ratio of Potassium in the Soil: {0:.2f}".format(data["K"].mean()))
print("Average Temperature in Celsius: {0:.2f}".format(data["temperature"].mean()))
print("Average Relative Humidity in %: {0:.2f}".format(data["humidity"].mean()))
print("Average PH of the Soil: {0:.2f}".format(data["ph"].mean()))
print("Average Rainfall in mm: {0:.2f}".format(data["rainfall"].mean()))
```

```
Average Ratio of Nitrogen in the Soil: 50.55
Average Ratio of Phosphorous in the Soil: 53.36
Average Ratio of Potassium in the Soil: 48.15
Average Temperature in Celsius: 25.62
Average Relative Humidity in %: 71.48
Average PH of the Soil: 6.47
Average Rainfall in mm: 103.46
```

In [65]:

```python
# Lets check the Summary Statistics for each of the Crops
#use the dropdown menu below to select individual crop
@interact
def summary(crops = list(data["label"].value_counts().index)):
    X = data[data["label"] == crops]
    print("---------------------------------------------")
    print("\033[1m" + "Statistics for Nitrogen" + "\033[0m")
    print("Minimum Nitrogen required :", X["N"].min())
    print("Minimum Nitrogen required :", X["N"].mean())
    print("Maximum Nitrogen required :", X["N"].max())

    print("---------------------------------------------")
    print("\033[1m" + "Statistics for Phosphorous" + "\033[0m")
    print("Minimum Phosphorous required :", X["P"].min())
    print("Minimum Phosphorous required :", X["P"].mean())
    print("Maximum Phosphorous required :", X["P"].max())

    print("---------------------------------------------")
    print("\033[1m" + "Statistics for Potassium" + "\033[0m")
    print("Minimum Potassium required :", X["K"].min())
```

```
interactive(children=(Dropdown(description='crops', options=('rice', 'maize', 'jute', 'cotton', 'coconut', 'pa…
```

In [72]:

```python
#lets compare the Average Requirement for each crops with average conditions
@interact
def compare (Conditions = ["N","P","K","temperature","humidity","ph","rainfall"]):
    print ("Average Value for", Conditions,"is {0:.2f}".format(data[Conditions].mean()))
    print("----------------------------------------------------------")
    print("Apple : {0:.2f}".format(data[(data["label"] == "apple")][Conditions].mean()))
    print("Banana : {0:.2f}".format(data[(data["label"] == "banana")][Conditions].mean()))
    print("Blackgram : {0:.2f}".format(data[(data["label"] == "blackgram")][Conditions].mean()))
    print("Chickpea : {0:.2f}".format(data[(data["label"] == "chickpea")][Conditions].mean()))
    print("Coconut : {0:.2f}".format(data[(data["label"] == "coconut")][Conditions].mean()))
    print("Coffee : {0:.2f}".format(data[(data["label"] == "coffee")][Conditions].mean()))
    print("Cotton : {0:.2f}".format(data[(data["label"] == "cotton")][Conditions].mean()))
    print("Grapes : {0:.2f}".format(data[(data["label"] == "grapes")][Conditions].mean()))
    print("Jute : {0:.2f}".format(data[(data["label"] == "jute")][Conditions].mean()))
    print("Kidneybeans : {0:.2f}".format(data[(data["label"] == "kidneybeans")][Conditions].mean()))
    print("Lentil : {0:.2f}".format(data[(data["label"] == "lentil")][Conditions].mean()))
    print("Maize : {0:.2f}".format(data[(data["label"] == "maize")][Conditions].mean()))
    print("Mango : {0:.2f}".format(data[(data["label"] == "mango")][Conditions].mean()))
    print("Mothbeans : {0:.2f}".format(data[(data["label"] == "mothbeans")][Conditions].mean()))
    print("mungbean : {0:.2f}".format(data[(data["label"] == "mungbean")][Conditions].mean()))
    print("Muskmelon : {0:.2f}".format(data[(data["label"] == "muskmelon")][Conditions].mean()))
    print("Orange : {0:.2f}".format(data[(data["label"] == "orange")][Conditions].mean()))
    print("Papaya : {0:.2f}".format(data[(data["label"] == "papaya")][Conditions].mean()))
    print("Pigeonpeas : {0:.2f}".format(data[(data["label"] == "pigeonpeas")][Conditions].mean()))
    print("Pomegranate : {0:.2f}".format(data[(data["label"] == "pomegranate")][Conditions].mean()))
    print("Rice : {0:.2f}".format(data[(data["label"] == "rice")][Conditions].mean()))
    print("Watermelon : {0:.2f}".format(data[(data["label"] == "watermelon")][Conditions].mean()))
```

```
interactive(children=(Dropdown(description='Conditions', options=('N', 'P', 'K', 'temperature', 'humidity', 'p…
```

```python
    print("\033[1m" + "Statistics for Rainfall" + "\033[0m")
    print("Minimum Rainfall required :", X["rainfall"].min())
    print("Minimum Rainfall required :", X["rainfall"].mean())
    print("Maximum Rainfall required :", X["rainfall"].max())
```

crops: rice

rice
maize
jute
cotton
coconut
papaya
orange
apple
muskmelon
watermelon
grapes
mango
banana
pomegranate
lentil
blackgram
mungbean
mothbeans
pigeonpeas

```
---------- --------
Statistics
Minimum Nit
Minimum Nit
Maximum Nit
---------- --------
Statistics
Minimum Pho
Minimum Pho
Maximum Pho
---------- --------
Statistics
Minimum Pot
Minimum Pot
Maximum Pot
---------- --------
Statistics
Minimum Tem 142
Minimum Tem 32210500005
Maximum Tem 5077
---------- --------
Statistics for Humidity
Minimum Humidity required : 80.12267476
Minimum Humidity required : 82.2728215389
Maximum Humidity required : 84.96907151
---------------------------------------------
Statistics for PH
Minimum PH required : 5.005306977
Minimum PH required : 6.425470922139999
Maximum PH required : 7.868474653
---------------------------------------------
```

In [12]:

```python
#lets compare the Average Requirement for each crops with average conditions
@interact
def compare (Conditions = ["N","P","K","temperature","humidity","ph","rainfall"]):
    print ("Average Value for", Conditions,"is {0:.2f}".format(data[Conditions].mean()))
    print("---------------------------------------------------------")
    print("Apple : {0:.2f}".format(data[(data["label"] == "apple")][Conditions].mean()))
    print("Banana : {0:.2f}".format(data[(data["label"] == "banana")][Conditions].mean()))
    print("Blackgram : {0:.2f}".format(data[(data["label"] == "blackgram")][Conditions].mean()))
    print("Chickpea : {0:.2f}".format(data[(data["label"] == "chickpea")][Conditions].mean()))
    print("Coconut : {0:.2f}".format(data[(data["label"] == "coconut")][Conditions].mean()))
    print("Coffee : {0:.2f}".format(data[(data["label"] == "coffee")][Conditions].mean()))
    print("Cotton : {0:.2f}".format(data[(data["label"] == "cotton")][Conditions].mean()))
    print("Grapes : {0:.2f}".format(data[(data["label"] == "grapes")][Conditions].mean()))
    print("Jute : {0:.2f}".format(data[(data["label"] == "jute")][Conditions].mean()))
    print("Kidneybeans : {0:.2f}".format(data[(data["label"] == "kidneybeans")][Conditions].mean()))
    print("Lentil : {0:.2f}".format(data[(data["label"] == "lentil")][Conditions].mean()))
    print("Maize : {0:.2f}".format(data[(data["label"] == "maize")][Conditions].mean()))
    print("Mango : {0:.2f}".format(data[(data["label"] == "mango")][Conditions].mean()))
    print("Mothbeans : {0:.2f}".format(data[(data["label"] == "mothbeans")][Conditions].mean()))
    print("mungbean : {0:.2f}".format(data[(data["label"] == "mungbean")][Conditions].mean()))
    print("Muskmelon : {0:.2f}".format(data[(data["label"] == "muskmelon")][Conditions].mean()))
    print("Orange : {0:.2f}".format(data[(data["label"] == "orange")][Conditions].mean()))
    print("Papaya : {0:.2f}".format(data[(data["label"] == "papaya")][Conditions].mean()))
    print("Pigeonpeas : {0:.2f}".format(data[(data["label"] == "pigeonpeas")][Conditions].mean()))
    print("P t(data[(data["label"] == "pomegranate")][Conditions].mean()))
    print("R (data["label"] == "rice")][Conditions].mean()))
    print("W (data[(data["label"] == "watermelon")][Conditions].mean()))
```

N
P
K
temperature
humidity
ph
rainfall

Conditions: temperature

```
Average Value for temperature is 25.62
---------------------------------------------------------
Apple : 22.63
Banana : 27.38
Blackgram : 29.97
```

In [13]:

```python
#lets make this function more interactive
@interact
def compare (Conditions = ["N","P","K","temperature","humidity","ph","rainfall"]):
    print("\033[1m" +"Crops which require greater than average", Conditions,"\n" + "\033[0m")
    print(data[data[Conditions] > data[Conditions].mean()]["label"].unique())
    print('---------------------------------------------------------------')
    print("\033[1m" +"Crops which require less than average", Conditions,"\n" + "\033[0m")
    print(data[data[Conditions] <= data[Conditions].mean()]["label"].unique())
```

Conditions temperature

**Crops which require greater than average temperature**

```
['rice' 'maize' 'pigeonpeas' 'mothbeans' 'mungbean' 'blackgram' 'lentil'
 'banana' 'mango' 'grapes' 'watermelon' 'muskmelon' 'orange' 'papaya'
 'coconut' 'cotton' 'jute' 'coffee']
---------------------------------------------------------------
```

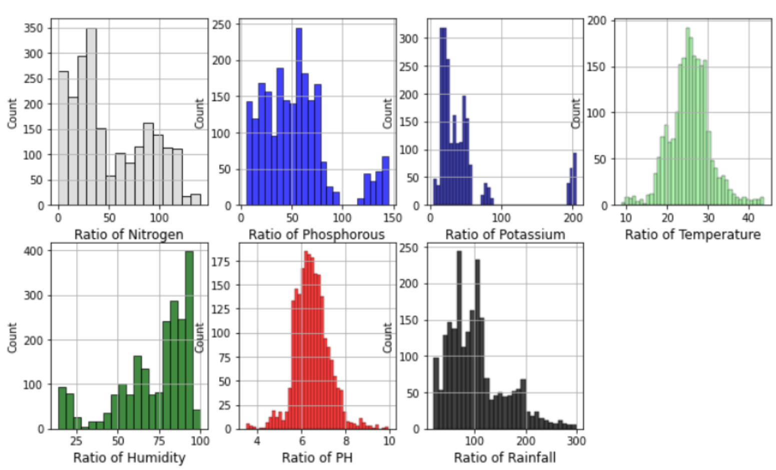
**Crops which require less than average temperature**

```
['rice' 'maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans'
 'blackgram' 'lentil' 'pomegranate' 'banana' 'grapes' 'watermelon' 'apple'
 'orange' 'papaya' 'coconut' 'cotton' 'jute' 'coffee']
```

## Distribution

In [117]:

```python
## Lets check the Distribution of Agricultural Conditions of dataset
plt.rcParams["figure.figsize"] = (12,7)

plt.subplot(2, 4, 1)
sns.histplot(data["N"], color = "lightgrey")
plt.xlabel("Ratio of Nitrogen", fontsize = 12)
plt.grid()

plt.subplot(2, 4, 2)
sns.histplot(data["P"], color = "blue")
plt.xlabel("Ratio of Phosphorous", fontsize = 12)
plt.grid()

plt.subplot(2, 4, 3)
sns.histplot(data["K"], color = "darkblue")
plt.xlabel("Ratio of Potassium", fontsize = 12)
plt.grid()

plt.subplot(2, 4, 4)
sns.histplot(data["temperature"], color = "lightgreen")
plt.xlabel("Ratio of Temperature", fontsize = 12)
plt.grid()

plt.subplot(2, 4, 5)
sns.histplot(data["humidity"], color = "darkgreen")
plt.xlabel("Ratio of Humidity", fontsize = 12)
plt.grid()

plt.subplot(2, 4, 6)
sns.histplot(data["ph"], color = "red")
plt.xlabel("Ratio of PH", fontsize = 12)
plt.grid()

plt.subplot(2, 4, 7)
```



In [15]:

```
# Lets find out some Interesting Facts

print("Some Interesting Patterns")
print("---------------------------------------------------------------------------------------------------")
print("Crops which require very High Ratio of Nitrogen Content in Soil:", data[data["N"]>120]["label"].unique())
print("Crops which require very High Ratio of Phosphorous Content in Soil:", data[data["P"]>100]["label"].unique())
print("Crops which require very High Ratio of Potassium Content in Soil:", data[data["K"]>200]["label"].unique())
print("Crops which require very High Rainfall:", data[data["rainfall"]>200]["label"].unique())
print("Crops which require very Low Temperature:", data[data["temperature"]<10]["label"].unique())
print("Crops which require very High Temperature:", data[data["temperature"]>40]["label"].unique())
print("Crops which require very Low Humidity:", data[data["humidity"]<20]["label"].unique())
print("Crops which require very High PH:", data[data["ph"]>9]["label"].unique())
print("Crops which require very Low PH:", data[data["ph"]<4]["label"].unique())
```

```
Some Interesting Patterns
---------------------------------------------------------------------------------------------------
Crops which require very High Ratio of Nitrogen Content in Soil: ['cotton']
Crops which require very High Ratio of Phosphorous Content in Soil: ['grapes' 'apple']
Crops which require very High Ratio of Potassium Content in Soil: ['grapes' 'apple']
Crops which require very High Rainfall: ['rice' 'papaya' 'coconut']
Crops which require very Low Temperature: ['grapes']
Crops which require very High Temperature: ['grapes' 'papaya']
Crops which require very Low Humidity: ['chickpea' 'kidneybeans']
Crops which require very High PH: ['mothbeans']
Crops which require very Low PH: ['mothbeans']
```

In [16]:

```
# Lets understand which crops can only be Grown in Summer, Wnter and Rainy Season

print ("\033[1m" +"Summer Crops"+ "\033[0m")

print(data[(data["temperature"]>30)&(data["humidity"]>50)]["label"].unique(),"\n")

print ("\033[1m" +"Winter Crops"+ "\033[0m")
print(data[(data["temperature"]<20)&(data["humidity"]>30)]["label"].unique(),"\n")

print ("\033[1m" +"Rainy Crops"+ "\033[0m")
print(data[(data["rainfall"]>200)&(data["humidity"]>30)]["label"].unique())
```

**Summer Crops**
['pigeonpeas' 'mothbeans' 'blackgram' 'mango' 'grapes' 'orange' 'papaya']

**Winter Crops**
['maize' 'pigeonpeas' 'lentil' 'pomegranate' 'grapes' 'orange']

**Rainy Crops**
['rice' 'papaya' 'coconut']

In [18]:

```
#Lets Cluster Crops
#Lets import the warning libraries so that we can avoid warning

from sklearn.cluster import KMeans
import warnings
warnings.filterwarnings("ignore")

#lets select Colums from the Dataset
x = data.loc[:,["N","P","K","temperature","humidity","ph","rainfall"]].values

#Let's check the shape of x
print(x.shape)

#Lets convert this data into a dataframe
x_data = pd.DataFrame(x)
x_data.head()
```

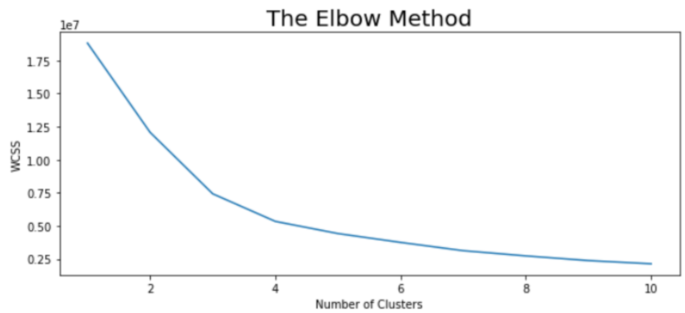
(2200, 7)

Out[18]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| **0** | 90.0 | 42.0 | 43.0 | 20.879744 | 82.002744 | 6.502985 | 202.935536 |
| **1** | 85.0 | 58.0 | 41.0 | 21.770462 | 80.319644 | 7.038096 | 226.655537 |
| **2** | 60.0 | 55.0 | 44.0 | 23.004459 | 82.320763 | 7.840207 | 263.964248 |
| **3** | 74.0 | 35.0 | 40.0 | 26.491096 | 80.158363 | 6.980401 | 242.864034 |
| **4** | 78.0 | 42.0 | 42.0 | 20.130175 | 81.604873 | 7.628473 | 262.717340 |

In [19]:

```python
#Lets determine the optimum Number of Clusters within the Dataset

plt.rcParams["figure.figsize"] = (10,4)

WCSS = []
for i in range(1,11):
    km = KMeans(n_clusters = i, init = "k-means++", max_iter = 300, n_init = 10, random_state = 0)
    km.fit(x)
    WCSS.append(km.inertia_)

#Lets plot the result
#Please note that on the chart below we have two Elbows, one ate 3 and another one at 4. we are going to use the greatest 4
plt.plot(range(1,11), WCSS)
plt.title("The Elbow Method", fontsize = 20)
plt.xlabel("Number of Clusters")
plt.ylabel("WCSS")
plt.show()
```




In [20]:

```python
#Next, we are going to implement the K Means algorithm to perform clustering analysis
km = KMeans(n_clusters = 4, init = "k-means++", max_iter = 300, n_init = 10, random_state = 0)
y_means = km.fit_predict(x)

#Lets find out the results
a = data["label"]
y_means = pd.DataFrame(y_means)
z = pd.concat([y_means, a], axis = 1)
z = z.rename(columns ={0:"cluster"})

#Lets check the Clusters of each Crops
print("Lets check the Results After Applying the K Means Clustering Analysis \n")
print("Crops in First Cluster:", z[z["cluster"] == 0]["label"].unique())
print("---------------------------------------------------------------------")
print("Crops in Second Cluster:", z[z["cluster"] == 1]["label"].unique())
print("---------------------------------------------------------------------")
print("Crops in Third Cluster:", z[z["cluster"] == 2]["label"].unique())
print("---------------------------------------------------------------------")
print("Crops in Fourth Cluster:", z[z["cluster"] == 3]["label"].unique())
```

```
Lets check the Results After Applying the K Means Clustering Analysis

Crops in First Cluster: ['maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans' 'mungbean'
 'blackgram' 'lentil' 'pomegranate' 'mango' 'orange' 'papaya' 'coconut']
---------------------------------------------------------------------
Crops in Second Cluster: ['maize' 'banana' 'watermelon' 'muskmelon' 'papaya' 'cotton' 'coffee']
---------------------------------------------------------------------
Crops in Third Cluster: ['grapes' 'apple']
---------------------------------------------------------------------
Crops in Fourth Cluster: ['rice' 'pigeonpeas' 'papaya' 'coconut' 'jute' 'coffee']
```

In [21]:
```python
#Lets build a predictive Module to help farmers deide the type of Crops given climate and soil conditions.
#Lets split the Dataset from Predictive Modeling

y = data["label"]
x = data.drop(["label"], axis = 1)

print("Shape of x:", x.shape)
print("Shape of y:", y.shape)
```

```
Shape of x: (2200, 7)
Shape of y: (2200,)
```

In [22]:
```python
#Now we are creating Training and Testing Sets for Validation of Results
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 =0)
print("The Shape of x train:", x_train.shape)
print("The Shape of x test:", x_test.shape)
print("The Shape of y train:", y_train.shape)
print("The Shape of y test:", y_test.shape)
```

```
The Shape of x train: (1760, 7)
The Shape of x test: (440, 7)
The Shape of y train: (1760,)
The Shape of y test: (440,)
```

In [23]:
```python
#Next we build a Predictive Model
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```

In [175]:
```python
#Lets evaluate the Model Performance, we see that the Recall in 97% this sugests we are accurate
from sklearn.metrics import classification_report

#Lets print the classification Report as well
#this table gives us the accuracy of the model, greater value of the Precision and Recall = better the Model
cr = classification_report(y_test,y_pred)
print(cr)
```

```
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        18
      banana       1.00      1.00      1.00        18
   blackgram       0.86      0.82      0.84        22
    chickpea       1.00      1.00      1.00        23
     coconut       1.00      1.00      1.00        15
      coffee       1.00      1.00      1.00        17
      cotton       0.89      1.00      0.94        16
      grapes       1.00      1.00      1.00        18
        jute       0.84      1.00      0.91        21
 kidneybeans       1.00      1.00      1.00        20
      lentil       0.94      0.94      0.94        17
       maize       0.94      0.89      0.91        18
       mango       1.00      1.00      1.00        21
   mothbeans       0.88      0.92      0.90        25
    mungbean       1.00      1.00      1.00        17
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        23
      papaya       1.00      0.95      0.98        21
  pigeonpeas       1.00      1.00      1.00        22
 pomegranate       1.00      1.00      1.00        23
        rice       1.00      0.84      0.91        25
  watermelon       1.00      1.00      1.00        17

    accuracy                           0.97       440
   macro avg       0.97      0.97      0.97       440
weighted avg       0.97      0.97      0.97       440
```

In [24]:

```python
#Now we are done with testing, now we are going to do predictions. First we check the head of the dataset
data.head()
```

Out[24]:

| | 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 |

In [186]:

```python
# Here input the values to generate predictions
prediction = model.predict((np.array([[78,
                                       50,
                                       43,
                                       30,
                                       100,
                                       8,
                                       100]])))
print("The Sugested Crop for Given Climatic Condition is :", prediction)
```

```
The Sugested Crop for Given Climatic Condition is : ['jute']
```