Name:Yuvraj Ghadage

RollNo:24207140

Class:Ty(AIDS-C)

In [35]:
```python
import pandas as pd
```

In [36]:
```python
df=pd.read_csv("Crop_recommendation.csv")
```

In [37]:
```python
df
```

Out[37]:

| | 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 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **2195** | 107 | 34 | 32 | 26.774637 | 66.413269 | 6.780064 | 177.774507 | coffee |
| **2196** | 99 | 15 | 27 | 27.417112 | 56.636362 | 6.086922 | 127.924610 | coffee |
| **2197** | 118 | 33 | 30 | 24.131797 | 67.225123 | 6.362608 | 173.322839 | coffee |
| **2198** | 117 | 32 | 34 | 26.272418 | 52.127394 | 6.758793 | 127.175293 | coffee |
| **2199** | 104 | 18 | 30 | 23.603016 | 60.396475 | 6.779833 | 140.937041 | coffee |

2200 rows × 8 columns

In [38]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   N            2200 non-null   int64  
 1   P            2200 non-null   int64  
 2   K            2200 non-null   int64  
 3   temperature  2200 non-null   float64
 4   humidity     2200 non-null   float64
 5   ph           2200 non-null   float64
 6   rainfall     2200 non-null   float64
 7   label        2200 non-null   object 
dtypes: float64(4), int64(3), object(1)
memory usage: 137.6+ KB
```

In [39]:
```python
df.describe()
```

Out[39]:

| | N | P | K | temperature | humidity | ph |
|---|---|---|---|---|---|---|
| **count** | 2200.000000 | 2200.000000 | 2200.000000 | 2200.000000 | 2200.000000 | 2200.000000 |
| **mean** | 50.551818 | 53.362727 | 48.149091 | 25.616244 | 71.481779 | 6.469480 |
| **std** | 36.917334 | 32.985883 | 50.647931 | 5.063749 | 22.263812 | 0.773938 |
| **min** | 0.000000 | 5.000000 | 5.000000 | 8.825675 | 14.258040 | 3.504752 |
| **25%** | 21.000000 | 28.000000 | 20.000000 | 22.769375 | 60.261953 | 5.971693 |
| **50%** | 37.000000 | 51.000000 | 32.000000 | 25.598693 | 80.473146 | 6.425045 |
| **75%** | 84.250000 | 68.000000 | 49.000000 | 28.561654 | 89.948771 | 6.923643 |
| **max** | 140.000000 | 145.000000 | 205.000000 | 43.675493 | 99.981876 | 9.935091 |

In [40]:

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

Out[40]:

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

In [41]:

```
df.sum()
```

Out[41]:

```
N                                                         111214
P                                                         117398
K                                                         105928
temperature                                         56355.736474
humidity                                           157259.914279
ph                                                  14232.856144
rainfall                                           227620.041915
label          ricericericericericericericericericericericeri...
dtype: object
```

In [42]:

```
df.head()
```

Out[42]:

| | 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 [43]:

```
df.tail()
```

Out[43]:

| | N | P | K | temperature | humidity | ph | rainfall | label |
|---|---|---|---|---|---|---|---|---|
| **2195** | 107 | 34 | 32 | 26.774637 | 66.413269 | 6.780064 | 177.774507 | coffee |
| **2196** | 99 | 15 | 27 | 27.417112 | 56.636362 | 6.086922 | 127.924610 | coffee |
| **2197** | 118 | 33 | 30 | 24.131797 | 67.225123 | 6.362608 | 173.322839 | coffee |
| **2198** | 117 | 32 | 34 | 26.272418 | 52.127394 | 6.758793 | 127.175293 | coffee |
| **2199** | 104 | 18 | 30 | 23.603016 | 60.396475 | 6.779833 | 140.937041 | coffee |

In [44]:
```
X = df.drop("label", axis=1)
y = df["label"]
```

In [45]:
```
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report, accuracy_score
import matplotlib.pyplot as plt
```

In [46]:
```
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

num_classes = len(np.unique(y_encoded))
```

In [47]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [48]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y_encoded, test_size=0.2, random_state=42)
```

In [49]:
```
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])
```

```
/home/admin1/anaconda3/lib/python3.12/site-packages/keras/src/layers/core/dense.py:87:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in the mod
el instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [50]:
```
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])
```

In [51]:
```
history = model.fit(
    X_train,
    y_train,
    epochs=50,
    batch_size=16,
    validation_split=0.2
)
```

```
Epoch 1/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.1760 - loss: 2.9206 - val_accurac
y: 0.3551 - val_loss: 2.2653
Epoch 2/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.4523 - loss: 1.9453 - val_accurac
y: 0.6591 - val_loss: 1.2714
Epoch 3/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.7665 - loss: 1.0723 - val_accurac
y: 0.8068 - val_loss: 0.7169
Epoch 4/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8647 - loss: 0.6082 - val_accurac
y: 0.8551 - val_loss: 0.4831
Epoch 5/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9077 - loss: 0.4123 - val_accurac
y: 0.9205 - val_loss: 0.3608
Epoch 6/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9462 - loss: 0.2918 - val_accurac
y: 0.9261 - val_loss: 0.2813
Epoch 7/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9562 - loss: 0.2325 - val_accurac
y: 0.9290 - val_loss: 0.2416
Epoch 8/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9620 - loss: 0.1887 - val_accurac
y: 0.9290 - val_loss: 0.2220
Epoch 9/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9634 - loss: 0.1692 - val_accurac
y: 0.9432 - val_loss: 0.1824
Epoch 10/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9713 - loss: 0.1444 - val_accurac
y: 0.9460 - val_loss: 0.1656
Epoch 11/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9675 - loss: 0.1307 - val_accurac
y: 0.9403 - val_loss: 0.1589
Epoch 12/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9642 - loss: 0.1289 - val_accurac
y: 0.9602 - val_loss: 0.1451
Epoch 13/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9723 - loss: 0.1102 - val_accurac
y: 0.9602 - val_loss: 0.1249
Epoch 14/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9702 - loss: 0.1075 - val_accurac
y: 0.9489 - val_loss: 0.1362
Epoch 15/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9681 - loss: 0.0968 - val_accurac
y: 0.9602 - val_loss: 0.1125
Epoch 16/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9779 - loss: 0.0814 - val_accurac
y: 0.9602 - val_loss: 0.1103
Epoch 17/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9835 - loss: 0.0781 - val_accurac
y: 0.9545 - val_loss: 0.1105
Epoch 18/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9775 - loss: 0.0716 - val_accurac
y: 0.9688 - val_loss: 0.0951
Epoch 19/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9753 - loss: 0.0742 - val_accurac
y: 0.9773 - val_loss: 0.0895
Epoch 20/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9820 - loss: 0.0614 - val_accurac
y: 0.9688 - val_loss: 0.0834
Epoch 21/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9810 - loss: 0.0661 - val_accurac
y: 0.9631 - val_loss: 0.0959
Epoch 22/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9874 - loss: 0.0587 - val_accurac
y: 0.9716 - val_loss: 0.0946
```

```
Epoch 23/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9811 - loss: 0.0542 - val_accurac
y: 0.9688 - val_loss: 0.0883
Epoch 24/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9828 - loss: 0.0500 - val_accurac
y: 0.9773 - val_loss: 0.0720
Epoch 25/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9809 - loss: 0.0491 - val_accurac
y: 0.9688 - val_loss: 0.0733
Epoch 26/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9920 - loss: 0.0442 - val_accurac
y: 0.9688 - val_loss: 0.0713
Epoch 27/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9813 - loss: 0.0506 - val_accurac
y: 0.9744 - val_loss: 0.0647
Epoch 28/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9847 - loss: 0.0501 - val_accurac
y: 0.9688 - val_loss: 0.0684
Epoch 29/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9820 - loss: 0.0460 - val_accurac
y: 0.9659 - val_loss: 0.0750
Epoch 30/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9820 - loss: 0.0477 - val_accurac
y: 0.9773 - val_loss: 0.0574
Epoch 31/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9885 - loss: 0.0395 - val_accurac
y: 0.9744 - val_loss: 0.0623
Epoch 32/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9905 - loss: 0.0358 - val_accurac
y: 0.9801 - val_loss: 0.0598
Epoch 33/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9920 - loss: 0.0347 - val_accurac
y: 0.9858 - val_loss: 0.0536
Epoch 34/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9879 - loss: 0.0389 - val_accurac
y: 0.9773 - val_loss: 0.0544
Epoch 35/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9849 - loss: 0.0356 - val_accurac
y: 0.9688 - val_loss: 0.0671
Epoch 36/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9861 - loss: 0.0358 - val_accurac
y: 0.9830 - val_loss: 0.0500
Epoch 37/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9895 - loss: 0.0325 - val_accurac
y: 0.9801 - val_loss: 0.0503
Epoch 38/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9864 - loss: 0.0353 - val_accurac
y: 0.9830 - val_loss: 0.0495
Epoch 39/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9927 - loss: 0.0246 - val_accurac
y: 0.9830 - val_loss: 0.0505
Epoch 40/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9871 - loss: 0.0320 - val_accurac
y: 0.9801 - val_loss: 0.0504
Epoch 41/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9956 - loss: 0.0242 - val_accurac
y: 0.9688 - val_loss: 0.0600
Epoch 42/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9930 - loss: 0.0254 - val_accurac
y: 0.9801 - val_loss: 0.0512
Epoch 43/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9886 - loss: 0.0287 - val_accurac
y: 0.9830 - val_loss: 0.0437
Epoch 44/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9915 - loss: 0.0282 - val_accurac
y: 0.9716 - val_loss: 0.0550
```

```
Epoch 45/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9915 - loss: 0.0273 - val_accurac
y: 0.9773 - val_loss: 0.0428
Epoch 46/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9928 - loss: 0.0230 - val_accurac
y: 0.9659 - val_loss: 0.0584
Epoch 47/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9901 - loss: 0.0244 - val_accurac
y: 0.9886 - val_loss: 0.0441
Epoch 48/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9931 - loss: 0.0243 - val_accurac
y: 0.9858 - val_loss: 0.0401
Epoch 49/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9920 - loss: 0.0237 - val_accurac
y: 0.9858 - val_loss: 0.0393
Epoch 50/50
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9937 - loss: 0.0218 - val_accurac
y: 0.9830 - val_loss: 0.0484
```

In [53]:

```python
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print("Test Accuracy:", test_accuracy)
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9702 - loss: 0.0828
Test Accuracy: 0.9704545736312866
```

In [ ]: