In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [11]:

```
df=pd.read_csv("crop.csv")
df.head()
```

Out[11]:

| | 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 [10]:

```
#checking for various crops present
crop_names=df['label'].unique()
crop_names
```

Out[10]:

```
array(['rice', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas',
       'mothbeans', 'mungbean', 'blackgram', 'lentil', 'pomegranate',
       'banana', 'mango', 'grapes', 'watermelon', 'muskmelon', 'apple',
       'orange', 'papaya', 'coconut', 'cotton', 'jute', 'coffee'],
      dtype=object)
```

In [4]:

```
#checking for the null values
has_null_values = df.isnull().any().any()

if has_null_values:
    print("The dataset has null values.")
else:
    print("The dataset does not have any null values.")
```

```
The dataset does not have any null values.
```

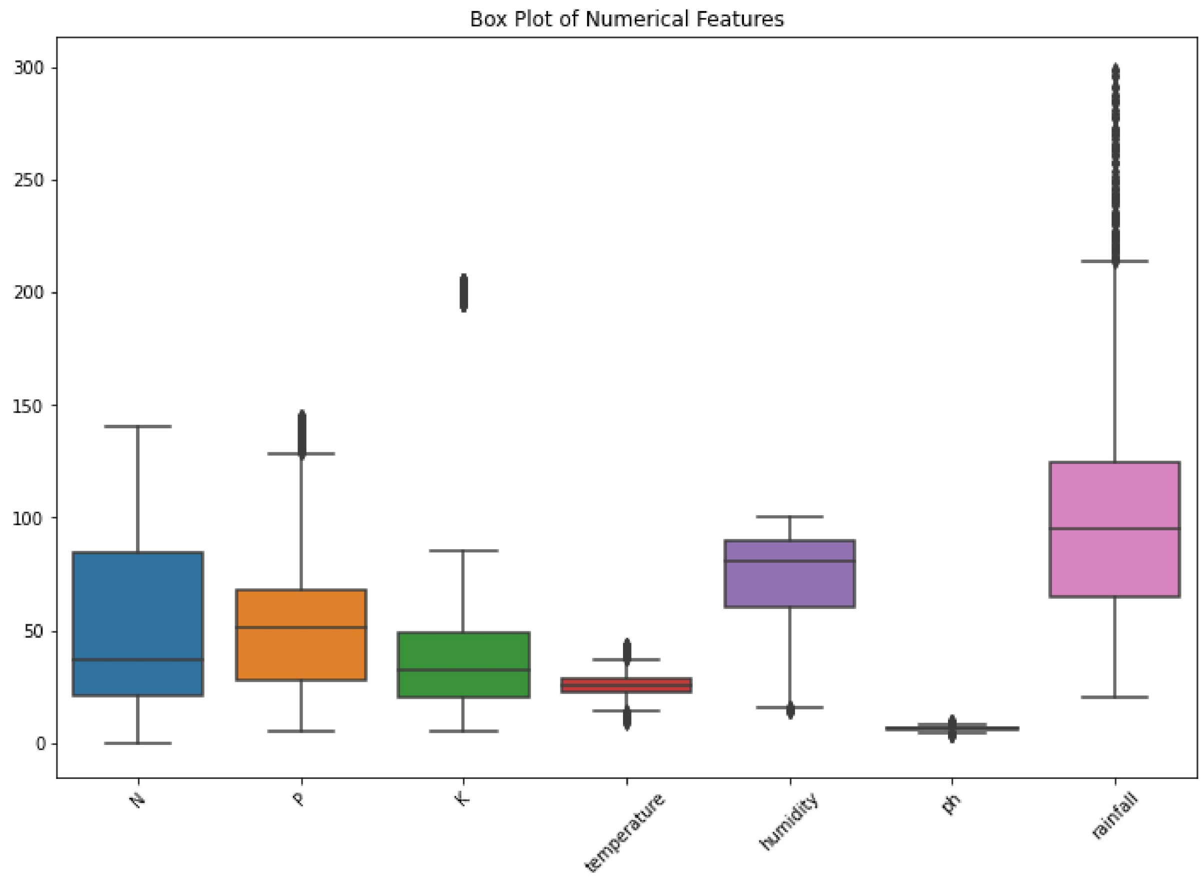
In [5]:

```
#checking for the duplicate rows
duplicate_rows = df[df.duplicated()]
num_duplicates = len(duplicate_rows)
print("Duplicate rows:")
print(duplicate_rows)
print(f"Total number of duplicates: {num_duplicates}")
```

```
Duplicate rows:
Empty DataFrame
Columns: [N, P, K, temperature, humidity, ph, rainfall, label]
Index: []
Total number of duplicates: 0
```

In [6]:

```
#checking for outliers in numerical features
numerical_features = df.select_dtypes(include=['float64', 'int64'])
plt.figure(figsize=(12, 8))
sns.boxplot(data=numerical_features)
plt.title('Box Plot of Numerical Features')
plt.xticks(rotation=45)
plt.show()
```



In [13]:

```
#seperating the feautures and target
features = df[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']]
target = df['label']
labels = df['label']
```

In [16]:

```
#feature scaling before splitting the datset
c=df.label.astype('category')
targets = dict(enumerate(c.cat.categories))
df['target']=c.cat.codes

y=df.target
X=df[['N','P','K','temperature','humidity','ph','rainfall']]




from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=1)

scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)

# we must apply the scaling to the test set as well that we are computing for the training set
X_test_scaled = scaler.transform(X_test)
```

In [28]:

```
#splitting the data into training and testing data
from sklearn.model_selection import train_test_split
Xtrain, Xtest, Ytrain, Ytest = train_test_split(features,target,test_size = 0.2,random_state =2)
```

In [30]:

```
#perform randomforest classification and predicting the accuracy score and the classification report
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix

RF = RandomForestClassifier(n_estimators=20, random_state=0)
RF.fit(Xtrain,Ytrain)

predicted_values = RF.predict(Xtest)

x = accuracy_score(Ytest, predicted_values)

print("RF's Accuracy is: ", x)
print("classification report is:")

print(classification_report(Ytest,predicted_values))
```

```
RF's Accuracy is:  0.990909090909091
classification report is:
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       0.94      1.00      0.97        16
    chickpea       1.00      1.00      1.00        21
     coconut       1.00      1.00      1.00        21
      coffee       1.00      1.00      1.00        22
      cotton       1.00      1.00      1.00        20
      grapes       1.00      1.00      1.00        18
        jute       0.90      1.00      0.95        28
 kidneybeans       1.00      1.00      1.00        14
      lentil       1.00      1.00      1.00        23
       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        26
   mothbeans       1.00      0.95      0.97        19
    mungbean       1.00      1.00      1.00        24
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        29
      papaya       1.00      1.00      1.00        19
  pigeonpeas       1.00      1.00      1.00        18
 pomegranate       1.00      1.00      1.00        17
        rice       1.00      0.81      0.90        16
  watermelon       1.00      1.00      1.00        15

    accuracy                           0.99       440
   macro avg       0.99      0.99      0.99       440
weighted avg       0.99      0.99      0.99       440
```

In [31]:

```
#predicting the crop by giving the inputs for various feutures
data = np.array([[104,18, 30, 23.603016, 60.3, 6.7, 140.91]])
prediction = RF.predict(data)
print(prediction)
```

```
['coffee']
```

In [34]:

```
data = np.array([[83, 100, 60, 30, 70.3, 7.5, 150.9]])
prediction = RF.predict(data)
print(prediction)
```

```
['banana']
```

In [35]:

```
'''based on the classification report, it appears that the model performs very well for most classes
with high precision, recall, and F1-score values close to 1.00. The accuracy of the model on the tes
is also quite high at 0.99 (99%).

However, it's worth noting that for the class "rice," the recall value is lower (0.81).
This indicates that the model may have difficulty correctly identifying instances of "rice" in the c
Further investigation may be needed to understand the reasons behind this lower recall value and to
improve the model's performance for this particular class.'''
```

Out[35]:

```
'based on the classification report, it appears that the model performs very well for
most classes,\nwith high precision, recall, and F1-score values close to 1.00. The ac
curacy of the model on the test dataset \nis also quite high at 0.99 (99%).\n\nHoweve
r, it\'s worth noting that for the class "rice," the recall value is lower (0.81). \n
This indicates that the model may have difficulty correctly identifying instances of
"rice" in the dataset.\nFurther investigation may be needed to understand the reasons
behind this lower recall value and to potentially \nimprove the model\'s performance
for this particular class.'
```

In [ ]: