In [2]:
```
# ================================
# 🌾 Crop Recommendation Project
# ================================

# 📦 1. Import Libraries
import os
import warnings
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.exceptions import ConvergenceWarning
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_sc
from sklearn.preprocessing import LabelEncoder, FunctionTransformer, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from imblearn.combine import SMOTETomek
import pickle

warnings.filterwarnings('ignore', category=ConvergenceWarning)

# ================================
# 📂 2. Data Collection
# ================================
data = pd.read_csv(r'data\Crop_recommendation.csv')
print("Data Shape:", data.shape)
display(data.head())

# ================================
# 🔍 3. Exploratory Data Analysis
# ================================
print("\nDataset Info:")
data.info()

print("\nSummary Statistics:")
display(data.describe())

print("\nUnique Crops:")
print(data['label'].unique())

# ================================
# 🧹 4. Data Cleaning & Encoding
# ================================
# Check for missing values
print("\nMissing Values Count:")
print(data.isnull().sum().sum())

# Encode crop labels
encoder = LabelEncoder()
data['label_encoded'] = encoder.fit_transform(data['label'])
```

```
label_mapping = pd.DataFrame({
    'Crop Name': encoder.classes_,
    'Encoded Value': range(len(encoder.classes_))
})
display(label_mapping)

# ================================
# 📊 5. Outlier Detection (Visualization)
# ================================
data.hist(figsize=(15, 15))
plt.show()

sns.boxplot(data, orient='h')
plt.title('Boxplot for Detecting Outliers')
plt.show()

# ================================
# 🧱 6. Outlier Removal (IQR Method)
# ================================
df = data.copy()

for col in df.select_dtypes(include=['int64', 'float64']).columns:
    Q1, Q3 = df[col].quantile([0.25, 0.75])
    IQR = Q3 - Q1
    lower, upper = Q1 - 1 * IQR, Q3 + 1 * IQR
    df = df[(df[col] >= lower) & (df[col] <= upper)]

print("Remaining Crops After Outlier Removal:")
print(df['label'].value_counts())

# ================================
# 🧠 7. Feature Encoding After Outlier Removal
# ================================
df['label_encoded'] = encoder.fit_transform(df['label'])
display(df[['label', 'label_encoded']].drop_duplicates().sort_values('label_encoded

# ================================
# 📈 8. Visualize Cleaned Data
# ================================
sns.boxplot(df, orient='h')
plt.title("Boxplot After Outlier Removal")
plt.show()


df.hist(figsize=(15, 15))
plt.show()
for col in df.columns[:-1]:
    sns.histplot(df[col], kde=True)
    plt.title(f"Distribution of {col}")
    plt.show()

# ================================
# 🧩 9. Feature & Target Split
# ================================
X = df.drop(columns=['label', 'label_encoded'])
```

```
y = df['label_encoded']

# ================================
# ⚙️ 10. Train-Test Split
# ================================
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# ================================
# ⚖️ 11. Handle Imbalance (SMOTE+Tomek)
# ================================
smote_tomek = SMOTETomek(random_state=42)
X_train, y_train = smote_tomek.fit_resample(X_train, y_train)

# ================================
# 🔄 12. Data Transformation
# ================================
ft = FunctionTransformer(np.log1p)
X_train = pd.DataFrame(ft.fit_transform(X_train), columns=X.columns)
X_test = pd.DataFrame(ft.transform(X_test), columns=X.columns)

scaler = StandardScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train), columns=X.columns)
X_test = pd.DataFrame(scaler.transform(X_test), columns=X.columns)

# ================================
# 🔍 13. Feature Selection (SFS)
# ================================
lr_fs = LogisticRegression()
fs = SFS(lr_fs, k_features="best", forward=True, scoring='accuracy')
fs.fit(X, y)

for i in range(1, 8):
    fs_i = SFS(lr_fs, k_features=i, forward=True, scoring='accuracy')
    fs_i.fit(X, y)
    print(f"Features: {fs_i.k_feature_names_}, No. of Features: {i}, CV Score: {fs_

# ================================
# 😈 14. Model Training & Evaluation
# ================================
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000, C=0.5, solver='lbfgs',
    "Random Forest": RandomForestClassifier(
        n_estimators=200, max_depth=5, min_samples_split=6,
        min_samples_leaf=4, max_features=0.6, bootstrap=True, random_state=42
    ),
    "XGBoost": XGBClassifier(
        max_depth=3, learning_rate=0.05, n_estimators=300,
        subsample=0.7, colsample_bytree=0.7,
        reg_lambda=3.0, reg_alpha=2.0, random_state=42, eval_metric='mlogloss'
    )
}

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
```

```
for name, model in models.items():
    print(f"\n{'='*10} {name} {'='*10}")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    print(f"Training Accuracy: {model.score(X_train, y_train)*100:.2f}%")
    print(f"Testing Accuracy: {model.score(X_test, y_test)*100:.2f}%")

    cv_score = cross_val_score(model, X_train, y_train, cv=cv, scoring='accuracy')
    print(f"Mean CV Accuracy: {cv_score.mean()*100:.2f}%")

    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))
```

```
Data Shape: (2200, 8)
```

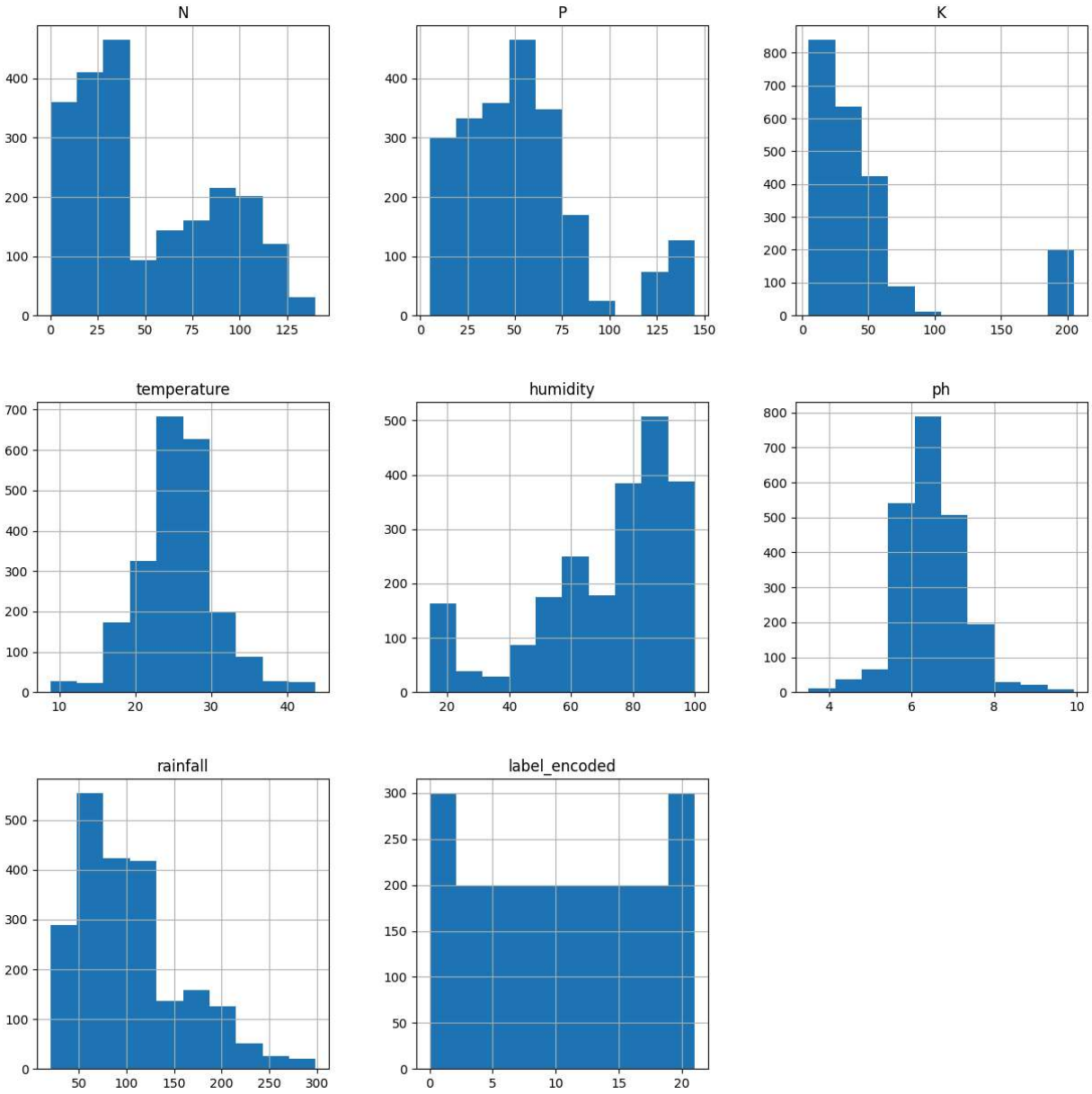
	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

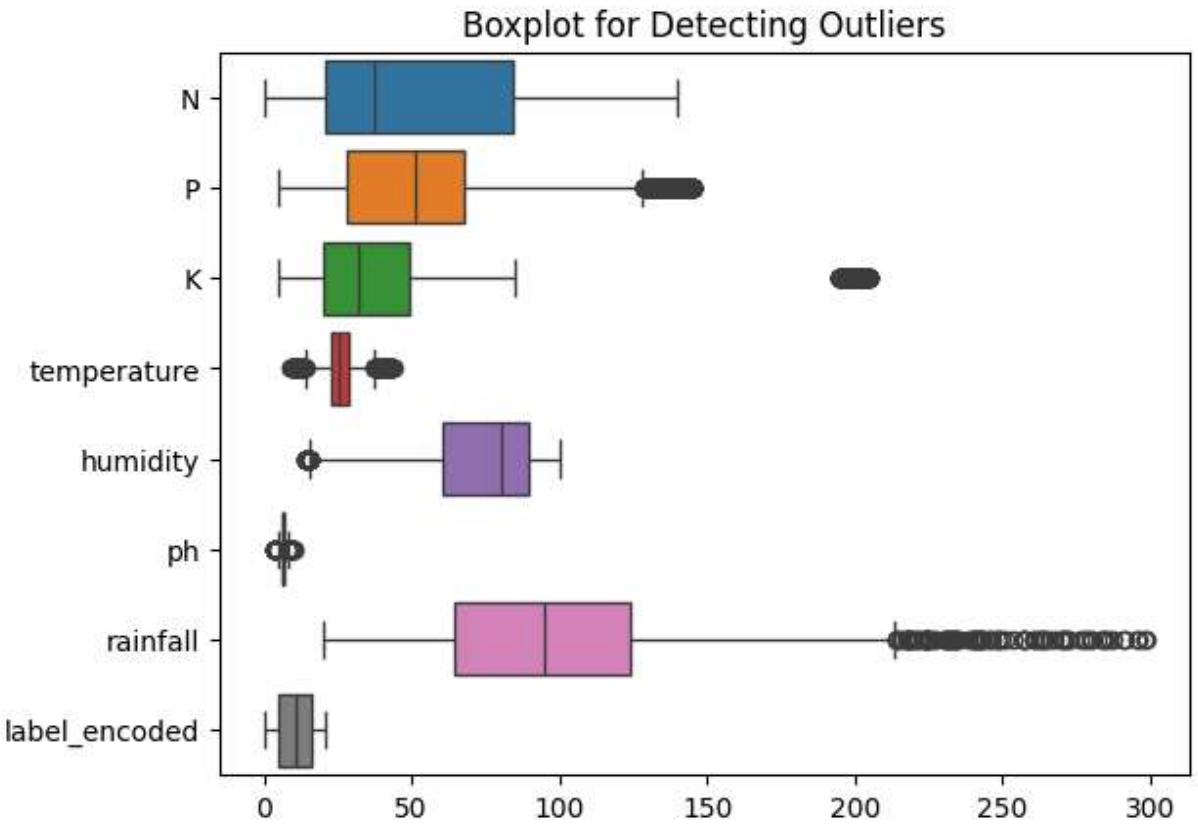
```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   N            2200 non-null   int64
 1   P            2200 non-null   int64
 2   K            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

Summary Statistics:
```

	N	P	K	temperature	humidity	ph	r
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.0
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.4
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.9
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.2
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.5
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.8
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.2
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.5

```
Unique Crops:
['rice' 'maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans'
 'mungbean' 'blackgram' 'lentil' 'pomegranate' 'banana' 'mango' 'grapes'
 'watermelon' 'muskmelon' 'apple' 'orange' 'papaya' 'coconut' 'cotton'
 'jute' 'coffee']

Missing Values Count:
0
```

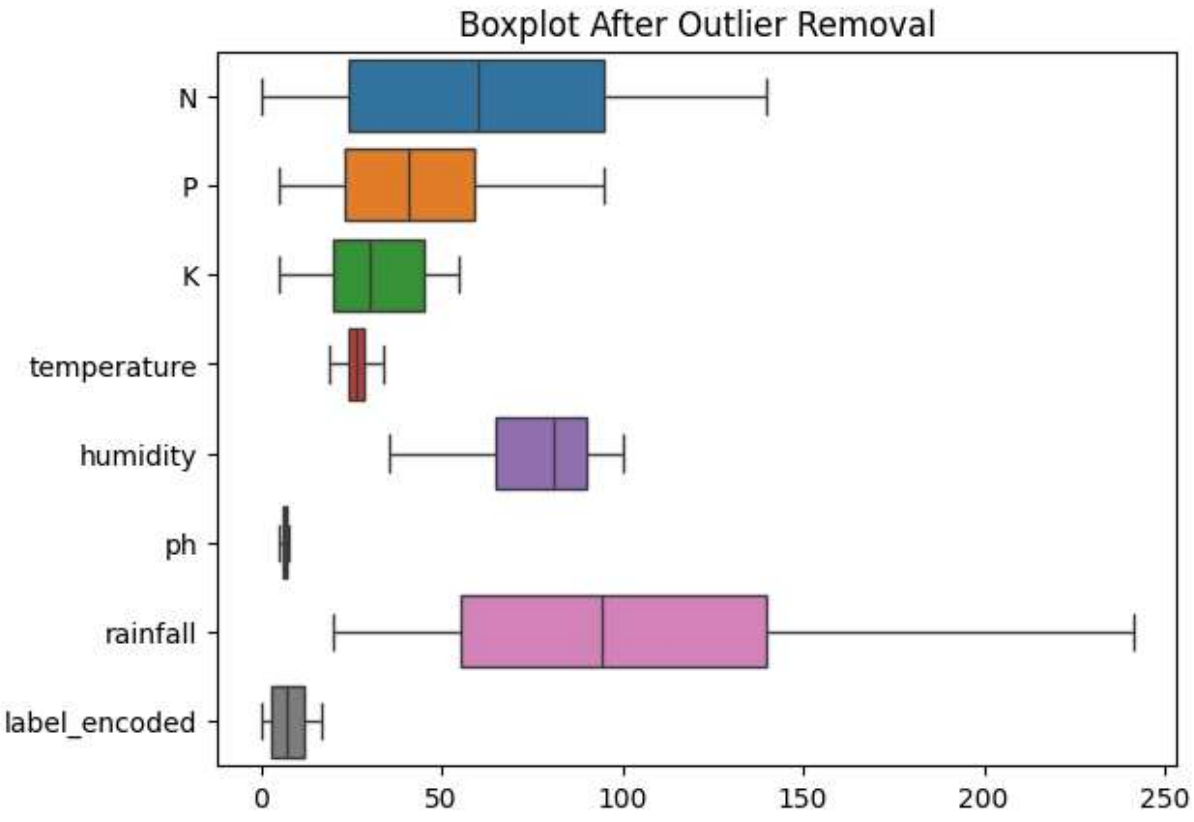
	Crop Name	Encoded Value
0	apple	0
1	banana	1
2	blackgram	2
3	chickpea	3
4	coconut	4
5	coffee	5
6	cotton	6
7	grapes	7
8	jute	8
9	kidneybeans	9
10	lentil	10
11	maize	11
12	mango	12
13	mothbeans	13
14	mungbean	14
15	muskmelon	15
16	orange	16
17	papaya	17
18	pigeonpeas	18
19	pomegranate	19
20	rice	20
21	watermelon	21

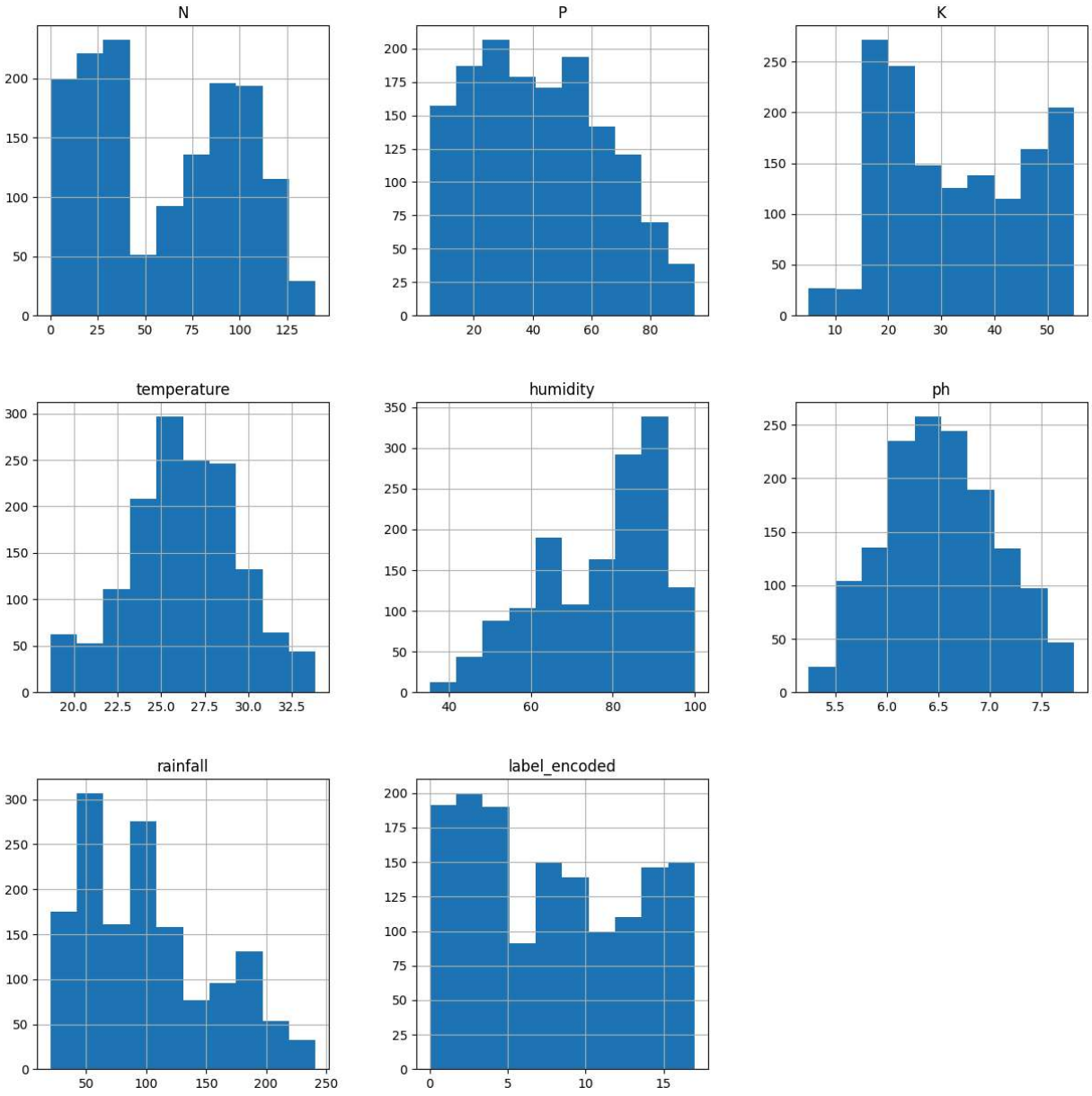

N
P
K
temperature
humidity
ph
rainfall
label_encoded

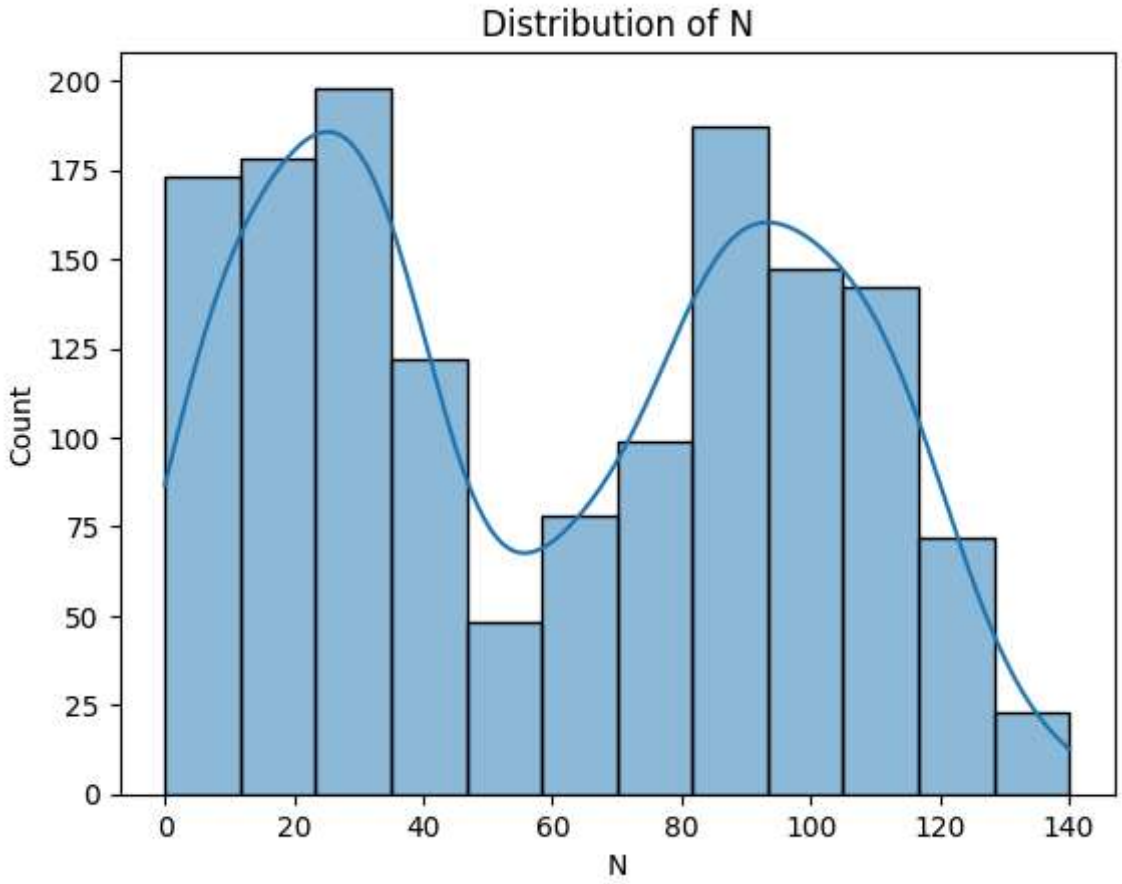


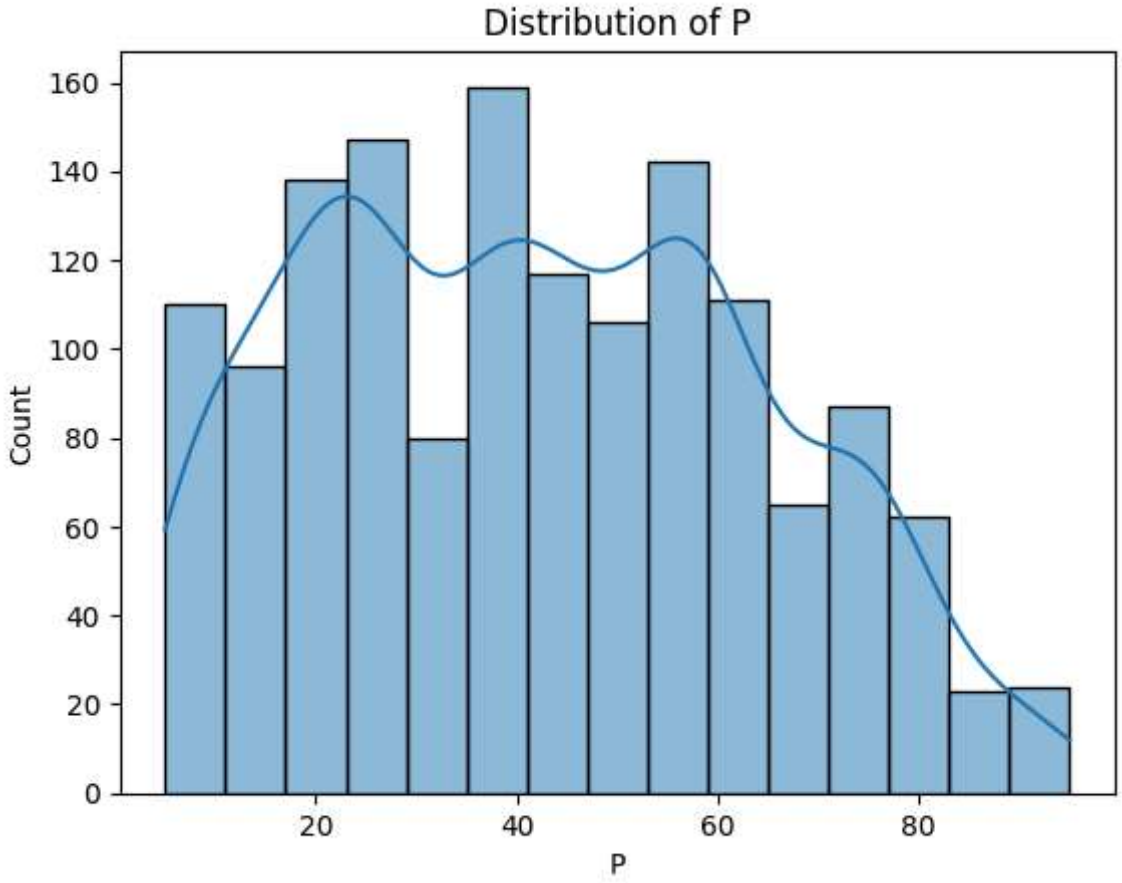

```
Remaining Crops After Outlier Removal:
label
banana         100
mungbean       100
coconut        100
jute           100
muskmelon      100
coffee         100
watermelon     100
pomegranate     93
blackgram       91
lentil          91
cotton          90
maize           90
mango           60
orange          58
pigeonpeas      53
papaya          52
rice            50
mothbeans       39
Name: count, dtype: int64
```

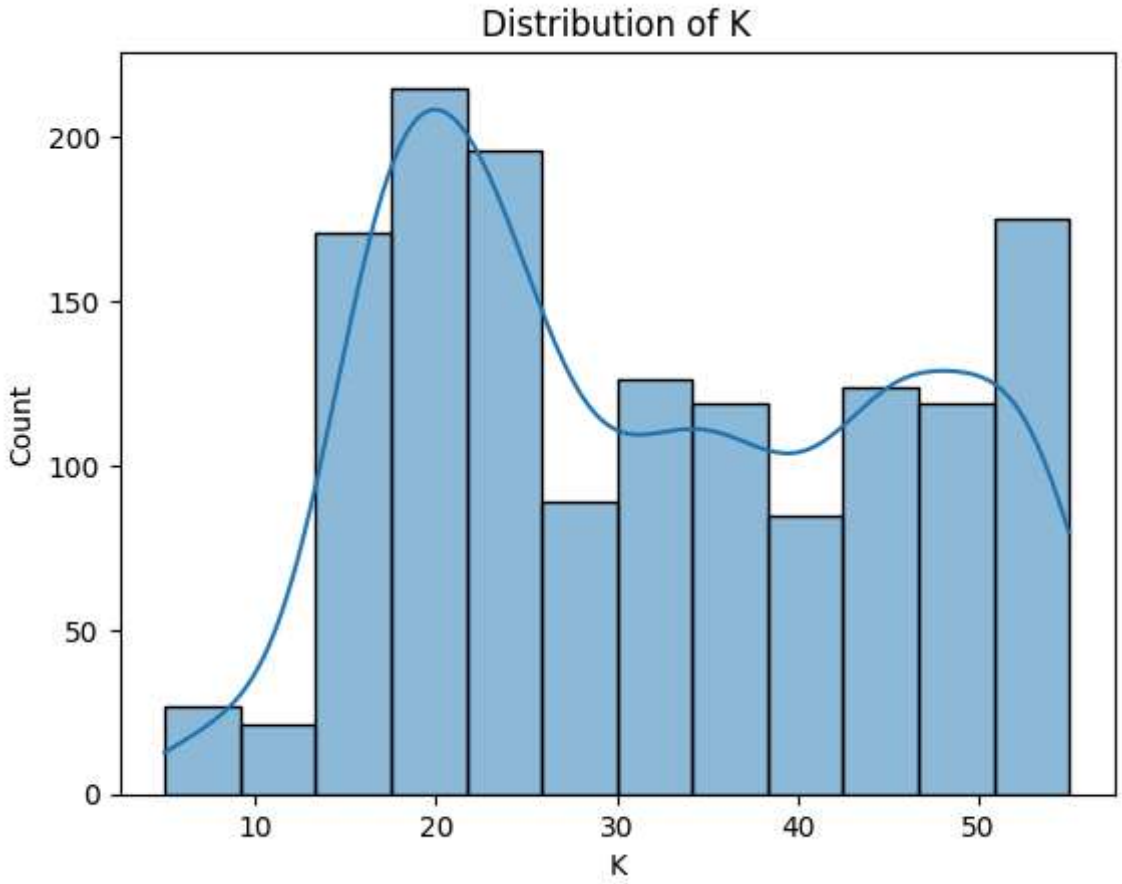
	label	label_encoded
1000	banana	0
700	blackgram	1
1800	coconut	2
2100	coffee	3
1900	cotton	4
2000	jute	5
800	lentil	6
100	maize	7
1100	mango	8
505	mothbeans	9
600	mungbean	10
1400	muskmelon	11
1601	orange	12
1705	papaya	13
402	pigeonpeas	14
900	pomegranate	15
0	rice	16
1300	watermelon	17

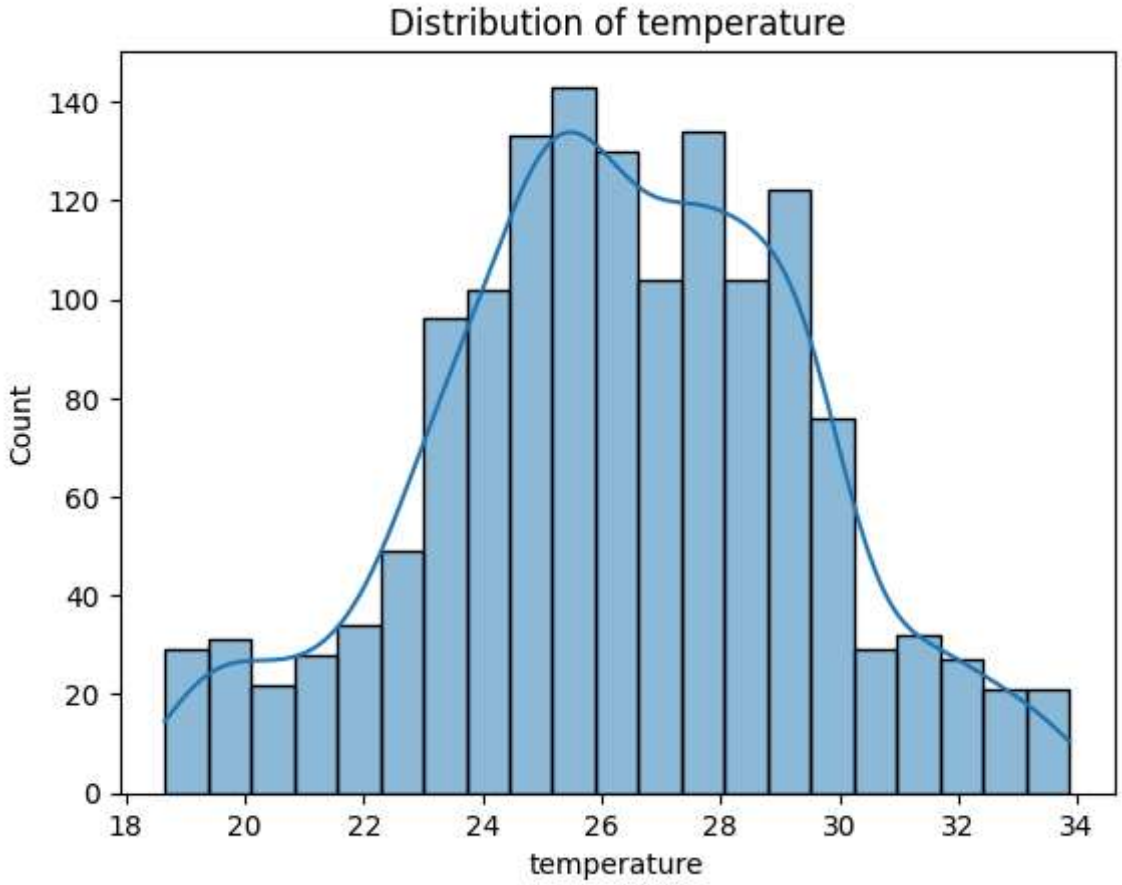

Boxplot After Outlier Removal
N
P
K
temperature
humidity
ph
rainfall
label_encoded
0
50
100
150
200
250

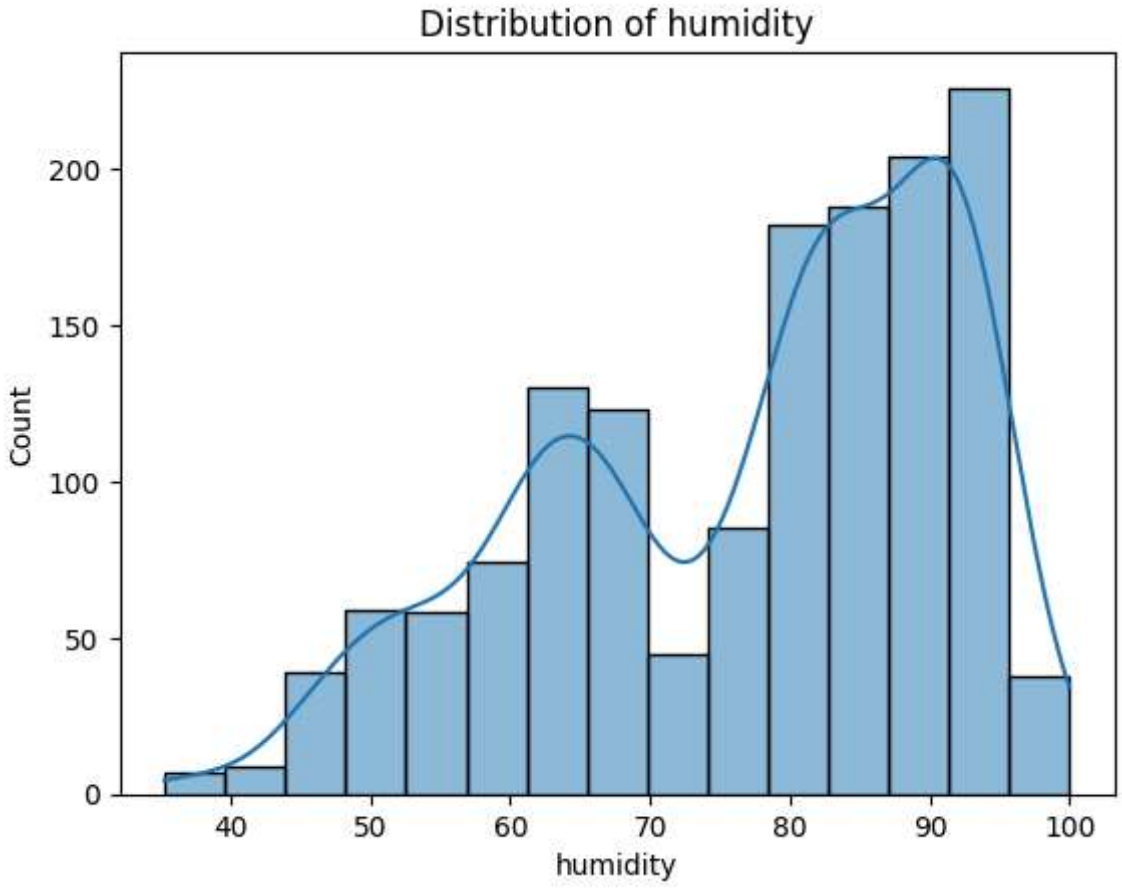

N
P
K
temperature
humidity
ph
rainfall
label_encoded

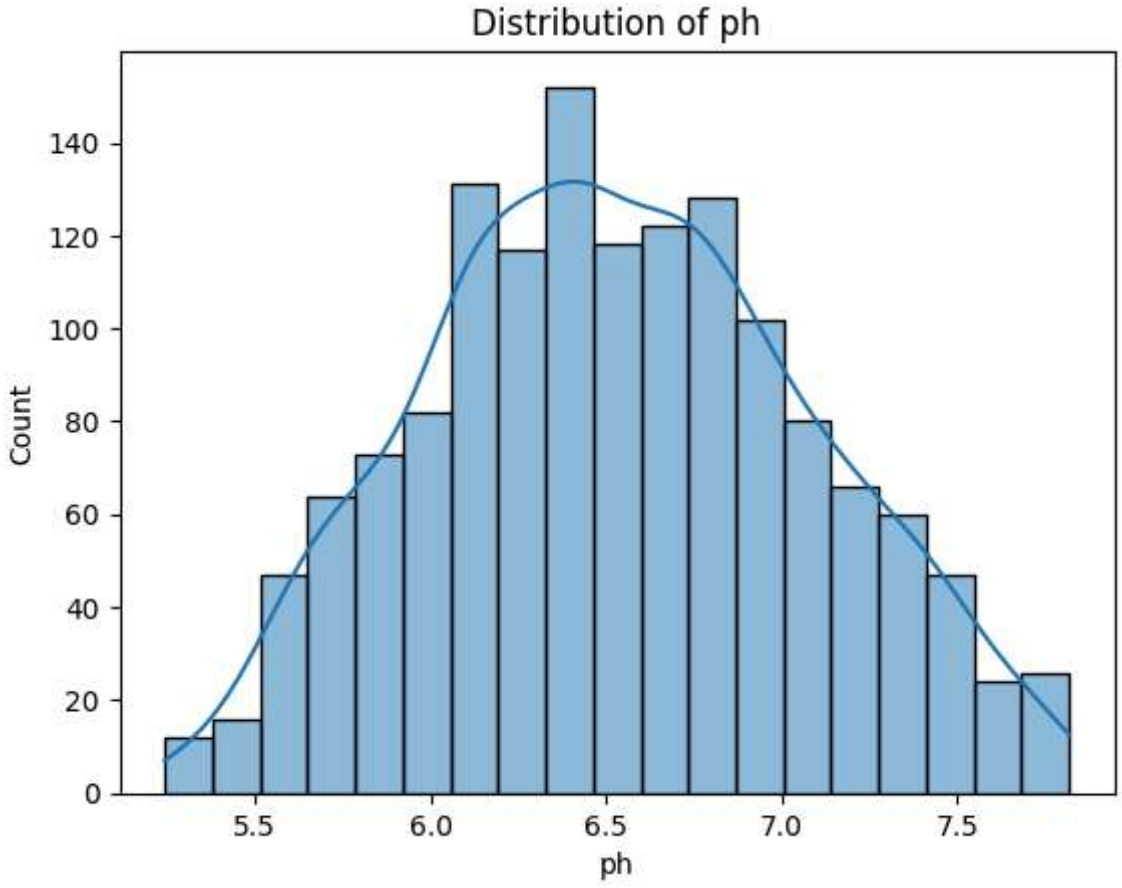

Distribution of N
Count
N
0
25
50
75
100
125
150
175
200
0
20
40
60
80
100
120
140

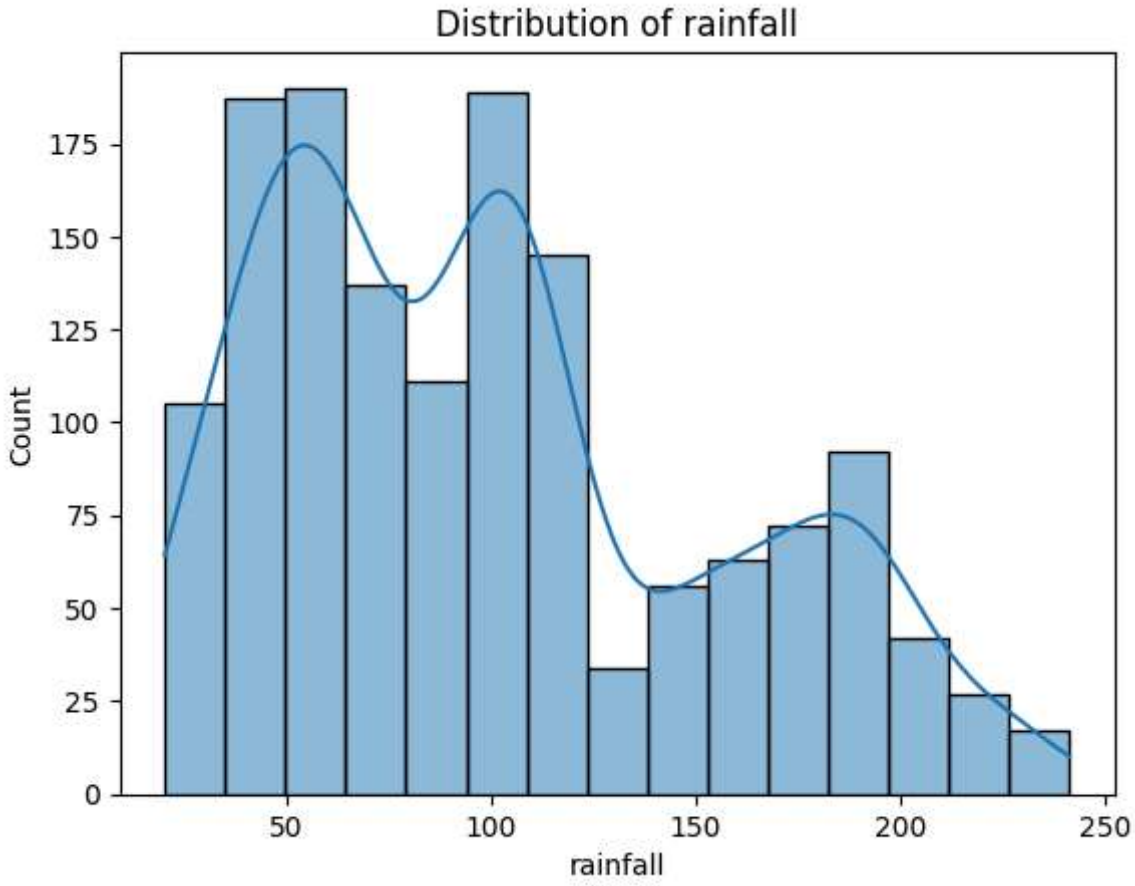

Distribution of P
Count
P
0
20
40
60
80
100
120
140
160
20
40
60
80

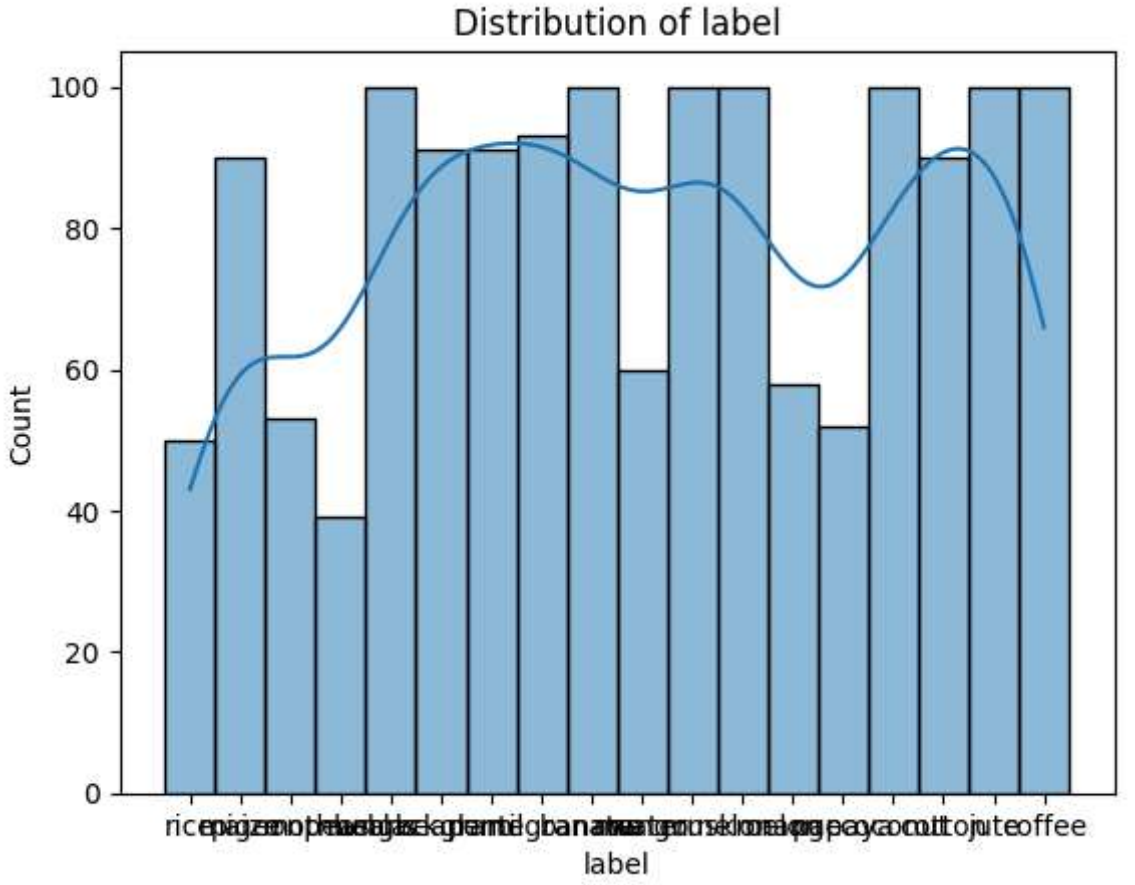

Distribution of K
Count
K
0
50
100
150
200
10
20
30
40
50


Distribution of temperature
Count
temperature
0
20
40
60
80
100
120
140
18
20
22
24
26
28
30
32
34


Distribution of humidity
Count
humidity
0
50
100
150
200
40
50
60
70
80
90
100


Distribution of ph
Count
ph
0
20
40
60
80
100
120
140
5.5
6.0
6.5
7.0
7.5


Distribution of rainfall
Count
rainfall
0
25
50
75
100
125
150
175
50
100
150
200
250


Distribution of label
Count
label
0
20
40
60
80
100

```
Features: ('rainfall',), No. of Features: 1, CV Score: 0.3518
Features: ('N', 'rainfall'), No. of Features: 2, CV Score: 0.4145
Features: ('N', 'humidity', 'rainfall'), No. of Features: 3, CV Score: 0.6728
Features: ('N', 'K', 'humidity', 'rainfall'), No. of Features: 4, CV Score: 0.8494
Features: ('N', 'K', 'temperature', 'humidity', 'rainfall'), No. of Features: 5, CV
Score: 0.9216
Features: ('N', 'P', 'K', 'temperature', 'humidity', 'rainfall'), No. of Features:
6, CV Score: 0.9509
Features: ('N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall'), No. of Featu
res: 7, CV Score: 0.9530

========== Logistic Regression ==========
Training Accuracy: 96.30%
Testing Accuracy: 95.24%
```

```
c:\Users\david\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\lin
ear_model\_logistic.py:1272: FutureWarning: 'multi_class' was deprecated in version
1.5 and will be removed in 1.7. From then on, it will always use 'multinomial'. Leav
e it to its default value to avoid this warning.
  warnings.warn(
c:\Users\david\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\lin
ear_model\_logistic.py:1272: FutureWarning: 'multi_class' was deprecated in version
1.5 and will be removed in 1.7. From then on, it will always use 'multinomial'. Leav
e it to its default value to avoid this warning.
  warnings.warn(
c:\Users\david\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\lin
ear_model\_logistic.py:1272: FutureWarning: 'multi_class' was deprecated in version
1.5 and will be removed in 1.7. From then on, it will always use 'multinomial'. Leav
e it to its default value to avoid this warning.
  warnings.warn(
c:\Users\david\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\lin
ear_model\_logistic.py:1272: FutureWarning: 'multi_class' was deprecated in version
1.5 and will be removed in 1.7. From then on, it will always use 'multinomial'. Leav
e it to its default value to avoid this warning.
  warnings.warn(
c:\Users\david\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\lin
ear_model\_logistic.py:1272: FutureWarning: 'multi_class' was deprecated in version
1.5 and will be removed in 1.7. From then on, it will always use 'multinomial'. Leav
e it to its default value to avoid this warning.
  warnings.warn(
c:\Users\david\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\lin
ear_model\_logistic.py:1272: FutureWarning: 'multi_class' was deprecated in version
1.5 and will be removed in 1.7. From then on, it will always use 'multinomial'. Leav
e it to its default value to avoid this warning.
  warnings.warn(
```

```
Mean CV Accuracy: 95.54%

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        20
           1       0.89      0.94      0.92        18
           2       1.00      1.00      1.00        20
           3       1.00      0.95      0.97        20
           4       0.95      1.00      0.97        18
           5       0.83      0.75      0.79        20
           6       0.94      0.89      0.91        18
           7       1.00      0.94      0.97        18
           8       1.00      1.00      1.00        12
           9       1.00      1.00      1.00         8
          10       1.00      1.00      1.00        20
          11       1.00      1.00      1.00        20
          12       1.00      1.00      1.00        12
          13       0.90      0.90      0.90        10
          14       1.00      1.00      1.00        11
          15       1.00      0.95      0.97        19
          16       0.57      0.80      0.67        10
          17       1.00      1.00      1.00        20

    accuracy                           0.95       294
   macro avg       0.95      0.95      0.95       294
weighted avg       0.96      0.95      0.95       294


========== Random Forest ==========
Training Accuracy: 99.16%
Testing Accuracy: 98.64%
Mean CV Accuracy: 99.02%

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        20
           1       1.00      1.00      1.00        18
           2       1.00      1.00      1.00        20
           3       1.00      1.00      1.00        20
           4       1.00      1.00      1.00        18
           5       1.00      0.80      0.89        20
           6       1.00      1.00      1.00        18
           7       1.00      1.00      1.00        18
           8       1.00      1.00      1.00        12
           9       1.00      1.00      1.00         8
          10       1.00      1.00      1.00        20
          11       1.00      1.00      1.00        20
          12       1.00      1.00      1.00        12
          13       1.00      1.00      1.00        10
          14       1.00      1.00      1.00        11
          15       1.00      1.00      1.00        19
          16       0.71      1.00      0.83        10
          17       1.00      1.00      1.00        20
```

```
    accuracy                           0.99       294
   macro avg       0.98      0.99      0.98       294
weighted avg       0.99      0.99      0.99       294


========== XGBoost ==========
Training Accuracy: 99.86%
Testing Accuracy: 98.64%
Mean CV Accuracy: 98.88%

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        20
           1       0.95      1.00      0.97        18
           2       1.00      1.00      1.00        20
           3       1.00      0.95      0.97        20
           4       1.00      1.00      1.00        18
           5       0.95      0.95      0.95        20
           6       1.00      0.94      0.97        18
           7       1.00      1.00      1.00        18
           8       1.00      1.00      1.00        12
           9       0.89      1.00      0.94         8
          10       1.00      1.00      1.00        20
          11       1.00      1.00      1.00        20
          12       1.00      1.00      1.00        12
          13       1.00      1.00      1.00        10
          14       1.00      0.91      0.95        11
          15       1.00      1.00      1.00        19
          16       0.91      1.00      0.95        10
          17       1.00      1.00      1.00        20

    accuracy                           0.99       294
   macro avg       0.98      0.99      0.98       294
weighted avg       0.99      0.99      0.99       294
```

In [ ]: