```
!pip install pandas numpy scikit-learn matplotlib seaborn

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.metrics import accuracy_score, classification_report, mean_squared_error
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.0.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.16.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.59.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
```

```
# 1. Crop Recommendation Dataset
crop_df = pd.read_csv("/content/archive (5).zip")

# 2. Fertilizer Prediction Dataset
fert_df = pd.read_csv("/content/archive (6).zip")

# 3. Crops NPK Dataset
npk_df = pd.read_csv("/content/archive (7).zip")

print("Crop Dataset:", crop_df.shape)
print("Fertilizer Dataset:", fert_df.shape)
print("NPK Dataset:", npk_df.shape)
```

```
Crop Dataset: (2200, 8)
Fertilizer Dataset: (99, 9)
NPK Dataset: (20000, 10)
```

```
# Make all crop names lowercase for consistency
crop_df['label'] = crop_df['label'].str.lower()
fert_df['Crop Type'] = fert_df['Crop Type'].str.lower()
npk_df['Crop'] = npk_df['Crop'].str.lower()

# Encode categorical features in Fertilizer Dataset
fert_df['Soil Type'] = LabelEncoder().fit_transform(fert_df['Soil Type'])
fert_df['Crop Type'] = LabelEncoder().fit_transform(fert_df['Crop Type'])
fert_df['Fertilizer Name'] = LabelEncoder().fit_transform(fert_df['Fertilizer Name'])

print("Unique crops in Crop dataset:", crop_df['label'].nunique())
print("Unique fertilizers:", fert_df['Fertilizer Name'].nunique())
```

```
Unique crops in Crop dataset: 22
Unique fertilizers: 7
```

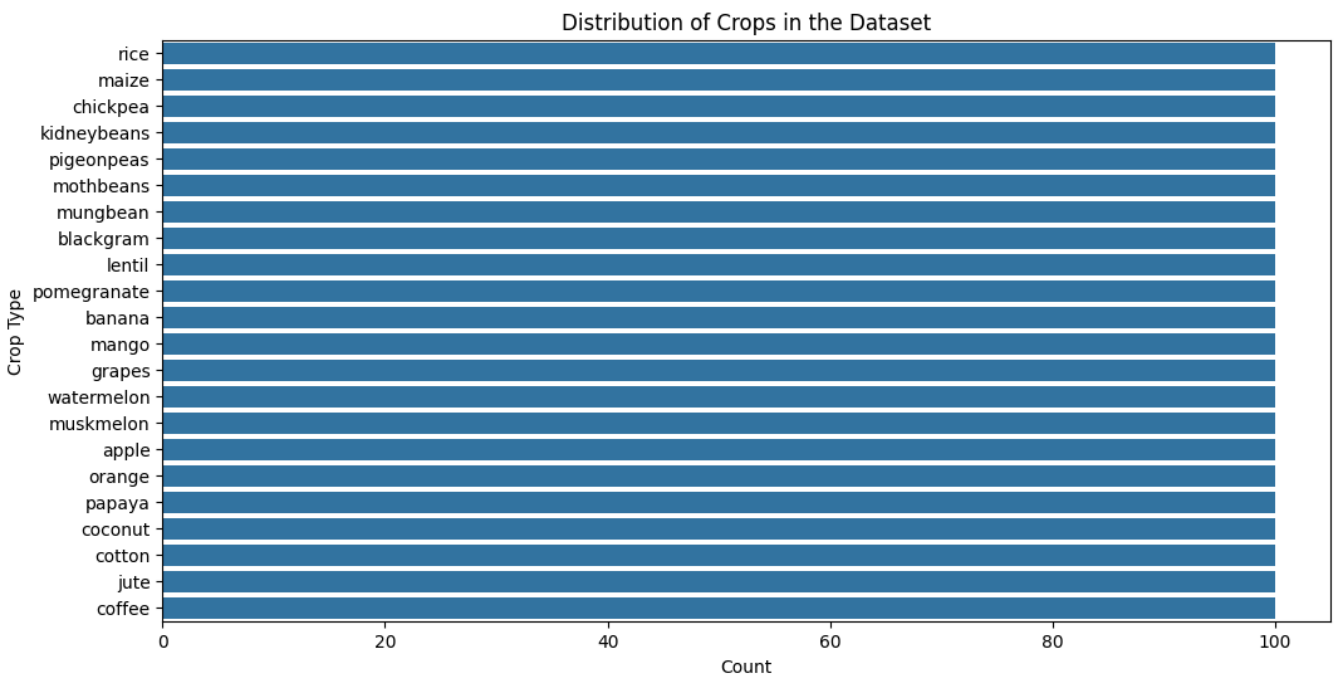
```
# Join based on crop name → get N, P, K requirements
merged_df = pd.merge(crop_df, npk_df, left_on="label", right_on="Crop", how="inner")

# Final merged dataset now has: soil features + weather + crop + NPK dose
print("Merged Dataset shape:", merged_df.shape)
merged_df.head()
```

Merged Dataset shape: (662000, 18)

	N	P	K	temperature	humidity	ph	rainfall	label	Nitrogen	Phosphorus	Potassium	Temperature	Humidity	pH_Value
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice	87.620827	78.747651	36.766993	34.950714	39.889324	8.382785
1	90	42	43	20.879744	82.002744	6.502985	202.935536	rice	44.583617	74.757998	39.330644	35.539910	45.471577	6.604554
2	90	42	43	20.879744	82.002744	6.502985	202.935536	rice	55.693347	68.562436	26.109595	22.077554	41.300125	5.822878
3	90	42	43	20.879744	82.002744	6.502985	202.935536	rice	108.417928	55.032441	93.475459	13.140251	70.035060	8.070740
4	90	42	43	20.879744	82.002744	6.502985	202.935536	rice	138.617285	43.615511	92.821651	31.276385	81.805868	5.085083

```
plt.figure(figsize=(12, 6))
sns.countplot(data=crop_df, y='label', order=crop_df['label'].value_counts().index)
plt.title('Distribution of Crops in the Dataset')
plt.xlabel('Count')
plt.ylabel('Crop Type')
plt.show()
```



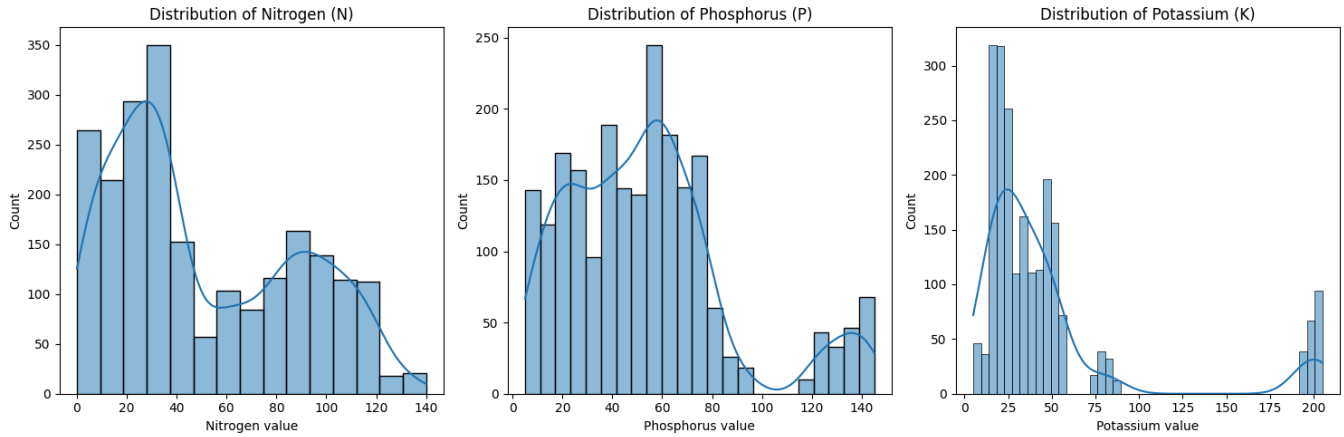

```
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
sns.histplot(crop_df['N'], kde=True)
plt.title('Distribution of Nitrogen (N)')
plt.xlabel('Nitrogen value')

plt.subplot(1, 3, 2)
sns.histplot(crop_df['P'], kde=True)
plt.title('Distribution of Phosphorus (P)')
plt.xlabel('Phosphorus value')

plt.subplot(1, 3, 3)
sns.histplot(crop_df['K'], kde=True)
plt.title('Distribution of Potassium (K)')
plt.xlabel('Potassium value')

plt.tight_layout()
plt.show()
```




```
print(fert_df.columns.tolist())
```

```
['Temparature', 'Humidity', 'Moisture', 'Soil Type', 'Crop Type', 'Nitrogen', 'Potassium', 'Phosphorous', 'Fertilizer Name']
```

```
# Strip spaces
fert_df.columns = fert_df.columns.str.strip()

# Features and Target
X_cls = fert_df[['Temparature','Humidity','Moisture','Soil Type','Crop Type',
                 'Nitrogen','Potassium','Phosphorous']]
y_cls = fert_df['Fertilizer Name']



# Train-Test Split
X_train_cls, X_test_cls, y_train_cls, y_test_cls = train_test_split(X_cls, y_cls, test_size=0.2, random_state=42)

# Encode categorical features
from sklearn.preprocessing import LabelEncoder
for col in ['Soil Type','Crop Type']:
    le = LabelEncoder()
    X_train_cls[col] = le.fit_transform(X_train_cls[col])
    X_test_cls[col] = le.transform(X_test_cls[col])

# Train model
clf_model = RandomForestClassifier(random_state=42)
clf_model.fit(X_train_cls, y_train_cls)

# Evaluate
y_pred_cls = clf_model.predict(X_test_cls)
print("Classification Accuracy:", accuracy_score(y_test_cls, y_pred_cls))
```

```
Classification Accuracy: 0.95
```

```
print(fert_df.columns.tolist())
```

```
['Temparature', 'Humidity', 'Moisture', 'Soil Type', 'Crop Type', 'Nitrogen', 'Potassium', 'Phosphorous', 'Fertilizer Name']
```

```
# Train Regression Model (Predict NPK dose)
X_reg = merged_df[['N','P','K','temperature','humidity','ph','rainfall']]
y_reg = merged_df[['Nitrogen','Phosphorus','Potassium']]  # required NPK dose

# Train-Test Split for Regression
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_reg, y_reg, test_size=0.2, random_state=42)

# Regression Model
reg_model = RandomForestRegressor(n_estimators=100, random_state=42)
reg_model.fit(X_train_reg, y_train_reg)

# Train Classification Model (Predict Fertilizer Name)
# Using the encoded features from the previous data preprocessing step
X_cls = fert_df[['Temparature', 'Humidity', 'Moisture', 'Soil Type', 'Crop Type', 'Nitrogen', 'Potassium', 'Phosphorous']]
y_cls = fert_df['Fertilizer Name']

# Train-Test Split for Classification
X_train_cls, X_test_cls, y_train_cls, y_test_cls = train_test_split(X_cls, y_cls, test_size=0.2, random_state=42)
```

```
# Classification Model
cls_model = RandomForestClassifier(n_estimators=100, random_state=42)
cls_model.fit(X_train_cls, y_train_cls)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
import joblib

joblib.dump(reg_model, "fertilizer_regression.pkl")
joblib.dump(cls_model, "fertilizer_classification.pkl")
```

```
['fertilizer_classification.pkl']
```

Start coding or generate with AI.