Aim

Machine learning project for crop yield prediction using climate and soil data. The dataset used "https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset",we have to apply at least two machine learning or deep learning models, and perform a comparative analysis of the models using appropriate performance metrics.

Load and explore the data

Reasoning: The first step is to load the dataset from the provided URL and perform initial exploration. This involves reading the data into a pandas DataFrame, displaying the head, printing info, and generating descriptive statistics.

```
import pandas as pd

# Load the dataset
df = pd.read_csv('/content/Crop_recommendation.csv')

# Display the first 5 rows
print("First 5 rows of the DataFrame:")
display(df.head())

# Print column names and their data types and get a summary of the DataFrame
print("\nDataFrame Info:")
df.info()

# Descriptive statistics for numerical columns
print("\nDescriptive statistics for numerical columns:")
display(df.describe())
```

```
First 5 rows of the DataFrame:
```

	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

```
DataFrame 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

Descriptive statistics for numerical columns:
```

	N	P	K	temperature	humidity	ph	rainfall
count	2200.000000	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	103.463655
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.958389
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.211267
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.551686
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.867624
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.267508
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.560117

Preprocess the data

Task:

Handle missing values, encode categorical variables, scale numerical features, and split the data into training and testing sets.

Reasoning: Check for missing values in the DataFrame.

```
print("Missing values before handling:")
print(df.isnull().sum())
```

```
Missing values before handling:
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

Reasoning: Since there are no missing values, the next step is to identify categor
categorical columns, and scale the numerical columns. Then split the data into training and testing sets.

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
import pandas as pd # Import pandas here

# Load the dataset (added this step)
try:
    df = pd.read_csv('/content/Crop_recommendation.csv')
    print("Dataset loaded successfully for preprocessing.")
except FileNotFoundError:
    print("Error: Crop_recommendation.csv not found. Please ensure the dataset is uploaded.")
    df = None

if df is not None:
    # Identify features (X) and target (y)
    X = df.drop('label', axis=1)
    y = df['label']

    # Identify categorical and numerical columns
    # Assuming 'label' is the only non-numerical and non-feature column if not dropped
    # If other object columns exist, they will be handled by LabelEncoder if in X
    categorical_cols = X.select_dtypes(include='object').columns
    numerical_cols = X.select_dtypes(exclude='object').columns

    # Encode categorical variables using Label Encoding and store the encoder
    label_encoder = LabelEncoder() # Store the encoder
    y_encoded = label_encoder.fit_transform(y)


    # Scale numerical features and store the scaler
    scaler = StandardScaler() # Store the scaler
    X[numerical_cols] = scaler.fit_transform(X[numerical_cols])

    # Split data into training and testing sets
    X_train, X_test, y_train_encoded, y_test_encoded = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

    print("Data preprocessing complete.")
    print("Shape of X_train:", X_train.shape)
    print("Shape of X_test:", X_test.shape)
    print("Shape of y_train_encoded:", y_train_encoded.shape)
    print("Shape of y_test_encoded:", y_test_encoded.shape)
else:
    print("Data loading failed. Cannot proceed with preprocessing.")
```

```
Dataset loaded successfully for preprocessing.
Data preprocessing complete.
Shape of X_train: (1760, 7)
Shape of X_test: (440, 7)
Shape of y_train_encoded: (1760,)
Shape of y_test_encoded: (440,)
```

Select and implement machine learning models

Task:

Choose at least two relevant ML/DL models for crop yield prediction and implement them.

Reasoning: Import the necessary libraries for the selected models.

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
```

Reasoning: Instantiate the selected models with default parameters.

```
# Instantiate Logistic Regression model
logistic_regression_model = LogisticRegression()
```

```
# Instantiate Decision Tree model
decision_tree_model = DecisionTreeClassifier()

print("Logistic Regression model instantiated:", logistic_regression_model)
print("Decision Tree model instantiated:", decision_tree_model)
```

```
Logistic Regression model instantiated: LogisticRegression()
Decision Tree model instantiated: DecisionTreeClassifier()
```

Train and evaluate the models

Task:

Train the selected models on the training data and evaluate their performance using appropriate metrics.

Reasoning: Train the selected models (Logistic Regression and Decision Tree) on the training data, make predictions on the test data, and then evaluate their performance using accuracy score and classification report.

```
from sklearn.metrics import accuracy_score, classification_report

# Train Logistic Regression model and make predictions
logistic_regression_model.fit(X_train, y_train_encoded)
logistic_regression_predictions = logistic_regression_model.predict(X_test)

# Train Decision Tree model and make predictions
decision_tree_model.fit(X_train, y_train_encoded)
decision_tree_predictions = decision_tree_model.predict(X_test)

# Evaluate Logistic Regression model
print("Logistic Regression Model Evaluation:")
print("Accuracy:", accuracy_score(y_test_encoded, logistic_regression_predictions))
print("Classification Report:\n", classification_report(y_test_encoded, logistic_regression_predictions))

# Evaluate Decision Tree model
print("\nDecision Tree Model Evaluation:")
print("Accuracy:", accuracy_score(y_test_encoded, decision_tree_predictions))
print("Classification Report:\n", classification_report(y_test_encoded, decision_tree_predictions))
```

```
Logistic Regression Model Evaluation:
Accuracy: 0.9636363636363636
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        23
           1       1.00      1.00      1.00        21
           2       0.90      0.95      0.93        20
           3       1.00      1.00      1.00        26
           4       1.00      1.00      1.00        27
           5       0.94      1.00      0.97        17
           6       0.94      1.00      0.97        17
           7       1.00      1.00      1.00        14
           8       0.83      0.87      0.85        23
           9       0.95      0.95      0.95        20
          10       0.85      1.00      0.92        11
          11       1.00      0.95      0.98        21
          12       1.00      1.00      1.00        19
          13       1.00      0.92      0.96        24
          14       1.00      1.00      1.00        19
          15       1.00      1.00      1.00        17
          16       1.00      1.00      1.00        14
          17       0.96      0.96      0.96        23
          18       0.91      0.87      0.89        23
          19       1.00      1.00      1.00        23
          20       0.88      0.79      0.83        19
          21       1.00      1.00      1.00        19

    accuracy                           0.96       440
   macro avg       0.96      0.97      0.96       440
weighted avg       0.96      0.96      0.96       440
```

```
Decision Tree Model Evaluation:
Accuracy: 0.9863636363636363
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        23
           1       1.00      1.00      1.00        21
           2       0.95      1.00      0.98        20
           3       1.00      1.00      1.00        26
           4       0.96      1.00      0.98        27
           5       1.00      1.00      1.00        17
           6       1.00      1.00      1.00        17
           7       1.00      1.00      1.00        14
           8       0.92      0.96      0.94        23
           9       1.00      1.00      1.00        20
          10       0.92      1.00      0.96        11
          11       1.00      0.95      0.98        21
          12       1.00      1.00      1.00        19
          13       1.00      0.92      0.96        24
          14       1.00      1.00      1.00        19
          15       1.00      1.00      1.00        17
          16       1.00      1.00      1.00        14
          17       1.00      1.00      1.00        23
          18       1.00      1.00      1.00        23
```

Perform comparative analysis

Task:

Compare the performance of the different models based on the evaluation metrics.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder

# Load the dataset
try:
    df = pd.read_csv('Crop_recommendation.csv')
except FileNotFoundError:
    print("Error: Crop_recommendation.csv not found. Please ensure the dataset is uploaded.")
    df = None

if df is not None:
    # Identify features (X) and target (y)
    X = df.drop('label', axis=1)
    y = df['label']

    # Identify categorical and numerical columns
    categorical_cols = X.select_dtypes(include='object').columns
    numerical_cols = X.select_dtypes(exclude='object').columns

    # Encode categorical variables using Label Encoding
    for col in categorical_cols:
        le = LabelEncoder()
        X[col] = le.fit_transform(X[col])

    # Scale numerical features
    scaler = StandardScaler()
    X[numerical_cols] = scaler.fit_transform(X[numerical_cols])

    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    print("Data preprocessing complete.")
    print("Shape of X_train:", X_train.shape)
    print("Shape of X_test:", X_test.shape)
    print("Shape of y_train:", y_train.shape)
    print("Shape of y_test:", y_test.shape)
else:
    print("Data loading failed. Cannot proceed with preprocessing and m
```

```
Data preprocessing complete.
Shape of X_train: (1760, 7)
Shape of X_test: (440, 7)
Shape of y_train: (1760,)
Shape of y_test: (440,)
```

Perform comparative analysis retry 1

Task:

Compare the performance of the different models based on the evaluation metrics.

Reasoning: Calculate and print the accuracy and classification report for both models, then provide a comparison based on these metrics.

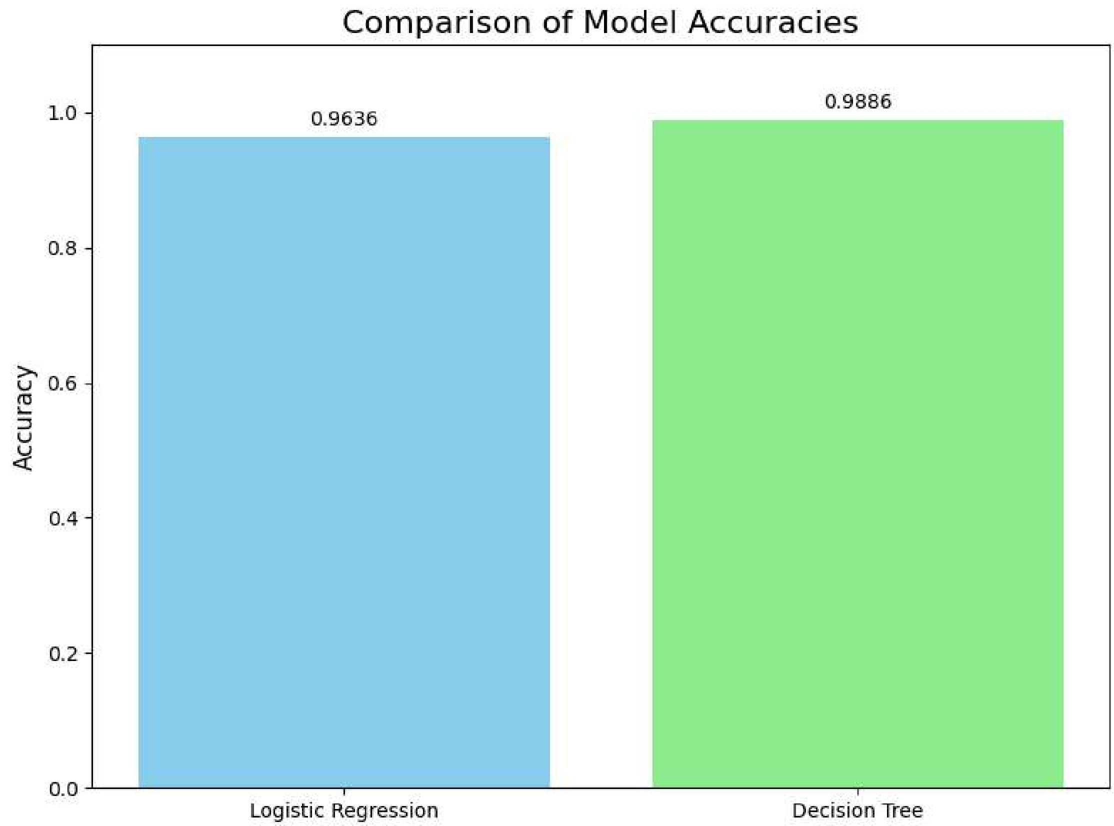
```
from sklearn.metrics import accuracy_score, classification_report

# Calculate accuracy for Logistic Regression
logistic_regression_accuracy = accuracy_score(y_test_encoded, logistic_regression_predictions)
print(f"Logistic Regression Model Accuracy: {logistic_regression_accuracy:.4f}")

# Print classification report for Logistic Regression
print("\nLogistic Regression Model Classification Report:")
print(classification_report(y_test_encoded, logistic_regression_predictions))

# Calculate accuracy for Decision Tree
decision_tree_accuracy = accuracy_score(y_test_encoded, decision_tree_predictions)
print(f"\nDecision Tree Model Accuracy: {decision_tree_accuracy:.4f}")

# Print classification report for Decision Tree
print("\nDecision Tree Model Classification Report:")
print(classification_report(y_test_encoded, decision_tree_predictions))

# Comparative analysis
print("\n--- Model Comparison ---")
if logistic_regression_accuracy > decision_tree_accuracy:
    print("Logistic Regression performed better than Decision Tree in terms of overall accuracy.")
elif decision_tree_accuracy > logistic_regression_accuracy:
    print("Decision Tree performed better than Logistic Regression in terms of overall accuracy.")
else:
    print("Both models performed similarly in terms of overall accuracy.")

print("\nFurther analysis of precision, recall, and F1-score for specific classes can be done by examining the classif
```

```
Logistic Regression Model Accuracy: 0.9636

Logistic Regression Model Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        23
           1       1.00      1.00      1.00        21
           2       0.90      0.95      0.93        20
           3       1.00      1.00      1.00        26
           4       1.00      1.00      1.00        27
           5       0.94      1.00      0.97        17
           6       0.94      1.00      0.97        17
           7       1.00      1.00      1.00        14
           8       0.83      0.87      0.85        23
           9       0.95      0.95      0.95        20
          10       0.85      1.00      0.92        11
          11       1.00      0.95      0.98        21
          12       1.00      1.00      1.00        19
          13       1.00      0.92      0.96        24
          14       1.00      1.00      1.00        19
          15       1.00      1.00      1.00        17
          16       1.00      1.00      1.00        14
          17       0.96      0.96      0.96        23
          18       0.91      0.87      0.89        23
          19       1.00      1.00      1.00        23
          20       0.88      0.79      0.83        19
```

```
          21       1.00      1.00      1.00        19

    accuracy                           0.96       440
   macro avg       0.96      0.97      0.96       440
weighted avg       0.96      0.96      0.96       440


Decision Tree Model Accuracy: 0.9886

Decision Tree Model Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        23
           1       1.00      1.00      1.00        21
           2       0.95      1.00      0.98        20
           3       1.00      1.00      1.00        26
           4       1.00      1.00      1.00        27
           5       1.00      1.00      1.00        17
           6       1.00      1.00      1.00        17
           7       1.00      1.00      1.00        14
           8       0.92      0.96      0.94        23
           9       1.00      1.00      1.00        20
          10       0.92      1.00      0.96        11
          11       1.00      1.00      1.00        21
          12       1.00      1.00      1.00        19
          13       1.00      0.92      0.96        24
          14       1.00      1.00      1.00        19
          15       1.00      1.00      1.00        17
          16       1.00      1.00      1.00        14
          17       1.00      1.00      1.00        23
          18       1.00      1.00      1.00        23
```

Data Visualizations

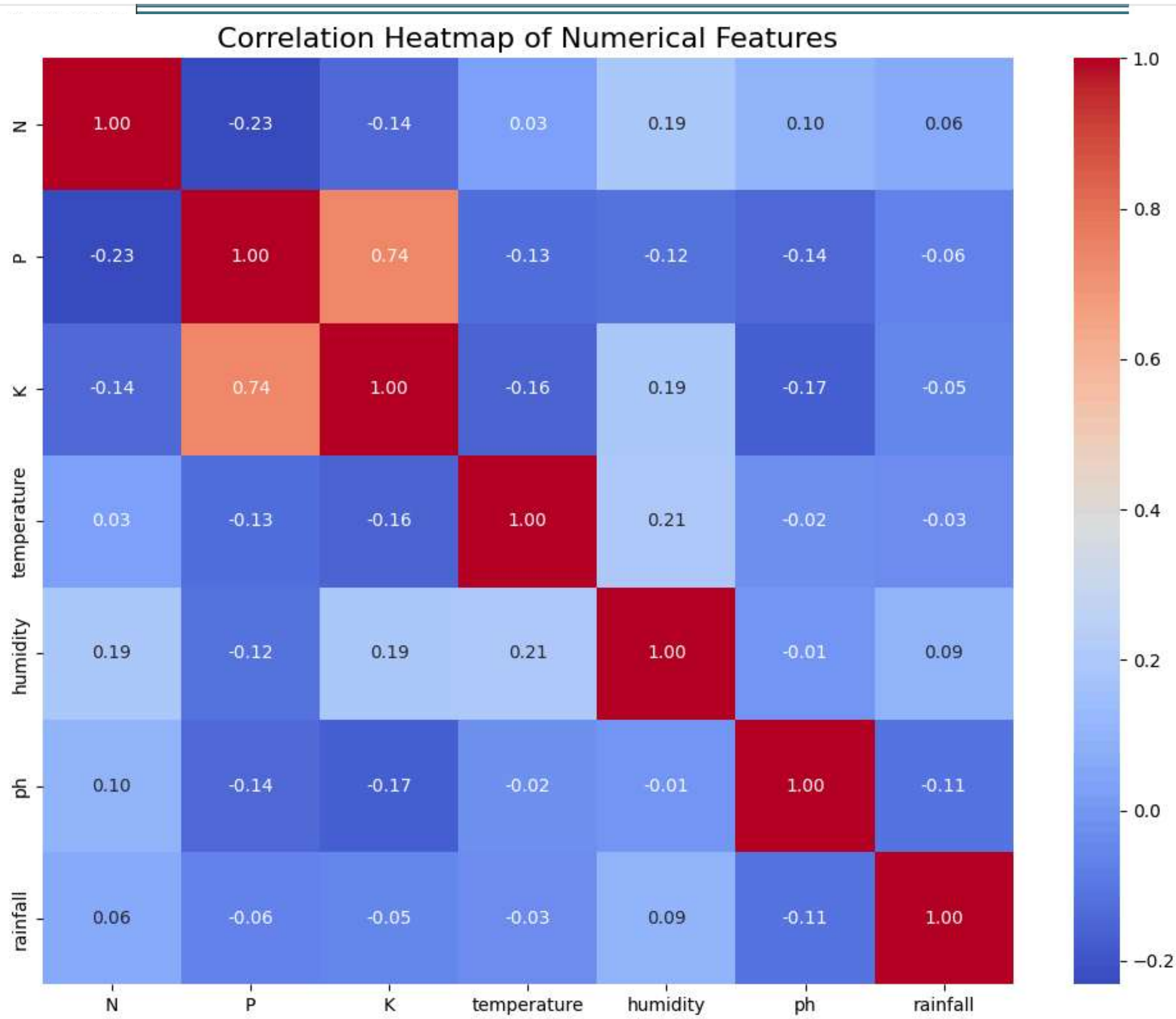
```python
import matplotlib.pyplot as plt
import numpy as np

# Model names and their accuracies
models = ['Logistic Regression', 'Decision Tree']
accuracies = [logistic_regression_accuracy, decision_tree_accuracy]

# Create a bar chart
fig, ax = plt.subplots(figsize=(8, 6))
bars = ax.bar(models, accuracies, color=['skyblue', 'lightgreen'])

# Add accuracy values on top of the bars
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.01, round(yval, 4), ha='center', va='bottom')

# Add title and labels
plt.title('Comparison of Model Accuracies', fontsize=16)
plt.ylabel('Accuracy', fontsize=12)
plt.ylim(0, 1.1) # Set y-axis limit to better visualize differences
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)

# Show the plot
plt.tight_layout()
plt.show()
```




```
import seaborn as sns
import matplotlib.pyplot as plt

# Count plot of the target variable (crop types)
plt.figure(figsize=(12, 6))
sns.countplot(data=df, y='label', order = df['label'].value_counts().index, palette='viridis')
plt.title('Distribution of Crop Types', fontsize=16)
plt.xlabel('Count', fontsize=12)
plt.ylabel('Crop Type', fontsize=12)
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-2517369638.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Correlation heatmap of numerical features
plt.figure(figsize=(10, 8))
sns.heatmap(df[numerical_cols].corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Numerical Features', fontsize=16)
plt.tight_layout()
plt.show()
```



Predict crop yield

Task:

Predict crop yield using the best-performing model on new data.

Reasoning: Create a pandas DataFrame with new data, scale the numerical features using the previously fitted scaler, use the best model to predict the encoded crop labels, and then use the label encoder to crop names.

```
# Create a DataFrame with new data
new_data = pd.DataFrame({
    'N': [80, 90],
    'P': [40, 45],
    'K': [40, 50],
    'temperature': [25.0, 28.0],
    'humidity': [60.0, 65.0],
    'ph': [6.5, 7.0],
    'rainfall': [150.0, 160.0]
})

# Apply the same scaling transformation to the numerical features of the new data
# Ensure the columns are in the same order as during training
new_data_scaled = scaler.transform(new_data[numerical_cols])

# Predict the encoded crop yield using the best model
predicted_encoded_labels = best_model.predict(new_data_scaled)

# Transform the predicted encoded labels back into original crop names
predicted_crop_yield = label_encoder.inverse_transform(predicted_encoded_labels)

# Print the predicted crop yields
print("Predicted crop yields for the new data:")
print(predicted_crop_yield)
```

```
Predicted crop yields for the new data:
['coffee' 'coffee']
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature na
  warnings.warn(
```

Summary: