AI CROP PREDICTION

PROJECT REPORT

Submitted in fulfilment for the JComponent of

Soft Computing (SWE1011)

CAL Course
in
M.Tech. – Software Engineering

by

Nancy Singh(22MIS0027)
Kartik Koacher(22MIS0166)
Ashish Singh(22MIS0173)

Under the guidance of

Dr. S. Hemalatha
SCORE

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science Engineering & Information Systems

Winter Semester 2024-25

TABLE OF CONTENTS

Abstract

Agricultural productivity and sustainability are crucially dependent on timely crop yield estimation and plant health diagnostics. This project leverages soft computing techniques, primarily Artificial Neural Networks (ANNs), to develop a robust and scalable system for predicting crop yield based on environmental and agricultural attributes such as region, season, crop type, and area cultivated. Using a real-world dataset from Karnataka, categorical features are one-hot encoded and combined with numerical features before being passed through a deep feedforward ANN trained using backpropagation and the Adam optimizer.

Additionally, Convolutional Neural Networks (CNNs) are employed in a complementary module for plant disease classification from leaf images. This model extracts spatial features using convolutional and pooling layers before classification through dense layers, providing a visual assessment of crop health. The combined use of ANN for yield prediction and CNN for disease detection introduces a hybrid approach that aligns with precision agriculture goals.

The system is evaluated using R^2 score for regression and accuracy for classification, achieving promising results in both domains. This multi-faceted approach demonstrates the potential of neural networks in automating key decisions in agricultural management, contributing to increased efficiency and reduced resource wastage.

1. Introduction

Agriculture plays a vital role in sustaining economies and ensuring food security, especially in developing countries like India where a significant portion of the population relies on farming for livelihood. However, modern agriculture faces several challenges such as improper crop selection, unpredictable yields due to changing climatic conditions, and the rapid spread of plant diseases. To address these issues and improve decision-making in agriculture, this project proposes a dual-model solution that integrates machine learning and deep learning techniques to support farmers with intelligent recommendations and disease diagnosis.

This project is divided into two major modules:

1.1 Crop Recommendation and Yield Prediction System

The first part of the project focuses on the implementation of an AI-powered system that recommends the most suitable crops for cultivation based on environmental and soil conditions. Furthermore, it estimates the potential yield of the selected crop using location-specific and seasonal data. Key features of this system include:

Input Parameters: Soil nutrient levels (Nitrogen, Phosphorus, Potassium), temperature, humidity, pH value, and rainfall.

Crop Recommendation: An Artificial Neural Network (ANN) model is trained to suggest the optimal crop for a given set of input conditions.

Yield Prediction: Using a Random Forest Regressor and other machine learning models, the system forecasts the expected crop yield based on factors like state, district, crop type, and season.

By providing data-driven insights, this module empowers farmers to make informed decisions that align with both environmental conditions and market demands.

1.2 CNN-Based Plant Disease Detection

The second module targets the early and accurate detection of plant diseases through image analysis. A Convolutional Neural Network (CNN) is trained on a large dataset of labeled images covering multiple crops and disease classes. Once trained, the model can analyze a photograph of a plant leaf and identify the type of disease (if any) it is affected by. Key highlights include:

Deep Learning Architecture: A custom CNN model that learns to differentiate between healthy and diseased leaves through multiple convolutional and pooling layers.

Image Processing: The system uses a preprocessing pipeline to normalize and resize images to the required input dimensions for the CNN.

Prediction and Label Mapping: The model's output is mapped to specific class labels corresponding to the diseases learned during training.

This module aids farmers in diagnosing diseases in real time, thereby reducing crop damage and minimizing reliance on expert consultation or trial-and-error methods.

2. Dataset Specification

To implement our comprehensive agriculture intelligence system, we utilize multiple datasets from Kaggle for tasks like crop recommendation, crop prediction, crop yield estimation, fertilizer recommendation, and plant disease classification.

1. Crop Recommendation Dataset

- **Description**: This dataset contains soil and environmental attributes like nitrogen (N), phosphorus (P), potassium (K), temperature, humidity, pH, and rainfall. The goal is to recommend the most suitable crop based on these inputs.
- **Preprocessing Steps**:
    - Handle missing values if any.
    - Normalize continuous features using standard or min-max scaling.
    - Encode categorical crop labels using label encoding or one-hot encoding.
- **Features Extracted**:
    - Numerical soil and environmental attributes.
    - Target variable: Crop label (e.g., rice, maize, cotton).

2. Crop Yield Prediction Dataset

- **Description**: This dataset includes features such as state, district, crop year, season, crop type, area under cultivation, and production. The target is to predict the yield (production/area).
- **Preprocessing Steps**:
    - One-hot encode categorical variables (state, district, season, crop).
    - Scale numerical variables like area and production.
- **Features Extracted**:
    - Encoded categorical features (state, season, etc.).
    - Area of cultivation.
    - Target: Crop yield (derived as production per unit area).

3. Fertilizer Recommendation Dataset

- **Description**: Contains environmental conditions (temperature, humidity, soil moisture), soil type, crop type, and current NPK levels to recommend suitable fertilizers.
- **Preprocessing Steps**:

- Encode soil and crop types into numerical values.
- Normalize continuous variables.

- **Features Extracted**:
  - Environmental features and soil nutrients (N, P, K).
  - Categorical features: Soil and crop types.
  - Target: Fertilizer name.

4. Crop Prediction Dataset (State-wise Crop Data)

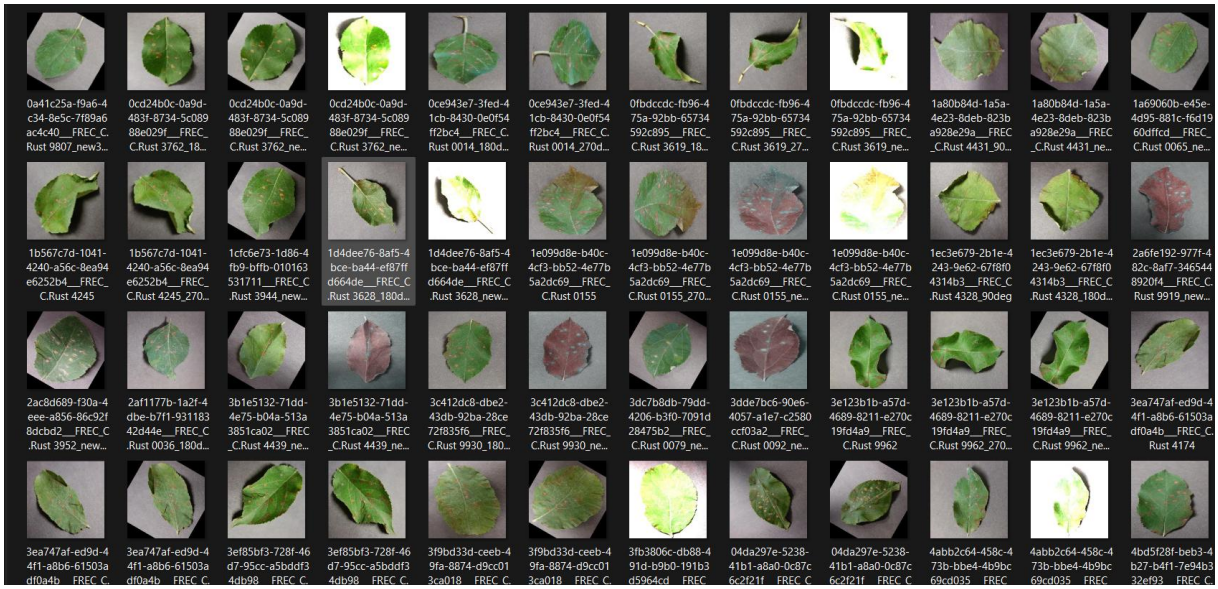
- **Description**: Contains information about crops grown in specific districts and seasons across Indian states. Helps in predicting potential crops based on region and season.
- **Preprocessing Steps**:
  - Encode categorical fields: State_Name, District_Name, Season.
  - Encode the target variable (Crop).
- **Features Extracted**:
  - Regional and seasonal indicators.
  - Target crop category.

5. Plant Disease Classification Dataset

- **Description**: Sourced from Kaggle - New Plant Diseases Dataset, this dataset includes thousands of labelled plant leaf images across multiple crop types and disease categories. Used for training a Convolutional Neural Network (CNN).
- **Preprocessing Steps**:
  - Resize all images to 100x100 pixels.
  - Normalize pixel values to [0, 1].
  - One-hot encode image labels.
- **Features Extracted**:
  - Pixel intensity arrays as input features.
  - Deep visual features learned through CNN layers.

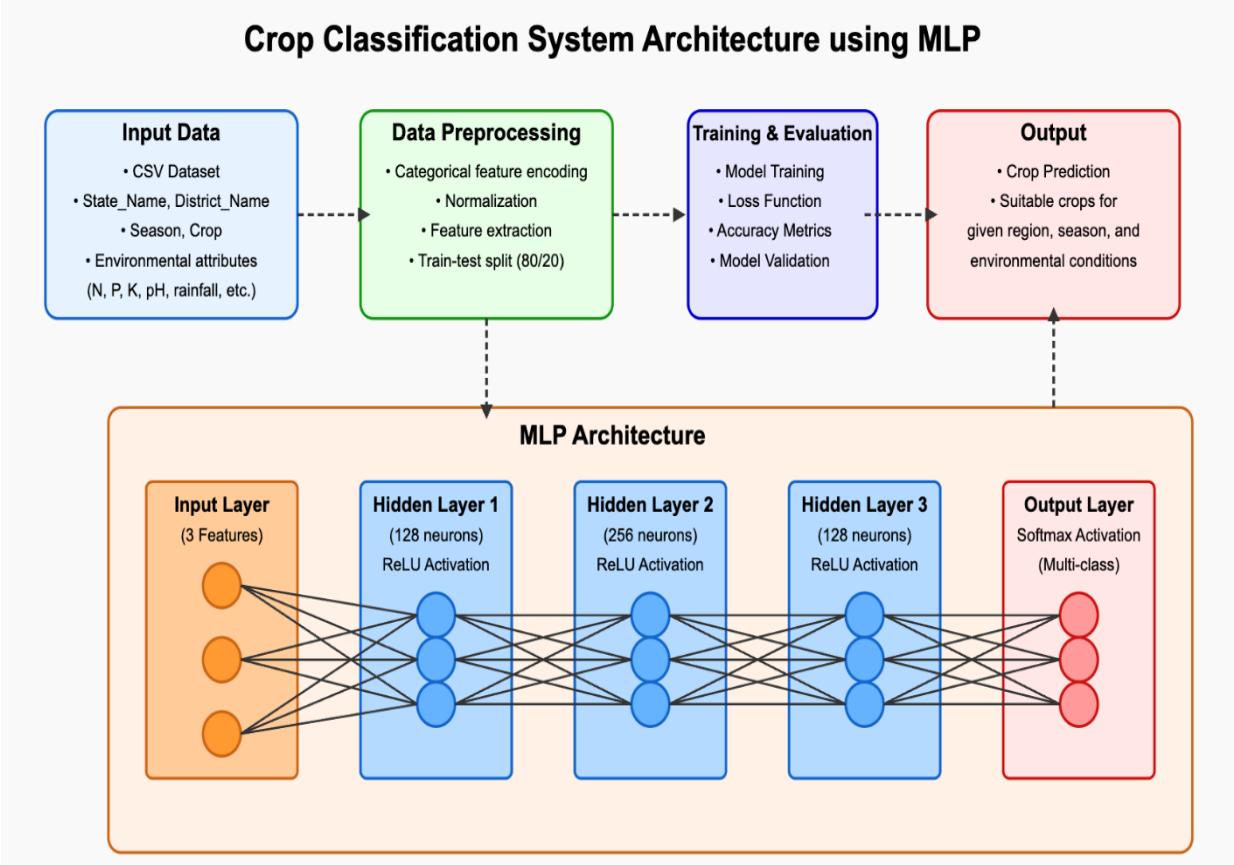
	A	B	C	D	E
1		State_Name	District_Name	Season	Crop
2	0	Andaman and Nicobar Islands	NICOBARS	Kharif	Arecanut
3	1	Andaman and Nicobar Islands	NICOBARS	Kharif	Other Kharif pulses
4	2	Andaman and Nicobar Islands	NICOBARS	Kharif	Rice
5	3	Andaman and Nicobar Islands	NICOBARS	Whole Year	Banana
6	4	Andaman and Nicobar Islands	NICOBARS	Whole Year	Cashewnut
7	5	Andaman and Nicobar Islands	NICOBARS	Whole Year	Coconut
8	6	Andaman and Nicobar Islands	NICOBARS	Whole Year	Dry ginger
9	7	Andaman and Nicobar Islands	NICOBARS	Whole Year	Sugarcane
10	8	Andaman and Nicobar Islands	NICOBARS	Whole Year	Sweet potato
11	9	Andaman and Nicobar Islands	NICOBARS	Whole Year	Tapioca
12	10	Andaman and Nicobar Islands	NICOBARS	Kharif	Arecanut
13	11	Andaman and Nicobar Islands	NICOBARS	Kharif	Other Kharif pulses
14	12	Andaman and Nicobar Islands	NICOBARS	Kharif	Rice
15	13	Andaman and Nicobar Islands	NICOBARS	Whole Year	Cashewnut
16	14	Andaman and Nicobar Islands	NICOBARS	Whole Year	Coconut
17	15	Andaman and Nicobar Islands	NICOBARS	Whole Year	Dry ginger
18	16	Andaman and Nicobar Islands	NICOBARS	Whole Year	Sugarcane
19	17	Andaman and Nicobar Islands	NICOBARS	Whole Year	Sweet potato
20	18	Andaman and Nicobar Islands	NICOBARS	Kharif	Rice
21	19	Andaman and Nicobar Islands	NICOBARS	Whole Year	Arecanut
22	20	Andaman and Nicobar Islands	NICOBARS	Whole Year	Banana
23	21	Andaman and Nicobar Islands	NICOBARS	Whole Year	Black pepper
24	22	Andaman and Nicobar Islands	NICOBARS	Whole Year	Cashewnut
25	23	Andaman and Nicobar Islands	NICOBARS	Whole Year	Coconut
26	24	Andaman and Nicobar Islands	NICOBARS	Whole Year	Dry chillies
27	25	Andaman and Nicobar Islands	NICOBARS	Whole Year	Dry ginger

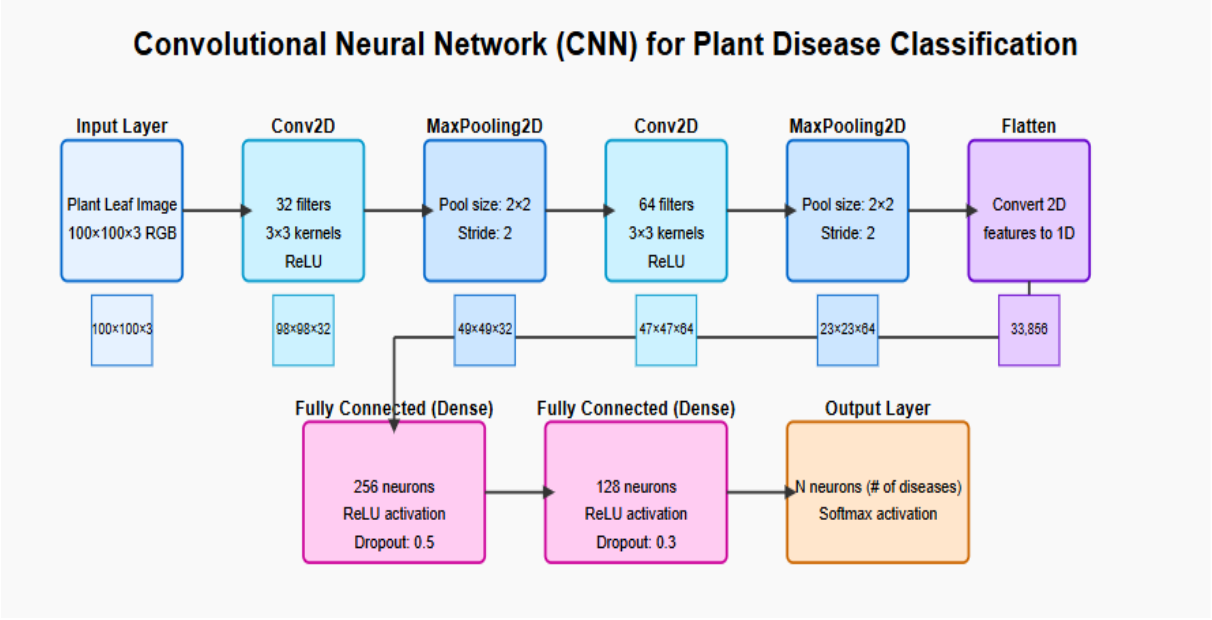
	A	B	C	D	E	F	G	H
1	N	P	K	temperatu	humidity	ph	rainfall	label
2	90	42	43	20.87974	82.00274	6.502985	202.9355	rice
3	85	58	41	21.77046	80.31964	7.038096	226.6555	rice
4	60	55	44	23.00446	82.32076	7.840207	263.9642	rice
5	74	35	40	26.4911	80.15836	6.980401	242.864	rice
6	78	42	42	20.13017	81.60487	7.628473	262.7173	rice
7	69	37	42	23.05805	83.37012	7.073454	251.055	rice
8	69	55	38	22.70884	82.63941	5.700806	271.3249	rice
9	94	53	40	20.27774	82.89409	5.718627	241.9742	rice
10	89	54	38	24.51588	83.53522	6.685346	230.4462	rice
11	68	58	38	23.22397	83.03323	6.336254	221.2092	rice
12	91	53	40	26.52724	81.41754	5.386168	264.6149	rice
13	90	46	42	23.97898	81.45062	7.502834	250.0832	rice
14	78	58	44	26.8008	80.88685	5.108682	284.4365	rice
15	93	56	36	24.01498	82.05687	6.984354	185.2773	rice
16	94	50	37	25.66585	80.66385	6.94802	209.587	rice
17	60	48	39	24.28209	80.30026	7.042299	231.0863	rice
18	85	38	41	21.58712	82.78837	6.249051	276.6552	rice
19	91	35	39	23.79392	80.41818	6.97086	206.2612	rice
20	77	38	36	21.86525	80.1923	5.953933	224.555	rice
21	88	35	40	23.57944	83.5876	5.853932	291.2987	rice
22	89	45	36	21.32504	80.47476	6.442475	185.4975	rice
23	76	40	43	25.15746	83.11713	5.070176	231.3843	rice
24	67	59	41	21.94767	80.97384	6.012633	213.3561	rice
25	83	41	43	21.05254	82.6784	6.254028	233.1076	rice
26	98	47	37	23.48381	81.33265	7.375483	224.0581	rice
27	66	53	41	25.07564	80.52389	7.778915	257.0039	rice



3. Experimental set-up

3.1. Architecture Diagram





3.2. Algorithms/Techniques used

- **Multilayer Perceptron (MLP) – Crop Recommendation**

  An MLP is a feedforward neural network that can learn complex patterns from structured data. It was used here to recommend suitable crops based on input parameters such as:

  - Nitrogen (N), Phosphorus (P), Potassium (K)
  - Temperature, Humidity, pH, Rainfall

  Key Techniques:

  - Label Encoding + One-Hot Encoding for crop labels
  - StandardScaler for input normalization
  - **Architecture:**
    - Input Layer: 7 neurons
    - Hidden Layers: [128 → 64 → 32] with ReLU & Dropout (0.3)
    - Output Layer: Softmax over crop classes
  - **Loss Function**: Categorical Crossentropy
  - **Optimizer**: Adam
  - **Training**: 100 epochs, early stopping (patience=10), validation split=0.2

- **Convolutional Neural Network (CNN) – Plant Disease Classification**

  A custom CNN (vanila CNN) was designed for classifying plant diseases from leaf images. CNNs are ideal for extracting spatial features from images.

  Key Techniques:

  - **Image Preprocessing:** Resizing to 128×128, normalization, label encoding
  - **Architecture:**
  - **3 Convolutional Blocks:** Conv2D → ReLU → MaxPooling
  - Flatten → Dense(128) → Dropout(0.5) → Softmax Output
  - **Loss Function:** Categorical Crossentropy
  - **Optimizer:** Adam
  - **Training:** ~20–30 epochs with validation split and optional augmentation

4. Software Requirement Specifications

4.1 Operating System

- Development OS: Windows 10 / 11 (64-bit) or Ubuntu 20.04+

4.2 Programming Language

- Python 3.8+ – Primary language for data processing, model building, and inference.

4.3 Libraries and Frameworks

For Crop Recommendation System (MLP-based)

- NumPy – Numerical computations
- Pandas – Data manipulation and analysis
- Scikit-learn – Data preprocessing (LabelEncoder, StandardScaler), train-test splitting
- TensorFlow / Keras – Building and training the MLP model
- Joblib – Saving and loading the trained model, encoders, and scaler

For Plant Disease Classification System (CNN-based)

- TensorFlow / Keras – CNN architecture creation and training
- OpenCV / PIL – Image preprocessing (resizing, normalization)
- Matplotlib / Seaborn – Visualizing model performance (optional)
- Augmentor / ImageDataGenerator – (Optional) for data augmentation during training

4.4 Development Tools

- Jupyter Notebook / VS Code / PyCharm – For writing and testing Python scripts
- Google Colab – (Optional) for training on GPU
- Git – Version control

4.5 Model Saving Format

- MLP Model: crop_model.h5
- CNN Model: plant_disease_model.h5
- Encoders/Scalers: label_encoder.pkl, scaler.pkl

5. Experimental Results & Discussion

5.1. Sample Code

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import joblib
import matplotlib.pyplot as plt
import seaborn as sns

# Load data
print("Loading dataset...")
data = pd.read_csv('preprocessed2.csv')
data['Season'] = data['Season'].str.strip()

# Drop unnecessary columns
if 'Unnamed: 0' in data.columns:
    data.drop(columns=['Unnamed: 0'], inplace=True)

# Encode input categorical variables
print("Encoding input features...")
state_encoder = LabelEncoder()
district_encoder = LabelEncoder()
season_encoder = LabelEncoder()

data['State_Encoded'] = state_encoder.fit_transform(data['State_Name'])
data['District_Encoded'] = district_encoder.fit_transform(data['District_Name'])
data['Season_Encoded'] = season_encoder.fit_transform(data['Season'])
```

```
# Encode target (crop)
print("Encoding target variable...")
crop_encoder = LabelEncoder()
data['Crop_Encoded'] = crop_encoder.fit_transform(data['Crop'])

X = data[['State_Encoded', 'District_Encoded', 'Season_Encoded']].values
y = data['Crop_Encoded'].values
num_classes = len(np.unique(y))

# Split dataset
print("Splitting dataset...")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train MLPClassifier
print("Training MLP model...")
mlp = MLPClassifier(hidden_layer_sizes=(128, 256, 128),          # 3 hidden layers
                    activation='relu',                           # ReLU activation
                    solver='adam',                               # Adam optimizer
                    max_iter=300,                                # Max 300 iterations
                    batch_size=32,                               # Mini-batch size
                    verbose=True,                                # Log training progress
                    random_state=42)

mlp.fit(X_train, y_train)

# Make predictions
y_pred = mlp.predict(X_test)
```

```
# Make predictions
y_pred = mlp.predict(X_test)

# Evaluation
accuracy = accuracy_score(y_test, y_pred)
print(f"\nTest Accuracy: {accuracy:.4f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=crop_encoder.classes_))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=False, cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.tight_layout()
plt.savefig("mlp_confusion_matrix.png")
plt.show()

# Save the model and encoders
print("Saving model and encoders...")
joblib.dump(mlp, 'mlp_crop_model.pkl')
joblib.dump({
    'state_encoder': state_encoder,
    'district_encoder': district_encoder,
    'season_encoder': season_encoder,
    'crop_encoder': crop_encoder
}, 'mlp_crop_encoders.pkl')

print("Model and encoders saved successfully!")
```

FOR CNN

```
# Define the CNN model
model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(100, 100, 3)))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())

model.add(layers.Dense(128, activation='relu'))

model.add(layers.Dropout(0.5))  # Optional dropout layer for regularization

model.add(layers.Dense(num_classes, activation='softmax'))  # Adjust num_classes based on your data

# Compile the model (excluded for brevity)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

5.2. Screenshots with Explanation

- **Crop Prediction**

```
def print_leaf(counts):
    total = sum(counts.values()) * 1.0
    probs = {}
    for lbl in counts.keys():
        probs[lbl] = str(int(counts[lbl] / total * 100)) + "%"
    return probs

def classify(row, node):
    if isinstance(node, Leaf):
        return node.predictions
    if node.question.match(row):
        return classify(row, node.true_branch)
    else:
        return classify(row, node.false_branch)

dt_model_final = joblib.load(r'filetest2.pkl')

# state = input("Enter state name: ")
# district = input("Enter district name: ")
# season = input("Enter season: ")
state='Uttar Pradesh'
district='Noida'
season='Rabi'
testing_data = [[state, district, season]]

for row in testing_data:
    Predict_dict = print_leaf(classify(row, dt_model_final))

count=0
for key, value in Predict_dict.items():
    if count>2:
        break
    print(key)
    print("  ,  ")
    count+=1
```

Output (Predicted crop for Rabi season in Uttar Pradesh Noida)

```
Barley
  ,
Gram
  ,
Linseed
  ,
```

The code implements a crop prediction system using a Multi-Layer Perceptron (MLP) classifier from scikit-learn. Below is a step-by-step explanation:

1. Dataset Loading & Preprocessing
    - Loads a preprocessed CSV file containing features such as State_Name, District_Name, Season, and Crop.
    - Removes unnecessary columns and strips white spaces from categorical fields.
2. Label Encoding
    - Encodes categorical features (State, District, Season) using LabelEncoder to

convert them into numeric values.

- The target variable Crop is also label encoded for model training.

3. Data Splitting
    - Splits the encoded dataset into training and testing sets using an 80:20 ratio.
4. Model Building & Training
    - Initializes an MLPClassifier with a hidden layer architecture of (128, 256, 128) and trains it with the training data.
    - Uses ReLU activation and the Adam optimizer with a maximum of 300 iterations.
5. Model Evaluation
    - Predicts on test data and evaluates performance using accuracy, classification report, and a confusion matrix.
    - Visualizes the confusion matrix using seaborn heatmap.
6. Model Saving
    - Saves the trained model and the encoders using joblib for future use.
7. Architecture Summary
    - Prints the input features, output classes, hidden layers, weights, and biases of the MLP network.

Similarly For Crop Recommendation

Input

```
def get_input_params():
    # n = int(input("Enter nitrogen (N): "))
    # p = int(input("Enter phosphorus (P): "))
    # k = int(input("Enter potassium (K): "))
    # temp = float(input("Enter temperature: "))
    # humidity = float(input("Enter humidity: "))
    # ph = float(input("Enter pH: "))
    # rainfall = float(input("Enter rainfall: "))
    n = 32
    p = 28
    k = 34
    temp = 35
    humidity = 80
    ph = 5
    rainfall = 223
    return np.array([[n, p, k, temp, humidity, ph, rainfall]])
```

Output

```
  warnings.warn(
1/1 [==============================] - 3s 3s/step
Predicted Crop: coconut
```

For Fertilizer Recommendation

```
def get_user_input():
    # n = int(input("Enter Nitrogen content: "))
    # p = int(input("Enter Phosphorus content: "))
    # k = int(input("Enter Potassium content: "))
    # temp = int(input("Enter Temperature: "))
    # humidity = int(input("Enter Humidity: "))
    # moisture = int(input("Enter Soil moisture: "))
    # soil = input("Enter Soil type: ")
    # crop = input("Enter Crop type: ")
    n= 32
    p= 30
    k= 38
    temp= 30
    humidity= 70
    moisture= 38
    soil= "loamy"
    crop= "wheat"
    return temp, humidity, moisture, soil, crop, n, k, p
```

Output

```
  warnings.warn(
1/1 [==============================] - 2s 2s/step
Recommended Fertilizer: 10-26-26
```

For plant Disease Detection

Input Given (TomatoYellowCurlVirus)

Output

```
# Path to user's image
user_image_path = "archive/test/test/TomatoYellowCurlVirus1.JPG"

# Target size used in CNN model (update according to your model input)
target_size = (100, 100)  # or (64, 64), etc., based on what your model was trained with

# Make predictions on user's image
predicted_class_index = predict_disease(user_image_path, loaded_model, target_size)

# Get the class label corresponding to the predicted index
class_directories = sorted([d for d in os.listdir(train_dir) if os.path.isdir(os.path.join(train_dir, d))])
class_label = class_directories[predicted_class_index]

print("Predicted Disease:", class_label)
```

```
[3]  ✓ 0.4s

...  1/1 [==============================] - 0s 107ms/step
     Predicted Disease: Tomato___Tomato_Yellow_Leaf_Curl_Virus
```

The following Python code demonstrates the prediction phase of a plant disease classification model using a trained Convolutional Neural Network (CNN). It loads a saved model, preprocesses the input image, performs prediction, and maps the result to a disease label.

1. Model Loading: The trained CNN model is loaded using load_model('model.h5'), where 'model.h5' is the saved model file.
2. Image Preprocessing: The preprocess_image() function converts the image to a fixed target size (100x100 pixels in this case), normalizes the pixel values to the [0,1] range, and reshapes it to match the model's expected input format (batch size, height, width, channels).
3. Prediction: The predict_disease() function performs inference using the CNN model and returns the index of the class with the highest prediction probability using np.argmax().

4. Label Mapping: To convert the predicted class index to a human-readable label, the code uses the folder names in the training dataset directory as class labels. The predicted index is matched to these folders using sorted(os.listdir(train_dir)).
5. Output: Finally, the predicted disease label is printed for the user-selected image.

This code segment allows for easy testing of plant disease classification using a single image and a pretrained deep learning model.