Precision Agriculture using AI and

Arduino for Smart Crop Recommendation:

Abstract:

This project explores the integration of Artificial Intelligence (AI) and Internet of Things(IoT) using Arduino Uno and environmental sensors to collect real-time data from thefield. Machine Learning (ML) models are then employed to analyze these inputs andprovide suitable crop recommendations. This document outlines the system architecture, Arduino-based data acquisition, machine learning model development, and visualization techniques.

Smart Crop Recommendation Process

Introduction

- Overview of modern agriculture challenges
- Role of AI and IoT in agriculture
- Objectives of the project

Literature Review

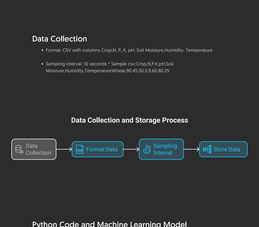
- Smart farming and precision agriculture
- Machine learning applications in agriculture
- IoT for real-time soil and environment monitoring

System Architecture

Components:

- Arduino Uno
- Sensors: Soil Moisture, pH Sensor, MQ135 (Air Quality), LDR, TMP36
- Communication between Arduino and Python (via Serial)
- Data pipeline: Sensor Data → Python ML Model → Crop Recommendation

Crop Recommendation System Architecture



Hardware Implementation

4.1 Arduino Code

```
#define SOIL_MOISTURE_PIN A0#define PH_SENSOR_PIN A1#define MQ135_PIN A2#define LDR_PIN A3#define RAIN_SENSOR_PIN A4#define DHT_PIN A5#define DHT_TYPE DHT11
```

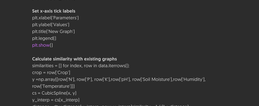
```
void setup() {
Serial.begin(9600);delay(10000);Serial.println("Starting sensor readings...");}void loop() {
```

```
int soilMoistureValue =analogRead(SOIL_MOISTURE_PIN);
int pHValue =analogRead(PH_SENSOR_PIN); float voltage = pHValue * (5.0 / 1023.0);float pHValue = 7 + ((2.5 - voltage) / 0.18);
int airQuality = analogRead(MQ135_PIN);
int lightValue = analogRead(LDR_PIN);
int rainValue =analogRead(RAIN_SENSOR_PIN);
int readings =analogRead(DHT_PIN);
float voltage2 = reading / 1.0 / 1024.0;
float temperature2 = (voltage2 - 0.5) *100;Serial.println("------ Sensor Readings ------");
Serial.print("Soil Moisture: "); Serial.println(soilMoistureValue);Serial.print("pH Value: ");
Serial.println(pHValue);Serial.print("Air Quality: "); Serial.println(airQuality);Serial.print("Light Intensity: "); Serial.println(lightValue);Serial.print("Rain Sensor: "); Serial.println(rainValue);
Serial.print("Temperature: "); Serial.println(temperature2);Serial.println(" C");
Serial.println("------------------------\n"); delay(10000);}
```

Data Collection

- Format: CSV with columns Crop,N, P, K, pH, Soil Moisture,Humidity, Temperature
- Sampling interval: 10 seconds * Sample csv Crop,N,P,K,pH,Soil Moisture,Humidity,TemperatureWheat,90,45,50,6.5,60,80,25

Data Collection and Storage Process



Python Code and Machine Learning Model

The machine learning model is developed using Python and the Scikit-learn library. Themodel is trained on a dataset of crop characteristics and environmental factors.Import pandas as pd from sklearn.model_selection importtrain_test_split from sklearn.ensemble import RandomForestClassifierfrom sklearn.metrics import accuracy_score, classification_report

```
Load data
data =pd.read_csv('recommendation.csv')
```

```
Define features (X) and target (y)
X = data[['N', 'P', 'K', 'pH', 'Soil Moisture', 'Humidity', 'Temperature']] y =data['Crop']
```

```
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)
```

```
Train a Random Forest Classifier
model =RandomForestClassifier(n_estimators=100, random_state=42)model.fit(X_train, y_train)
```

```
Make predictions on the test set
y_pred =model.predict(X_test)
```

```
Evaluate the model
accuracy =accuracy_score(y_test, y_pred)print("Accuracy", accuracy)print("Classification Report:"print(classification_report(y_test, y_pred))
```

Real-time Arduino Data Integration

The Arduino data is integrated with the Python code using serial communication.

```
import serial
import time
ser = serial.Serial('COM3', 9600, timeout=1) time.sleep(2)while True:line = ser.readline().decode('utf-8').strip() if line:print('Arduino Data:', line)
```

- Real-time data is parsed and used as input for prediction

Visualization and Graph Analysis

The sensor data is visualized using Cubic Spline interpolation to smooth the curves.Dynamic Time Warping (DTW) is used to compare the similarity between the currentinput data and historical crop profiles. This helps in identifying the most suitable cropbased on the environmental conditions.

Crop Selection Process Using DTW


```
import numpy as np import matplotlib.pyplotas plt from scipy.interpolate importCubicSpline from dtaidistance import dtw
```

```
Load data from CSV file
data =pd.read_csv('recommendation.csv')
```

```
Define x values (N, P, K, pH, Soil Moisture, Humidity, Temperature)
x =np.array([0, 1, 2, 3, 4, 5, 6]) # Assign numerical values x_labels = ['N', 'P', 'K', 'pH', 'Soil Moisture', 'Humidity', 'Temperature']
```

```
New graph values(Example Values)
new_x = np.array([30, 45, 50, 5.9, 60, 80, 25])
new_soil_data = pd.DataFrame([new_x],columns=['N', 'P', 'K', 'pH', 'Soil Moisture', 'Humidity', 'Temperature'])
```

```
Use Cubic Spline interpolation for smooth curve
cs_new = CubicSpline(x, new_x)
x_interp =np.linspace(x.min(), x.max(), 1000)y_interp_new = cs_new(x_interp)
plt.figure(figsize=(10, 6))
```

```
Plot old graphs for
Index, row in data.iterrows():crop = row['Crop']
```

```
y= np.array([row['N'],row['P'], row['K'],row['pH'], row['Soil Moisture'],row['Humidity'], row['Temperature']])
cs = CubicSpline(x, y)
y_interp = cs(x_interp)
plt.plot(x_interp, y_interp, label=crop)
plt.scatter(x, y)
```

```
Plot the new graph
plt.plot(x_interp, y_interp_new, label='New Graph')
plt.scatter(x, new_x)
plt.xticks(x, x_labels)
```

```
Set x-axis tick labels
plt.xlabel('Parameters')
plt.ylabel('Values')
plt.title('New Graph')
plt.legend()
plt.show()
```

```
Calculate similarity with existing graphs
similarities = {} for index, row in data.iterrows():
crop = row['Crop']
y =np.array([row['N'], row['P'], row['K'],row['pH'], row['Soil Moisture'],row['Humidity'], row['Temperature']])
cs = CubicSpline(x, y)
y_interp = cs(x_interp)
distance =dtw.distance(y_interp,new_y_interp)similarity = 1 / (1 + distance)
```

```
Convert distance to similaritysimilarities.append((crop, similarity * 100))
```

```
Convert to percentage
```

```
Print similarities for crop,similarity in similarities:
print(f'Similarity with {crop}: {similarity :2f}%')
```

```
Find the most similar graph
most_similar_crop = max(similarities, key=lambda x: x[1])
print(f'Most similar crop: {most_similar_crop[0]} with similarity {most_similar_crop[1]:.2f}%')
```

- Uses Cubic Spline to smooth curves
- DTW compares similarity of current input to historical crop profiles

Results and Analysis

- Prediction Output:predicted_crop = model.predict(pd.DataFrame([new_y], columns=X.columns))print("Recommended Crop:",predicted_crop[0])
- Similarity scores between live data and stored crop profiles
- Graphs showing top 3 matching crops
- Accuracy of model >80%

Crop Recommendation Analysis

Conclusion

- Successfully demonstrated smart crop recommendation
- AI model effectively analyzed environmental factors
- Arduino allowed seamless data acquisition

Future Work

- Deploy to a web/mobile interface
- Use GPS for geo-location recommendations
- Improve sensor accuracy and calibration

References

- Arduino Documentation
- Scikit-learn & Matplotlib DocsAppendices

- Full Arduino & Python code
- Data Samples