AI-FARM ASSISTANT

SUBMITTED BY: JINU JOJI

REG NO: 23BCE8495

Smart Farming with AI: Helping Farmers Make Better Decisions

This tool is here to help farm workers make better decisions about their crops by using artificial intelligence and the evaluation of information. By studying a wide number of factors such as soil moisture, temperature, humidity, and many other key conditions, the model accurately predicts the degree to which a crop is healthy or specifically recommends the best crops to grow. The system takes actual farm data, tidies it up, processes it, along with training a machine learning model (Random Forest Classifier) to recognize many patterns and create precise predictions. With this, farmers can:

Recognize when crops are unhealthy quickly and then act decisively to prevent wide-ranging harm. To cut down on waste, use water, fertilizers, along with other resources in an efficient way.

Make many informed decisions. Base these decisions on real-time farm conditions.

This tool helps farmers increase their yields to a great extent, substantially reduce risks, and make farming more sustainable and profitable to a large degree in the long run by using AI in agriculture.

**CODE:**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, accuracy\_score

import joblib

from sklearn.preprocessing import LabelEncoder

try:

    data = pd.read\_csv('Crop\_recommendationV2.csv')

    print("Dataset loaded successfully!")

except FileNotFoundError:

    print("Error: Dataset file 'Crop\_recommendationV2.csv' not found!")

    exit()

except pd.errors.EmptyDataError:

    print("Error: Dataset is empty!")

    exit()

except pd.errors.ParserError:

    print("Error: Dataset file is corrupt or improperly formatted!")

    exit()

print("Dataset Columns:", list(data.columns))

possible\_targets = ['crop\_health', 'label', 'growth\_stage']

target\_column = None

for col in possible\_targets:

    if col in data.columns:

        target\_column = col

        break

if target\_column is None:

    print("Error: No valid target variable found in dataset!")

    exit()

print(f"Using '{target\_column}' as the target variable.")

if data[target\_column].dtype == 'object':

    le = LabelEncoder()

    y = le.fit\_transform(data[target\_column])

else:

    y = data[target\_column]

numerical\_columns = data.select\_dtypes(include=['number']).columns.tolist()

exclude\_columns = [target\_column]  # Exclude target variable from features

feature\_columns = [col for col in numerical\_columns if col not in exclude\_columns]

if not feature\_columns:

    print("Error: No numerical features available for training!")

    exit()

X = data[feature\_columns]

X.fillna(X.median(), inplace=True)

try:

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

    print("Dataset split into training and testing sets.")

except ValueError as e:

    print(f"Error during train-test split: {e}")

    exit()

try:

    model = RandomForestClassifier(n\_estimators=100, random\_state=42)

    model.fit(X\_train, y\_train)

    print("Model training completed successfully!")

except Exception as e:

    print(f"Error during model training: {e}")

    exit()

try:

    y\_pred = model.predict(X\_test)

except Exception as e:

    print(f"Error during prediction: {e}")

    exit()

try:

    accuracy = accuracy\_score(y\_test, y\_pred)

    print(f"Accuracy: {accuracy:.2f}")

    print("Classification Report:\n", classification\_report(y\_test, y\_pred))

except Exception as e:

    print(f"Error during evaluation: {e}")

try:

    joblib.dump(model, 'farm\_assistant\_model.pkl')

    print("Model saved successfully as 'farm\_assistant\_model.pkl'.")

except Exception as e:

    print(f"Error while saving the model: {e}")

