|  |
| --- |
| Title: Crop Recommendation System Using Machine Learning |
| Done by : Pritesh Ram Keshri Project Number : 15, Submission Date: 17th July 2025 |
| **Overview:**  This project aims to recommend the most suitable crop to cultivate based on soil and weather conditions using a machine learning classification model. The model is trained on a labeled dataset containing key agricultural factors such as nutrient levels and environmental parameters, helping farmers make data-driven decisions. |
| **Libraries used:**   * **pandas, numpy –** Data manipulation and array handling * **scikit-learn –** Model building, evaluation, preprocessing * **joblib –** Model serialization * **streamlit –** Web interface for real-time predictions |
| **Dataset Details:**   * **Source:https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset(Crop\_recommendation.csv)** * **Features:** * N (Nitrogen) * P (Phosphorus) * K (Potassium) * temperature (in °C) * humidity (in %) * ph (pH value of soil) * rainfall (in mm) * **Target: Label (crop name like rice, maize, etc.)** |
| **APIs Integrated:**  No external APIs were used. |
| **Source code 1: File Name : Model Training (train.py)**  import pandas as pd  from sklearn.model\_selection import train\_test\_split  from sklearn.ensemble import RandomForestClassifier  from sklearn.preprocessing import LabelEncoder  from sklearn.metrics import accuracy\_score, classification\_report  import joblib  # Load dataset  df = pd.read\_csv("data/Crop\_recommendation.csv")  # Encode target labels  le = LabelEncoder()  df['label'] = le.fit\_transform(df['label'])  # Feature and target split  X = df.drop('label', axis=1)  y = df['label']  # Train-test split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # Model training  model = RandomForestClassifier(random\_state=42)  model.fit(X\_train, y\_train)  # Evaluation  y\_pred = model.predict(X\_test)  print("Accuracy:", accuracy\_score(y\_test, y\_pred))  print("\nClassification Report:\n", classification\_report(y\_test, y\_pred, target\_names=le.classes\_))  # Save model and encoder  joblib.dump(model, "crop\_model.pkl")  joblib.dump(le, "label\_encoder.pkl") |
| **Source code 2: Streamlit Web App (app.py)**  import streamlit as st  import joblib  import numpy as np  # Load model and label encoder  model = joblib.load("crop\_model.pkl")  le = joblib.load("label\_encoder.pkl")  # Page setup  st.set\_page\_config(page\_title="Crop Recommendation System", page\_icon="🌿", layout="centered")  # UI styling  st.markdown("""  <style>  html, body, [class\*="css"] {  font-family: 'Segoe UI', sans-serif;  }  .main-box {  background-color: #1e1e1e;  padding: 2rem;  border-radius: 12px;  color: #f0f0f0;  }  h1, h2, h3, .stMarkdown {  color: #f0f0f0 !important;  }  .stButton > button {  background-color: #28a745;  color: white;  font-weight: bold;  border-radius: 6px;  }  </style>  """, unsafe\_allow\_html=True)  # Title  st.title("🌿 Crop Recommendation System")  # Input form  with st.container():  with st.form("crop\_form"):  st.markdown('<div class="main-box">', unsafe\_allow\_html=True)  st.subheader("📋 Enter Soil & Weather Conditions")  col1, col2 = st.columns(2)  with col1:  N = st.number\_input("Nitrogen (N)", min\_value=0, max\_value=140, value=90)  P = st.number\_input("Phosphorus (P)", min\_value=5, max\_value=145, value=42)  K = st.number\_input("Potassium (K)", min\_value=5, max\_value=205, value=43)  ph = st.slider("pH Level", 3.0, 10.0, 6.5)  with col2:  temperature = st.slider("Temperature (°C)", 0.0, 50.0, 20.8)  humidity = st.slider("Humidity (%)", 10.0, 100.0, 82.0)  rainfall = st.slider("Rainfall (mm)", 20.0, 300.0, 202.9)  submitted = st.form\_submit\_button("🌾 Recommend Crop")  st.markdown('</div>', unsafe\_allow\_html=True)  # Prediction output  if submitted:  input\_data = np.array([[N, P, K, temperature, humidity, ph, rainfall]])  prediction = model.predict(input\_data)  predicted\_crop = le.inverse\_transform(prediction)[0]  st.success(f"✅ \*\*Recommended Crop:\*\* `{predicted\_crop}`")  st.balloons() |
| **Source code 3: Label Encoding & Model Saving Snippet**  from sklearn.preprocessing import LabelEncoder  import joblib  # Encode labels  le = LabelEncoder()  df['label'] = le.fit\_transform(df['label'])  # Train model...  # Save model and encoder  joblib.dump(model, "crop\_model.pkl")  joblib.dump(le, "label\_encoder.pkl") |
|  |
|  |
| **Output screenshots:**  **1.Crop recommended based on input:Rice-** Streamlit UI with input values recommending **rice.**    **2.** **Crop recommended based on input: Coffee:** Streamlit UI with input values recommending coffee.    **3. Model Accuracy -** Terminal output showing model training results and command-line prediction of rice. |
| **What you learned:**  Through this project, I gained practical knowledge of:   * Machine learning model development using real-world agri-data * Data preprocessing and label encoding * UI creation using Streamlit * Model evaluation and deployment using joblib * Practical application of classification algorithms in precision agriculture |
| **What the Skills you gained:**   * Data preprocessing and cleaning * Classification using RandomForest * Model serialization and loading * Building responsive ML web apps * Agricultural domain problem-solving using ML |
| **Real Time Applications:**   * Crop recommendation for farmers based on soil test reports * Agri-tech platforms or Kisan apps for smart farming decisions * Government or NGO tools for guiding sustainable agriculture |
| **Further Enhancement Suggestions :**   1. Add real-time weather API to auto-fetch live data for accurate predictions. 2. Include fertilizer recommendations based on soil nutrients and crop. 3. Deploy as a mobile app for better accessibility to farmers. 4. Support multiple languages to make it user-friendly across regions. |