|  |
| --- |
| Title: Fertilizer Recommendation System Using Machine Learning |
| Done by : Pritesh Ram Keshri Project Number : 16, Submission Date: 20th July 2025 |
| **Overview:**  This project implements a **Fertilizer Recommendation System** using machine learning techniques. Based on input soil parameters and environmental conditions, the system recommends the most suitable type of fertilizer. The goal is to assist farmers in selecting fertilizers that optimize crop yield while maintaining soil health.  The system includes separate modules for training the model, making predictions, defining model architecture, and utility functions. It is structured for scalability, modularity, and integration into agricultural tech platforms. |
| **Libraries used:**  **pandas, numpy –** Data manipulation and numerical operations  **scikit-learn –** Model building and evaluation  **joblib –** Saving and loading trained models  **argparse, os, json –** Command-line interface and configuration management |
| **Dataset Details:**  **Source**: Custom/Simulated dataset - https://www.kaggle.com/code/jshndeep/fertilizer-recommendation-system  **Format:** CSV file (e.g., fertilizer\_data.csv)  **Features:**   * Soil nutrients: Nitrogen (N), Phosphorus (P), Potassium (K) * Environmental parameters: Temperature, Humidity, Moisture * Soil Type and Crop Type (Categorical)   **Target:** Fertilizer Name (e.g., Urea, DAP, Potash) |
| **APIs Integrated:**  No external APIs were integrated. All training and prediction processes are executed locally through modular Python scripts |
| **Source code 1: App Configuration and Styling**  **File Name : Importing libraries, train.py**  import pandas as pd  import numpy as np  from sklearn.ensemble import RandomForestClassifier  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import LabelEncoder  import joblib  # Load dataset  df = pd.read\_csv("fertilizer\_data.csv")  # Encode categorical features  le\_crop = LabelEncoder()  le\_soil = LabelEncoder()  le\_fert = LabelEncoder()  df['Crop Type'] = le\_crop.fit\_transform(df['Crop Type'])  df['Soil Type'] = le\_soil.fit\_transform(df['Soil Type'])  df['Fertilizer Name'] = le\_fert.fit\_transform(df['Fertilizer Name'])  # Feature and target  X = df.drop("Fertilizer Name", axis=1)  y = df["Fertilizer Name"]  # Split data  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # Train model  model = RandomForestClassifier()  model.fit(X\_train, y\_train)  # Save model and encoders  joblib.dump(model, "fertilizer\_model.pkl")  joblib.dump(le\_crop, "crop\_encoder.pkl")  joblib.dump(le\_soil, "soil\_encoder.pkl")  joblib.dump(le\_fert, "fertilizer\_encoder.pkl") |
| **Source code 2: model.py**  import joblib  class FertilizerModel:  def \_\_init\_\_(self):  self.model = joblib.load("fertilizer\_model.pkl")  self.crop\_encoder = joblib.load("crop\_encoder.pkl")  self.soil\_encoder = joblib.load("soil\_encoder.pkl")  self.fertilizer\_decoder = joblib.load("fertilizer\_encoder.pkl")  def predict(self, features):  pred = self.model.predict([features])  return self.fertilizer\_decoder.inverse\_transform(pred)[0] |
| **Source code 3: predict.py**  import argparse  from model import FertilizerModel  import json  # CLI input  parser = argparse.ArgumentParser()  parser.add\_argument("--input", type=str, required=True, help="JSON input file")  args = parser.parse\_args()  # Load input  with open(args.input, "r") as f:  data = json.load(f)  features = [  data["N"],  data["P"],  data["K"],  data["temperature"],  data["humidity"],  data["moisture"],  data["soil\_type\_encoded"],  data["crop\_type\_encoded"]  ]  # Predict  model = FertilizerModel()  fertilizer = model.predict(features)  print("Recommended Fertilizer:", fertilizer) |
| **Source Code 4: utils.py**    from sklearn.preprocessing import LabelEncoder  def encode\_categories(df):  le\_crop = LabelEncoder()  le\_soil = LabelEncoder()  df['Crop Type'] = le\_crop.fit\_transform(df['Crop Type'])  df['Soil Type'] = le\_soil.fit\_transform(df['Soil Type'])  return df, le\_crop, le\_soil  def save\_encoders(le\_crop, le\_soil):  import joblib  joblib.dump(le\_crop, "crop\_encoder.pkl")  joblib.dump(le\_soil, "soil\_encoder.pkl") |
|  |
| **Output screenshots:**  Here , are the snapshots of my working project.  **1.** **Streamlit Web UI Output - This is the starting interface of my project.**    **2. Prediction Result:** Fertilizer recommendation shown in terminal.    **3. Logs or Visuals:** Correlation Heatmap |
| **What you learned:**  Through this project, I gained hands-on experience with:   * Developing an end-to-end recommendation system using ML * Data encoding and transformation for classification tasks * Model evaluation and result interpretation * Deploying a model using Python CLI structure |
| **What the Skills you gained:**   * ML classification with scikit-learn * Feature engineering and label encoding * Model serialization and usage in prediction * Modular project structuring with reusable components |
| **Real Time Applications:**   * Fertilizer advisory for farmers using soil test kits * Integration into agri-mobile apps or kiosks * Precision agriculture and sustainable farming platforms * Agri-tech solutions by NGOs and government portals |
| **Further Enhancement Suggestions:**   * Add weather API for real-time context * Integrate crop stage or season for better accuracy * Deploy as a web/mobile app for accessibility * Include multi-language support for farmer usability |