CODE :

```
# ------------------------------------------------------------
# AI POWERED YIELD PREDICTION ASSISTANT USING WEATHER
# FULL SINGLE PYTHON FILE (RUN THIS IN VS CODE)
# ------------------------------------------------------------

import pandas as pd
import numpy as np
import requests
import joblib
from flask import Flask, request, jsonify
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import os

# ------------------------------------------------------------
# 1) GENERATE SYNTHETIC DATA (You can replace with CSV)
# ------------------------------------------------------------

def generate_synthetic_data(n=300, seed=42):
    np.random.seed(seed)

    avg_temp = np.random.normal(25, 5, n)            # °C
    total_rain = np.abs(np.random.normal(200, 100, n))  # mm
```

```
    humidity = np.clip(np.random.normal(60, 15, n), 20, 100)
    soil_moisture = np.clip(np.random.normal(30, 8, n), 5, 80)
    fertilizer = np.random.randint(0, 2, n)  # 0 or 1

    # True yield formula (synthetic)
    crop_yield = (
        0.04 * avg_temp +
        0.01 * total_rain +
        0.03 * soil_moisture +
        0.5 * fertilizer +
        np.random.normal(0, 1.2, n)
    )

    df = pd.DataFrame({
        "avg_temp": avg_temp,
        "total_rain": total_rain,
        "humidity": humidity,
        "soil_moisture": soil_moisture,
        "fertilizer": fertilizer,
        "yield": crop_yield.round(2)
    })
    return df


# ------------------------------------------------------------
# 2) LOAD DATA (Use CSV if available)
# ------------------------------------------------------------
```

```
def load_data(csv_path="yield_data.csv"):
    if os.path.exists(csv_path):
        print("Loading data from CSV...")
        return pd.read_csv(csv_path)
    else:
        print("CSV not found — generating synthetic dataset...")
        df = generate_synthetic_data()
        df.to_csv(csv_path, index=False)
        return df



# ------------------------------------------------------------
# 3) TRAIN MODEL
# ------------------------------------------------------------

def train_model():
    df = load_data()

    X = df[["avg_temp", "total_rain", "humidity", "soil_moisture", "fertilizer"]]
    y = df["yield"]

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42
    )

    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
```

```
joblib.dump(scaler, "scaler.joblib")

model = RandomForestRegressor(n_estimators=120, random_state=42)
model.fit(X_train_scaled, y_train)

y_pred = model.predict(X_test_scaled)

# Metrics
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print("\nModel Performance:")
print(f"RMSE: {rmse:.3f}")
print(f"R2 Score: {r2:.3f}")

joblib.dump(model, "yield_model.joblib")
print("\nModel saved as yield_model.joblib")

# Plot Actual vs Predicted
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, color='blue')
mn, mx = min(y_test), max(y_test)
plt.plot([mn, mx], [mn, mx], 'r--', label="Ideal")
plt.xlabel("Actual Yield")
plt.ylabel("Predicted Yield")
plt.title("Actual vs Predicted Crop Yield")
plt.grid(True)
```

```
    plt.legend()
    plt.savefig("yield_plot.png")
    plt.show()

    print("Plot saved as yield_plot.png")



# ------------------------------------------------------------
# 4) WEATHER API FETCHER (OPTIONAL)
# ------------------------------------------------------------

def get_weather(city, api_key):
    """
    Fetch temperature, humidity, rainfall, wind from OpenWeatherMap API.
    """
    url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_key}&units=metric"

    r = requests.get(url)
    data = r.json()

    main = data.get("main", {})
    rain = data.get("rain", {}).get("1h", 0)

    return {
        "temp": main.get("temp"),
        "humidity": main.get("humidity"),
        "rain_mm": rain
```

```
    }


# ------------------------------------------------------------
# 5) FLASK API FOR REAL-TIME PREDICTION
# ------------------------------------------------------------

app = Flask(_name_)

# Load model + scaler
model = None
scaler = None

def load_model():
    global model, scaler
    model = joblib.load("yield_model.joblib")
    scaler = joblib.load("scaler.joblib")
    print("Model + Scaler Loaded Successfully!")


@app.route("/")
def home():
    return "AI Yield Prediction API Running!"


@app.route("/predict", methods=["POST"])
def predict_yield():
    data = request.json
```

```
    required = ["avg_temp", "total_rain", "humidity", "soil_moisture", "fertilizer"]

    for r in required:
        if r not in data:
            return jsonify({"error": f"Missing {r}"}), 400

    features = np.array([[data["avg_temp"],
                          data["total_rain"],
                          data["humidity"],
                          data["soil_moisture"],
                          data["fertilizer"]]])

    features_scaled = scaler.transform(features)
    pred = model.predict(features_scaled)[0]

    return jsonify({"predicted_yield": round(float(pred), 2)})


# ------------------------------------------------------------
# 6) MAIN RUNNER
# ------------------------------------------------------------

if _name_ == "_main_":
    print("\nTraining model...")
    train_model()

    print("\nLoading model for API...")
```

load_model()

print("\nStarting Flask API on http://127.0.0.1:5000")

app.run(debug=True)

OUT PUT :