```
import requests
import time
import matplotlib.pyplot as plt
import joblib
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from termcolor import colored

# Store readings for plotting11
last_readings = []



# Train the model (executed only once)
def train_model():
    # Example dataset (replace 'your_dataset.csv' with the actual dataset)
    data = pd.read_csv('Functioning_Dataset.csv')

    # Features and labels
    X = data[['Temperature', 'Voltage', 'Humidity']]
    y = data['State']  # 0: NORMAL, 1: ABNORMAL

    # 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)

    # Evaluate the model
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(colored(f"Model Accuracy: {accuracy * 100:.2f}%", "green"))

    # Save the trained model
    joblib.dump(model, 'threshold_classifier.pkl')



# Classify sensor data using the trained model
def classify_sensor_data(sensor_data, prev_voltage=None, prev_temp=None):
    model = joblib.load('threshold_classifier.pkl')

    # Check for drastic voltage change within 10 seconds
    voltage_status = colored("NORMAL", "green")
    if prev_voltage is not None and abs(sensor_data['voltage'] - prev_voltage) > 2:
        voltage_status = colored("FAULT - Potential Short-Circuit!", "red")

    # Check for excessive heat or low temperature
    temp_status = colored("NORMAL", "green")
    if sensor_data['temperature'] > 28:
        temp_status = colored("Excessive Heat (Adjust Panel Direction)", "red")
    elif sensor_data['temperature'] < 23:
        temp_status = colored("Low Temperature (Adjust Panel Direction)", "red")

    # Check for excessive humidity
    humidity_status = colored("NORMAL", "green")
    if sensor_data['humidity'] > 80:
        humidity_status = colored("Excessive Humidity (Longer Ignition Time)", "red")

    # Determine overall state
    if "FAULT" in voltage_status or "Adjust Panel" in temp_status or "Excessive Humidity" in humidity_status:
        state = colored("ABNORMAL", "red")
    else:
        state = colored("NORMAL", "green")

    return state, voltage_status, temp_status, humidity_status


def fetch_thingspeak_data():
    """Fetches data from Thingspeak."""
    params = {
        "api_key": "S10GZM6GJ1TCIFV4",
        "results": 1  # Get the latest entry
    }
    try:
        response = requests.get(f"https://api.thingspeak.com/channels/2754568/feeds.json",
params=params)
        response.raise_for_status()
        data = response.json()
        if "feeds" in data and data["feeds"]:
            latest_entry = data["feeds"][0]
            return {
                "temperature": float(latest_entry.get("field2", 0)),
                "voltage": float(latest_entry.get("field1", 0)),
                "humidity": float(latest_entry.get("field3", 0))
            }
    except Exception as e:
        print(colored(f"Error fetching data: {e}", "red"))
        return None


def get_manual_data():
    """Allows the user to manually input sensor values."""
    try:
        temperature = float(input("Enter temperature value (°C): "))
        voltage = float(input("Enter voltage value (V): "))
        humidity = float(input("Enter humidity value (%): "))
        return {"temperature": temperature, "voltage": voltage, "humidity": humidity}
    except ValueError:
        print(colored("Invalid input. Please enter numeric values.", "red"))
        return None


def plot_graph(readings):
    """Plots the last 5 readings."""
    if len(readings) < 5:
        print(colored("Not enough readings to plot. Need at least 5.", "yellow"))
        return

    timestamps = range(1, len(readings) + 1)
    temperatures = [r["temperature"] for r in readings]
    voltages = [r["voltage"] for r in readings]
    humidities = [r["humidity"] for r in readings]

    plt.figure(figsize=(10, 6))
    plt.plot(timestamps, temperatures, marker="o", label="Temperature (°C)")
    plt.plot(timestamps, voltages, marker="o", label="Voltage (V)")
    plt.plot(timestamps, humidities, marker="o", label="Humidity (%)")
    plt.xlabel("Readings")
    plt.ylabel("Values")
    plt.title("Sensor Data for Last 5 Readings")
    plt.legend()
    plt.grid()
    plt.show()


def main():
    # Train the model if not already trained
    try:
        model = joblib.load('threshold_classifier.pkl')
        print(colored("Loaded pre-trained model.", "cyan"))
    except FileNotFoundError:
        print(colored("Training model...", "yellow"))
        train_model()

    prev_voltage = None
    prev_temp = None

    try:
        while True:
            print(colored("\nSelect input method:", "cyan"))
            print(colored("1. Manual Input", "cyan"))
            print(colored("2. Fetch from Thingspeak", "cyan"))
            choice = input("Enter your choice (1 or 2): ")

            if choice == "1":
                for _ in range(5):
                    sensor_data = get_manual_data()
                    if sensor_data:
                        last_readings.append(sensor_data)
                        if len(last_readings) > 5:
                            last_readings.pop(0)
                        print(colored(f"\nSensor Data: {sensor_data}", "cyan"))
                        state, voltage_status, temp_status, humidity_status =
classify_sensor_data(sensor_data,
                                                      prev_voltage,
                                                      prev_temp)
                        print(f"State Analysis: {state}")
                        print(f"Voltage Status: {voltage_status}")
                        print(f"Temperature Status: {temp_status}")
                        print(f"Humidity Status: {humidity_status}")
            prev_voltage = sensor_data['voltage']
            prev_temp = sensor_data['temperature']
        plot_graph(last_readings)

      elif choice == "2":
        for _ in range(5):
          sensor_data = fetch_thingspeak_data()
          if sensor_data:
            last_readings.append(sensor_data)
            if len(last_readings) > 5:
              last_readings.pop(0)
            print(colored(f"\nSensor Data: {sensor_data}", "cyan"))
            state, voltage_status, temp_status, humidity_status =
classify_sensor_data(sensor_data,
                                                  prev_voltage,
                                                  prev_temp)
            print(f"State Analysis: {state}")
            print(f"Voltage Status: {voltage_status}")
            print(f"Temperature Status: {temp_status}")
            print(f"Humidity Status: {humidity_status}")
            prev_voltage = sensor_data['voltage']
            prev_temp = sensor_data['temperature']
          time.sleep(10)
        plot_graph(last_readings)

      else:
        print(colored("Invalid choice. Please select 1 or 2.", "red"))
        continue

  except KeyboardInterrupt:
    print(colored("Exiting program.", "yellow"))
```

```
if _name_ == "_main_":
   main()
```