Intelligent Delivery Drone Planner using Simulated Annealing and Perceptron Classification

Objective:

The goal of this project is to simulate a drone-based delivery system that selects the most efficient route using Simulated Annealing while avoiding locations with unsafe weather conditions predicted using a Perceptron classifier. Unsafe locations will increase the delivery cost.

Learning Outcomes:

By completing this project, students will learn to:

- Train and apply a Perceptron classifier.
- Use binary classification to influence decision-making in another algorithm.
- Implement Simulated Annealing for solving an optimization problem.
- Integrate machine learning into a real-world scenario.

Dataset Description:

You are provided with an Excel file: weather_data_linearly_separable.xlsx

- **Number of samples:** 200
- **Features:**
  - Temperature (°C)
  - Humidity (%)
  - Wind Speed (km/h)
- **Label:**
  - SafeToFly:
    - 0 → Safe to fly
    - 1 → Unsafe to fly

Project Tasks

1. Train a Perceptron Classifier

- Use the provided Excel data to train a Perceptron.
- Classify inputs as SafeToFly = 0 or 1.

2. Simulate Delivery Nodes

- Create N cities or delivery nodes (e.g., 10).
- For each city, assign:
  - Random weather values (or manually defined)

    - 2D coordinates (X, Y) to simulate geographic locations
- Predict whether each city is safe or unsafe using your Perceptron model.

3. Construct a Cost Matrix

- Use **Euclidean distance** between each pair of cities as the base cost.
- If a city is unsafe (SafeToFly = 1), add a penalty (e.g., +50) to the route cost for visiting it.

4. Apply Simulated Annealing

- Start with a random route visiting all cities.
- Apply Simulated Annealing to minimize total delivery cost.
- Cost function must consider both distance and weather-based penalty.

Expected Output (with GUI Requirement)

Your program must include a **Graphical User Interface (GUI)** that allows the user to interact with the system and view the delivery optimization process. The GUI should provide the following features:

1. User Input via GUI

The interface **must allow the user to enter**:

- The **number of delivery cities**.
- The **location (X, Y coordinates)** of each city.
- The **weather conditions** for each city:
    - Temperature (°C)
    - Humidity (%)
    - Wind Speed (km/h)
- The parameters required for **Simulated Annealing**, such as:
    - Initial temperature
    - Cooling rate

2. Visual Display of Routes

The GUI must graphically display:

- The **initial delivery route** (random order of cities)
    - Connected on a 2D plane with city labels.
    - Display the **total route distance** before optimization.
- The **final optimized delivery route** after applying Simulated Annealing
    - Show the new connection order.
    - Display the **new total optimized route distance**.
    - Optionally use a different colour or line style to highlight changes.

3. Weather Predictions Display

- Show for each city whether it is predicted as:
    - **Safe to Fly (0)** — can be shown in green or with a checkmark.
    - **Unsafe to Fly (1)** — shown in red or with a warning icon.

- Optionally, display the weather features for each city in a table or popup.

4. Route Cost Analysis

- Show a comparison between:
    - The **route cost before optimization**
    - The **route cost after optimization**
    - Including any **penalties** added for visiting unsafe cities

5. Optional Enhancements

- Allow the user to:
    - Modify weather values manually and re-run predictions
    - Modify the location of the city
    - Re-optimize the route with new Simulated Annealing parameters