# Overview

You are provided with a synthetic dataset of **100 polygons** (land parcels). Each polygon includes:

- A unique integer `parcel_id`
- A floating-point `carbon_store` attribute
- A floating-point `cost`

Your goal is to demonstrate proficiency in:

1. **Setting up a Python environment** with relevant geospatial libraries.
2. **Loading, inspecting, and transforming geospatial data** (coordinate systems, polygon manipulations).
3. **Performing spatial operations**.
4. **Formulating and solving a small optimization problem** with constraints based on budget and adjacency.
5. **Communicating your approach and results** clearly, via both code and a short report.

# The Data

- **File Name**: `land_parcels.shp` (plus its associated files)
- **CRS**: EPSG:4326 (WGS84)
- **Number of Polygons**: 100
- **Attributes**:
    - `parcel_id`
    - `carbon_store`
    - `cost`

# Your Tasks

1. **Environment Setup**
    - Create a new Python environment (conda/virtualenv/uv).
    - Install all required geospatial libraries, and an optimization library of your choice (e.g., `pulp`/`pyomo`).
    - Include explicit instructions in your `README.md` (e.g., an `environment.yml` or `requirements.txt`) so others can replicate your setup.
2. **Load & Inspect the Data**
    - Read the shapefile `land_parcels.shp` into a GeoDataFrame.
    - Summarize the dataset:
        - How many polygons?
        - Minimum, average, maximum area (in km²) of the polygons (after reprojecting, see next step).
        - Range of `carbon_store` and `cost` attributes.
3. **Reproject & Filter**

    - Reproject the polygons to a projected coordinate system EPSG:3347
    - Filter out polygons below a reasonable area threshold to remove any outliers. Count how many were removed. If none are below your threshold, you need not remove any.
4. **Compute Adjacency**
   Determine which polygons are "adjacent" in the sense of sharing **an edge** (not just a point).
    - Create and store an adjacency list (or adjacency matrix) for your polygons.
5. **Optimization Task**
    - **Objective**: Select a subset of polygons to **maximize** the total `carbon_store`.
    - **Constraint #1 (Budget)**: The total `cost` of the selected polygons must not exceed 50% of the sum of `cost` of *all* polygons (after filtering).
    - **Constraint #2 (No adjacency)**: No two polygons that share an edge may both be selected.
    - (Optional) **Constraint #3 (Area)**: The total area of the selected polygons must be at least 25% of the total area (after filtering).
    - Use a Python-based MILP solver (e.g., PuLP, Pyomo) to formulate and solve this problem.
    - Output the chosen polygons, total cost, and total carbon store.
6. **Results & Discussion**
    - Provide a short summary:
        - How many polygons were chosen?
        - What is the sum of `carbon_store` for the chosen set?
        - Total cost vs. budget?
        - Did you observe any interesting "edge cases" or quirks?

# What We're Looking For

1. **Correctness & Completeness:** Does your solution correctly handle the data's coordinate system, adjacency, and optimization constraints?
2. **Code Clarity & Organization:** Is your code modular and well-documented?
3. **Spatial Reasoning:** Did you properly handle the geometry/adjacency operations?
4. **Optimization Approach:** Are your decision variables, objective, and constraints set up correctly? Did you debug any solver or data issues successfully?
5. **Communication:** Is the final notebook (and report) clear, concise, and easy to follow?

# Submission

Please email your solution and include a link to a **public GitHub repository** containing:

1. **README.md**
    - Briefly explain the purpose of the project.
    - Provide clear setup instructions.
    - Include any references or notes if applicable.
2. **Jupyter Notebook** (e.g., `solution.ipynb`)
    - Document your step-by-step solution: data loading, adjacency checks, optimization approach, results, etc.
    - Include code cells and explanations so someone can replicate or follow along.

3. **Short Report**
    - You can integrate this into the same notebook at the end.
    - Summarize your approach, methodology, any challenges, key results, and your analysis of the solution.
4. **Final Output**
    - Clearly indicate which polygons were selected by your optimization model (plot them), their total cost, and total `carbon_store`.
    - Any relevant charts or tables you produced along the way are welcome.

# Note on AI Usage

**Completely using AI to generate your entire solution is discouraged.** In the interview, you will be asked to walk through your code in detail, so if you rely heavily on AI, you must still be able to thoroughly explain and justify every part of your solution. We also have AI detectors, so we will know if your solution is largely AI generated. However, using AI to generate snippets of code or troubleshoot errors is acceptable, provided you fully understand the resulting code and can discuss how it works.

You should ideally not need more than 4 hours to complete this assessment. Eitherway, you should submit the completed assessment via email by Sunday (March 23) 8 pm ET.

Good luck, and we look forward to reviewing your submission!