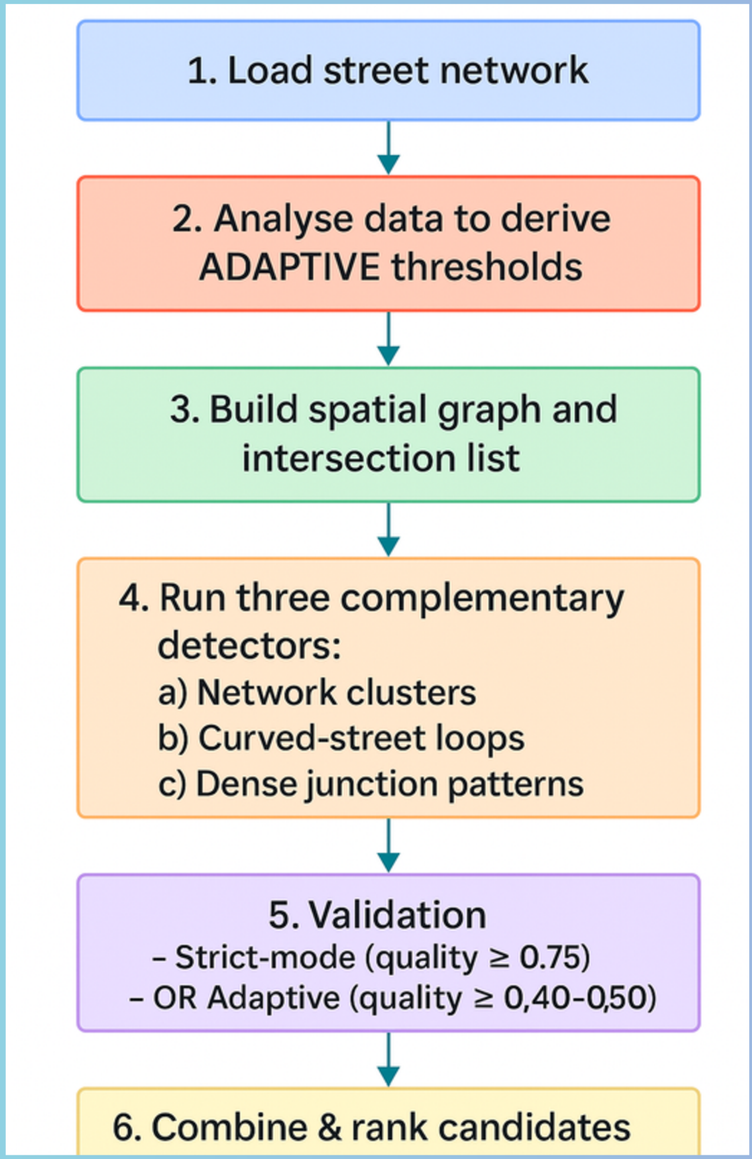
PS-2:

Detecting Missing Roundabouts for Database Integration.

WorkFlow:



Load Street Network

Import road and intersection data (e.g., from OpenStreetMap).

Analyze Data & Set Adaptive Thresholds
Calculate local thresholds based on street layout patterns.

Build Graph & Intersection List
Create a network graph of streets and list all junctions.
Run 3 Detectors
a) Network clusters
b) Curved-street loops
c) Dense junction areas

Validate Results
Use strict (≥ 0.75) or adaptive (≥ 0.40–0.50) quality checks.
Combine & Rank Candidates
Merge detector outputs, rank by quality, and optionally compare with known data.

Function	What it does	Brief Understanding
calculate_circularity	Measures how "round" a shape is using the formula 4π × Area ÷ (Perimeter²). A perfect circle gives 1.0.	Is the shape really a circle, or more like a weird blob? This tells us how close we are to a perfect roundabout.
minimum_enclosing_circle	Finds the smallest possible circle that covers a group of points.	It draws the tightest possible circle around a bunch of road points—handy for fitting a roundabout shape.
validate_street_approaches	Checks if enough different roads meet the candidate roundabout, and if they're spaced out enough in direction.	Makes sure it’s not just a driveway loop—looks for at least 3 roads coming into the circle from different angles.
is_likely_residential_area	Looks for many short road segments in the area, which usually means a cul-de-sac or housing loop.	Helps avoid mistakes by skipping areas that look like residential turnarounds, not real roundabouts.
validate_roundabout_geometry_strict	Runs all the checks above (circle shape, road approaches, area type) using strict rules.	This is the “tough judge”—only keeps roundabouts that look very correct and complete.
validate_roundabout_adaptive	Same as above, but uses flexible thresholds depending on the local road network.	This is the “more forgiving judge”—accepts good-enough roundabouts in places where roads are less regular.

Data-driven Threshold Estimation

analyze_data_characteristics(streets_gdf, intersections)

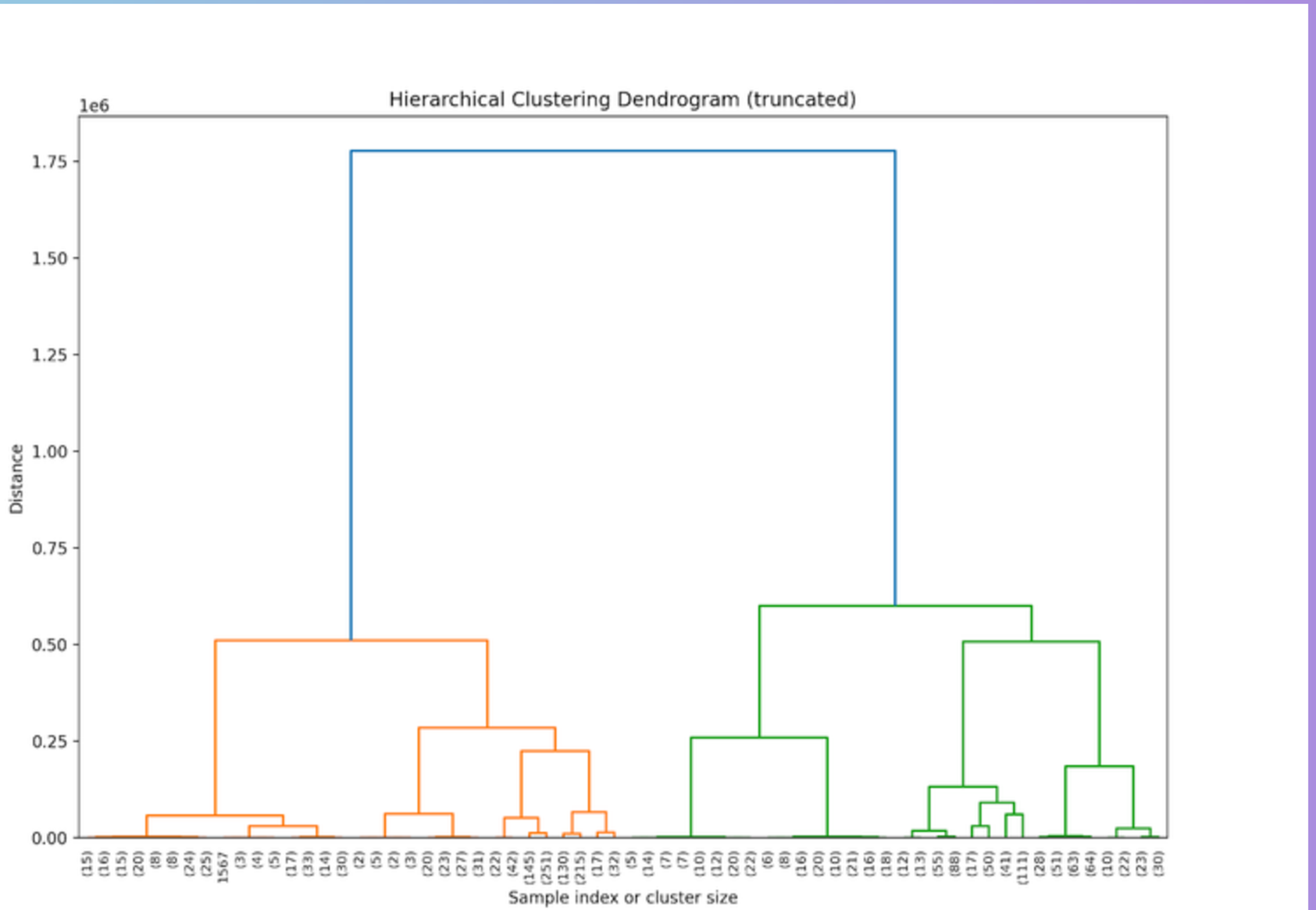
- **Computes street-length quartiles, intersection density (per km²) and maps them to:**
- **Radius range – median street length × [0.05 … 0.5].**
- **DBSCAN eps / min_samples – tighter in dense urban cores, looser in rural areas.**
- **Curvature & approach minima.**
- **Those values propagate through every adaptive pipeline call.**

We study the local roads to understand how long streets are and how close intersections are.
Then we adjust roundabout detection settings (like size limits and clustering) based on that.
This makes the algorithm smart enough to work well in both cities and rural areas.

Street Data Loader

load_and_analyze_street_data()

- **Looks for Streets.shp (or common fall-back filenames) and exits loudly if not found.**
- **Converts to EPSG : 3857 (Web Mercator) when the source CRS is geographic so that length, distance, buffer work in metres.**
- **Drops null / empty geometries and prints column summaries – invaluable when ingesting heterogeneous government data.**
- **Builds an R-tree (gdf.sindex) once to be reused everywhere else.**


Hierarchical Clustering Dendrogram (truncated)
1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Distance
(15) (16) (15) (20) (8) (8) (24) (25) 1567 (3) (4) (5) (17) (33) (14) (30) (2) (5) (2) (3) (20) (23) (27) (31) (22) (42) (145) (251) (130) (215) (17) (32) (5) (14) (7) (7) (10) (12) (20) (22) (6) (8) (16) (20) (10) (21) (16) (18) (12) (13) (55) (88) (17) (50) (41) (111) (28) (51) (63) (64) (10) (22) (23) (30)
Sample index or cluster size

Building the Street-Network Graph

- **R-tree index: first drop every street-segment's bounding box into an R-tree so neighbour look-ups are O(log N) instead of a brute-force scan.**

- **Smart sampling: for huge cities we cap processing at SPATIAL_SAMPLE segments (e.g. 20 k). This keeps the nested-loop stage from exploding while still giving dense coverage across the map.**

- **Neighbour query: for each sampled segment we ask the R-tree for any boxes that overlap its own box. That gives us a short candidate list of potentially intersecting segments.**

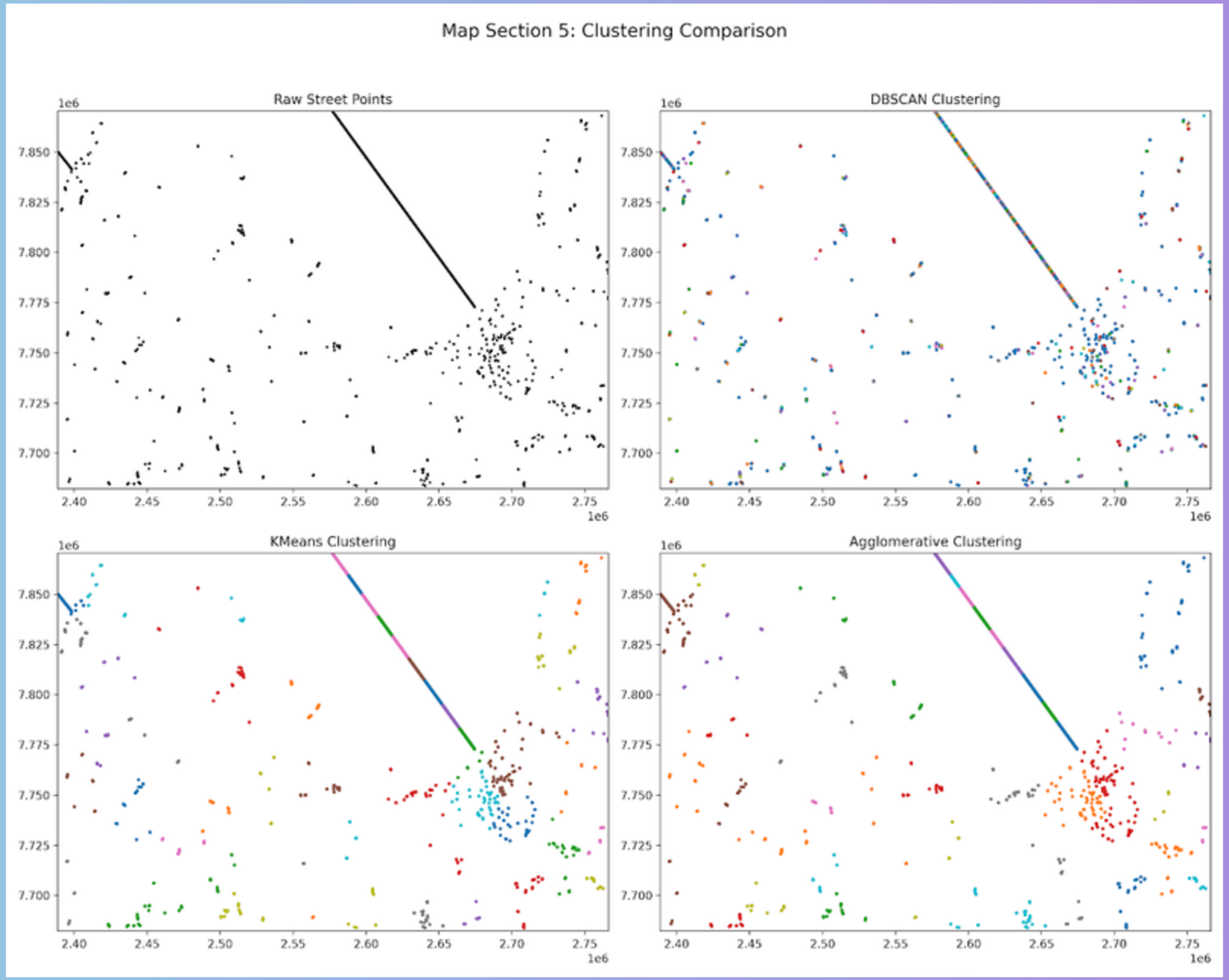
- **Precise intersection test: run Shapely's exact line-line intersection on those candidates to confirm true crossings and grab the intersection point geometry.**

- **Artifacts we keep:**
- **intersections list of (Point, $segID_1$, $segID_2$) pairs (capped at 100 k)**
- **intersection_points just the Point geometries (handy for clustering)**
- **connections dict mapping each segID to the set of neighbour segIDs**
- **high_connectivity_streets segments that touch three or more others—these flag complex junctions for later logic.**

- **Runtime profile: the precise intersection step dominates; sampling plus periodic status prints ("12 500 / 20 000 segments processed…") keeps the build both transparent and tractable.**

Three Strict Detection Pipelines

- **Same detectors, data-tuned:** Each “adaptive” version clones the strict logic but swaps in DBSCAN eps, min_samples, radius & curvature windows drawn from the dataset’s own street-length stats and intersection density.
- **Gentler gates for recall:** Validation score need ≥ 0.40 (vs 0.70) and confidence ≥ 0.45, letting weaker but real patterns survive.
- **Dynamic geometry bounds:** Radius range = 5 %–50 % of median street length; curvature threshold drops when roads are shorter.
- **Outcome:** Recovers legitimate mini-roundabouts in rural or low-connectivity suburbs that strict mode would flag as “too thin.”


Map Section 5: Clustering Comparison
Raw Street Points
DBSCAN Clustering
KMeans Clustering
Agglomerative Clustering
1e6
7.850
7.825
7.800
7.775
7.750
7.725
7.700
2.40
2.45
2.50
2.55
2.60
2.65
2.70
2.75

Adaptive-Mirror Feature	Novel Mechanism (How it works)	Why It Beats Traditional Approaches
Data-driven hyper-parameters	analyze_data_characteristics() derives DBSCAN eps, min_samples, radius & curvature windows from live street-length stats and intersection density.	No hand-tuning city-by-city; one model deploys worldwide and self-calibrates in seconds.
Recall-first safety net	After strict mode rejects a candidate, adaptive validators accept with score ≥ 0.40 & confidence ≥ 0.45 (vs 0.70/0.75).	Recovers rural or low-connectivity mini-roundabouts that fixed-threshold systems miss—without hurting overall precision.
Context-scaled geometry bounds	Radius window = 5 %–50 % × median street length; curvature threshold lowers when roads are shorter.	Handles 15 m village circles and 60 m urban rotaries with the same formula—avoids brittle “one-size-fits-all” rules.
Explainable hierarchical confidence	Confidence = 0.4 × geom_score + 0.3 × street_factor + 0.3 × intersection_factor; every candidate stores validation details.	Outputs a numeric quality metric that downstream apps can threshold differently for navigation, cartography, or QA—traditional tools give only pass/fail.

Thank You