# **Pipeline Methodology and Verification**

## **1. Data Cleaning**

Our dataset contains mobile pings from over 9,000 devices, tracked across four quarters. Each ping includes a measure of horizontal accuracy, representing the error in meters — with 0 being perfectly accurate.

To ensure data reliability, we remove all pings with a horizontal accuracy greater than **50 meters**, as such imprecise locations could significantly bias the analysis. This step eliminates **6.70%** of the total pings.

While even pings with 30–40 meters of inaccuracy can introduce some noise, we mitigate this issue through clustering, which averages locations and reduces the impact of such outliers. The trade-off between precision and data retention is important to consider. For context:

* Removing pings above 30 meters removes **10.88%** of the data.
* A stricter threshold of 10 meters would remove **49.31%** of the data.

A graph with numbers and lines

AI-generated content may be incorrect.

Figure 1 : Distribution of horizontal accuracy

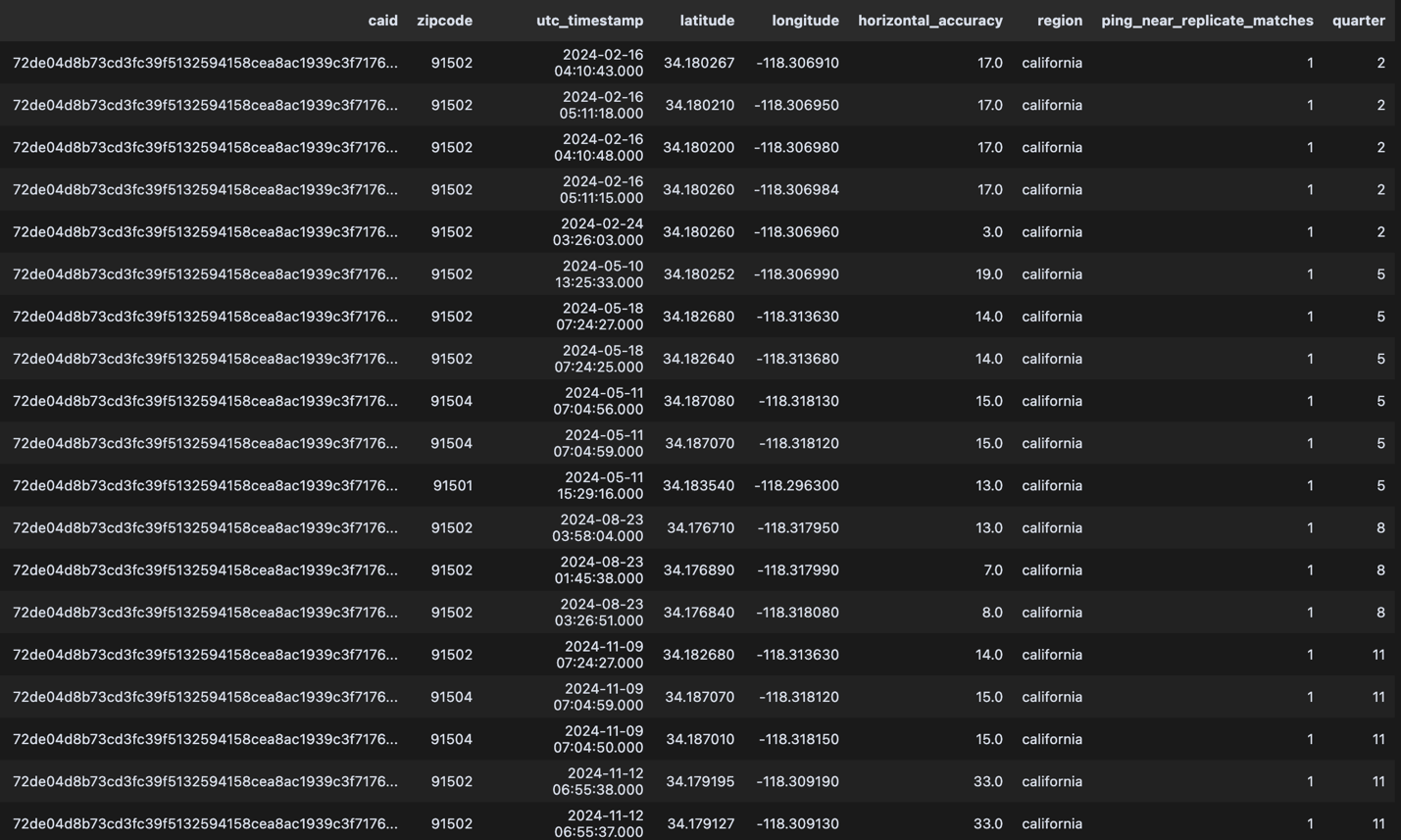
## **2. Clustering**

### **2.1 Selecting a Device**

To validate our pipeline, we selected a CAID (device identifier) with relatively few pings per quarter for manual inspection:

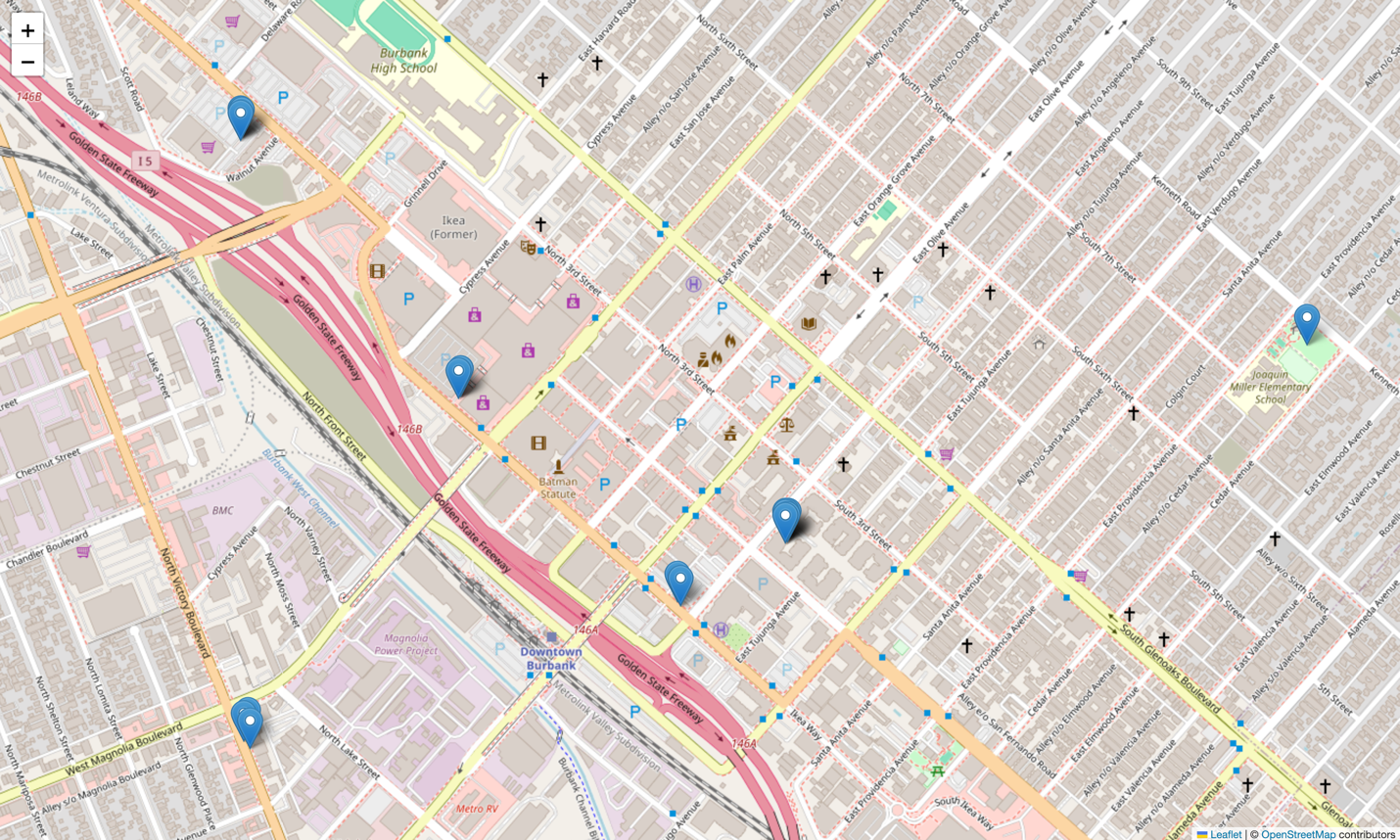
72de04d8b73cd3fc39f5132594158cea8ac1939c3f717633f4b6d939adf09235

This device has **19 pings in total** across all quarters.



### **2.2 Visual Inspection of Pings**

We visualize all pings on a map. While only 19 points are recorded, some overlap visually. Manual inspection reveals that the pings are located at **six distinct addresses**, which should ideally correspond to six clusters.



### **2.3 Clustering with DBSCAN**

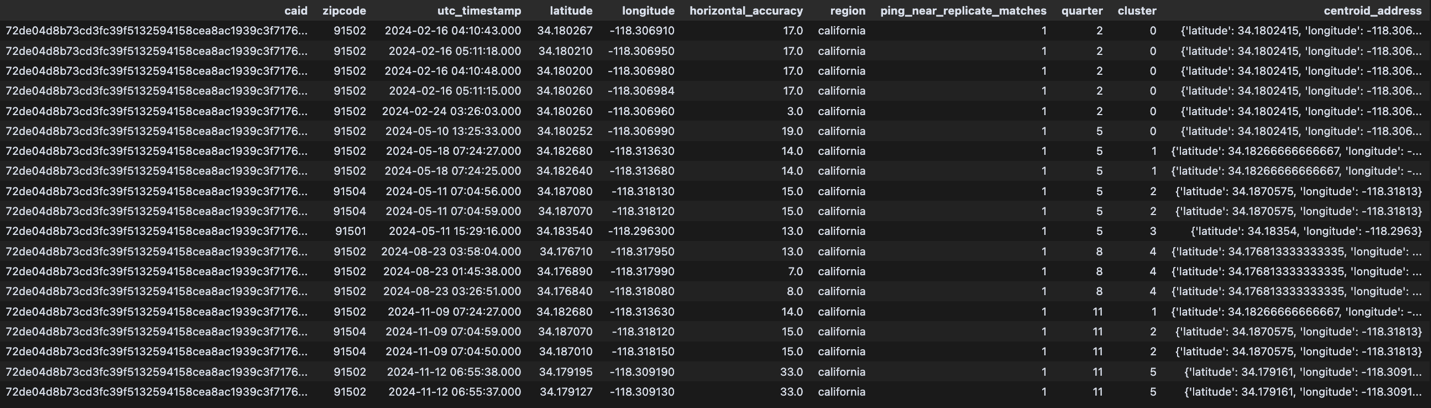
We use the DBSCAN algorithm, which relies on two parameters:

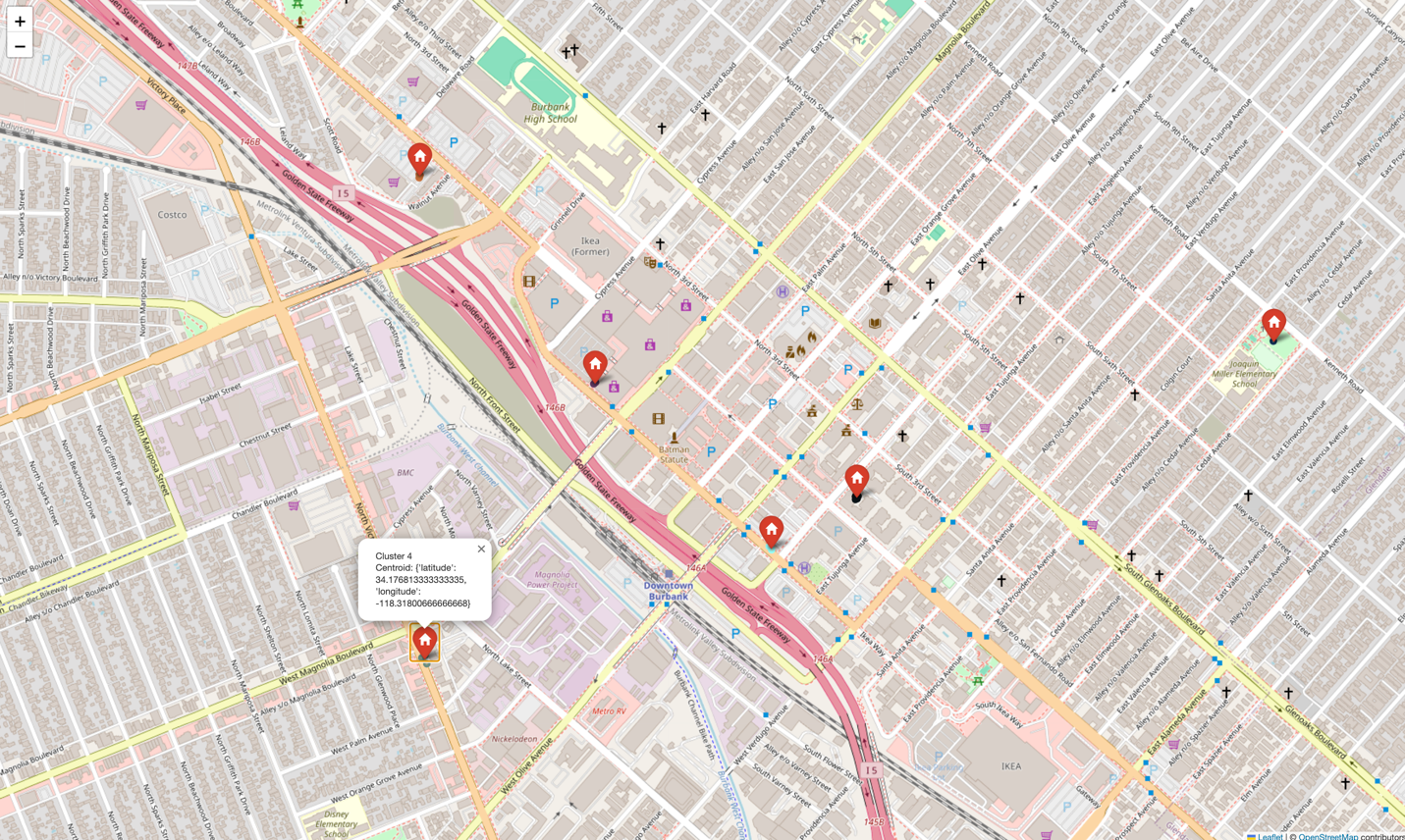
* **eps**: the radius around each point (set to 50 meters),
* **min\_samples**: the minimum number of neighboring points required to form a cluster (set to 1).

**How DBSCAN works (simplified explanation):**

1. Pick a random point.
2. Count neighbors within radius eps.
   * If neighbors ≥ min\_samples, mark it as a core point and start a cluster.
3. Expand the cluster by connecting other nearby core points.
4. Repeat until all points are visited.
5. Any point not reachable from a cluster is labeled as noise (not applicable here since min\_samples = 1).

**Result:**  
The algorithm successfully identifies **6 clusters (0 to 5),** matching the visual assessment.





## **3. Calculating Indicators**

We manually verify the indicators by generating two datasets:

* initial\_data\_with\_day\_periods.csv: raw pings with time labels.
* data\_with\_indicators.csv: computed indicators for each device per quarter.

**Notes:**

* Although we compute a Weekend Focus Score in the dataframe, it is excluded from the stability calculation and in subsequent LLM predictions.
* The “weights” that we describe only happen explicitly if we compute a traditional ML model. They don’t get an explicit value when we use an LLM for all predictions.

## Stability Score Formula

The **Stability Score** is calculated using a weighted combination of key features:

**Stability Score =**  
W₁ × Day Consistency Score +  
W₂ × Evening Consistency Score +  
W₃ × Night Consistency Score +  
W₄ × Unique Hours +  
W₅ × Max Consecutive Hours +  
W₆ × log(Total Pings) +  
W₇ × Consistency Score +  
W₈ × Dominance Score +  
W₉ × Time Window Coverage +  
W₁₀ × Entropy

## Metric Definitions

* **W₁–W₃**: Weights for consistency during **day**, **evening**, and **night** periods (based on presence during those time windows across the quarter).
  + This answers the question : “What percentage of days/evenings/nights did the cluster appear in a specific quarter?”. For example, if a mobile pinger at an address every night, the night consistency score is 100%.
  + **Note that day is between 8 am and 8pm (excluded), evening is between 8pm and 12pm (excluded), night is between 12pm and 4am. Also note that if no nights happen during a quarter, then the night consistency score is not 0, it is nan (same logic applies for the 3 different periods of the day)**
* **W₄**: Weight for **Unique Hours** — favors clusters with broader hourly presence.
* **W₅**: Weight for **Max Consecutive Hours** — captures uninterrupted stays (also accounting for silent hours).
* **W₆**: Weight for **Log Total Pings** — log-scaled to reward frequent clusters while reducing the impact of outliers.
* **W₇**: Weight for **Consistency Score** — measures how often a cluster appears across the quarter relative to total days with any activity (e.g. The mobile has pinged at this address every day 🡪 Consistency score = 100%)
* **W₈**: Weight for **Dominance Score** — the proportion of all device pings in the quarter that occurred in this cluster.
* **W₉**: Weight for **Time Window Coverage**, computed as:

  Time Window Coverage =  
  (Indicator\_day + Indicator\_evening + Indicator\_night) / 3

  Where Indicator\_window = 1 if the cluster was active in that time window.

* **W₁₀**: Weight for **Entropy Score**, which captures how evenly the pings are distributed across the hours of the day.  
    Lower entropy = focused/stable (e.g., always at night);  
    Higher entropy = scattered presence.

## **4. Assigning Clusters to Eviction Addresses**

### **4.1 Geocoding Eviction Addresses**

We retrieve eviction addresses and use the **Google Geocoding API** to obtain their latitude and longitude. The result is saved in the file:  
mobile\_data\_addresses\_feb\_geocoded\_sample.csv.

### **4.2 Matching Clusters to Addresses**

Each cluster is compared to all geocoded eviction addresses. If the closest address is within **50 meters**, the cluster is assigned to that address.

## **5. Identifying the Stable Address Using an LLM**

For each [mobile device, quarter] pair:

* We input the top 5 clusters (by ping count) and their indicators into a Large Language Model (LLM).
* The LLM selects the cluster most likely to represent the **main address** during that quarter.

The predictions are stored in:  
filtered\_data\_with\_main\_address\_per\_user\_gemini\_example.csv.