Dynamic Pricing for Urban Parking Lots - Capstone Project — Summer Analytics Hackathon 2025

Project Overview:

This project aims to optimize pricing for urban parking lots using real-time data.
Static pricing models lead to issues like:

- Overcrowding in high-demand zones
- Underutilization in others

To address these problems, this system dynamically adjusts prices based on:

- Occupancy rates
- Queue lengths
- Nearby traffic levels
- Special event indicators
- Type of incoming vehicle
- Competitor lot prices

Tech Stack Used:

- **Python 3.12** – Core language for data processing and modeling
- **Pandas & NumPy** – Data manipulation and numerical operations
- **Pathway** – Real-time data simulation and stream processing
- **Geopy** – Distance calculation for competitive pricing logic
- **Bokeh** – Real-time interactive visualizations
- **Google Colab / Jupyter** – Development and execution environment

Models Used:

1. Model 1: Baseline Linear Pricing

- Formula: $\text{Price}[t+1] = \text{Price}[t] + \alpha \times (\text{Occupancy} / \text{Capacity})$
- A simple linear update model based solely on how full the lot is

2. Model 2: Demand-Based Pricing

- Factors considered:
    - Occupancy ratio
    - Queue length
    - Traffic level
    - Special day flag
    - Vehicle type (e.g. truck vs. bike)
- Formula:
  $\text{Demand} = \alpha \times (\text{Occupancy} / \text{Capacity}) + \beta \times \text{Queue} - \gamma \times \text{Traffic} + \delta \times \text{IsSpecial} + \varepsilon \times \text{VehicleWeight}$
  $\text{Price} = \text{Base} \times (1 + \lambda \times \text{NormalizedDemand})$
- Price bounded between **0.5x** and **2x** the base price for stability

3. Model 3: Competitive Pricing (Optional)

- Adds geo-awareness using latitude/longitude
- Detects cheaper or more expensive lots within a 300m radius
- Price may decrease if nearby lots are cheaper or reroute is needed
- Price may increase if surrounding lots are full or expensive

Real-Time Pricing Workflow:

1. **Timestamp Creation**: Merges date and time columns to form a valid timestamp
2. **Streaming Simulation**: Uses `simulate_stream()` to mimic real-time entry
3. **Model Application**:

    - Applies Model 2 to calculate base demand-adjusted price
    - If competitors are nearby, Model 3 modifies the price further
4. **Logging & Visualization**: Stores all prices with timestamps for each lot
5. **Visualization**: Real-time plots created using Bokeh

Visual Output Summary:

- Real-time line graphs per lot showing how prices evolve
- Changes in price are **smooth** and **explainable**
- Prices respond in real-time to traffic, vehicle type, and demand

References:

- Summer Analytics 2025
- Pathway Documentation
- Bokeh Visualization Docs
- Geopy Library
- Mermaid Diagrams