Dynamic Pricing for Urban Parking Lots

Summer Analytics 2025 – Capstone Project

By: Titus Yankson (ekowyankson18@gmail.com)

Hosted by: Consulting & Analytics Club × Pathway

INTRODUCTION

Urban parking is a limited, high-demand resource. Cities often use static pricing models that fail to adapt to real-time changes in demand, leading to either overcrowded lots or underutilized spaces.

This project seeks to optimize parking utilization by dynamically pricing slots based on real-time demand, congestion, vehicle type, and nearby competition using a streaming simulation via Pathway.

PROBLEM STATEMENT

Fixed pricing for urban parking leads to:

- Overcrowding during peak hours
- Low usage during off-peak times
- Lack of economic incentives to guide driver behavior

This project addresses these inefficiencies by building an intelligent, real-time dynamic pricing engine for 14 parking lots.

DATASET DESCRIPTION

- Days: 73
- Parking Lots: 14
- Sampling: 18 time points per day (every 30 mins from 08:00–16:30)
- Features:

CATEGORY	FEATURES
Location	Latitude, Longitude
Parking Metrics	Occupancy, Capacity, Queue Length
Vehicle Info	Vehicle Type (car, bike, truck)
Environment	Traffic Level, Special Day
Time	Timestamp (Date + Time)

METHODOLOGY

Three models were built incrementally:

- Model 1 – Baseline Linear Pricing

  Price increases linearly with occupancy ratio:

$$\text{Price}_{t+1} = \text{Price}_t + \alpha \cdot \left(\frac{\text{Occupancy}}{\text{Capacity}} \right)$$

- Model 2 – Demand-Based Pricing

  Demand is computed as a weighted sum of real-time signals:

$$\text{Demand} = \alpha \cdot \left(\frac{\text{Occupancy}}{\text{Capacity}} \right) + \beta \cdot \text{QueueLength} + \gamma \cdot \text{TrafficLevel} + \delta \cdot \text{SpecialDay} + \epsilon \cdot \text{VehicleTypeWeight}$$

Then:

$$\text{Price}_t = \text{BasePrice} \cdot (1 + \lambda \cdot \text{NormalizedDemand})$$

- Model 3 – Competitive Pricing with Rerouting

  Incorporates spatial competition using Haversine distance.

  If a nearby lot (within 1 km) is cheaper and available:

  - Suggest rerouting (reroute = 1)
  - Or reduce price to stay competitive

DEMAND FUNCTION AND WEIGHT JUSTIFICATION

Feature	**Weight**	**Rationale**
Occupancy	0.4	Core signal for capacity stress
Queue Length	0.2	Proxy for demand pressure
Traffic Level	0.2	External congestion
Special Day	0.1	Event-driven surges
Vehicle Type	0.1	Prioritizing higher-demand vehicles

Demand is normalized to keep prices bounded:

Min: 0.5 × BasePrice, **Max**: 2 × BasePrice

REROUTING LOGIC

Used Haversine formula to compute distance between lots:

Python code:

```
“def    haversine(lat1, lon1, lat2, lon2):
            ...
            return distance_km
“
```

If current lot is ≥ 95% full and nearby lots are cheaper → suggest rerouting.

Pricing adjustments are made based on competitors' prices.

Implementation Details

Tool	**Purpose**
Python	Core Logic
Pandas	Data Cleaning and Processing
Numpy	Numeric Operations
Pathway	Real-time Simulation Engine
Bokeh	Interactive Visualization
Colab	Execution Environment

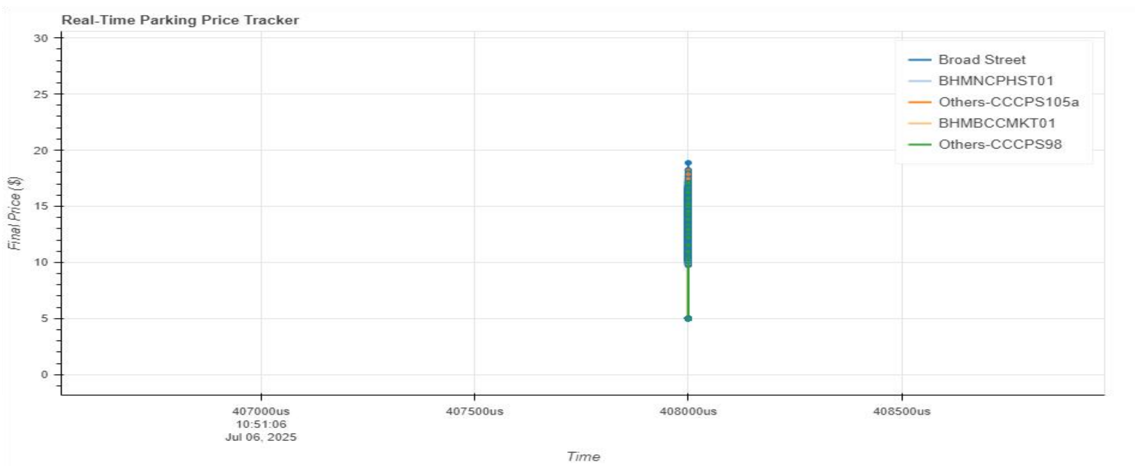
All logic was implemented from scratch without ML libraries like scikit-learn.

VISUALIZATION (BOKEH)

An interactive **Bokeh** dashboard shows:

- final_price vs time for each lot_id
- Hover tool reveals reroute status, timestamp, price
- Line plot with togglable legend

See sample visual below:



Results Summary

Lot	Occupancy	Capacity	Final Price	Reroute
BHMBCCMKT01	146	577	$ 14.64	0
BHMNCPHST01	213	1200	$ 11.06	0
BHMEURBRD01	467	470	$ 17.66	0

Insights:

- Prices adjust gradually, not erratically
- Lots with rising demand hit upper price bounds
- Rerouting activates only when full & nearby alternatives exist

CONCLUSION

This project demonstrates the effectiveness of a real-time pricing system using minimal libraries and intelligent logic.

Strengths:

- Realistic modeling of parking dynamics.
- Real-time simulation using Pathway.
- Visualization aligns pricing with behaviour.

Future Work:

- Add revenue maximization objective.
- Predictive modeling using time-series.
- Expand rerouting with driver navigation integration.

Appendix

- Pathway docs: pathway.com/developers.
- Dataset: Provided CSV (14 lots × 73 days × 18 intervals).
- Code: Google Colab Notebook submission.