Capstone Report: Dynamic Pricing for Urban Parking Lots

Project Overview

This capstone project, conducted as part of Summer Analytics 2025, addresses the issue of inefficient and static pricing in urban parking lots. The solution involves developing a dynamic pricing engine that uses real-time data to intelligently adjust parking fees based on current demand, vehicle types, and surrounding conditions. The project incorporates real-time data processing using Pathway, pricing model logic using Python (NumPy, Pandas), and real-time visualization via Bokeh.

Dataset Description

The dataset includes parking data from 14 urban parking lots over a period of 73 days. Each day includes 18 time-slices from 8:00 AM to 4:30 PM, captured at 30-minute intervals. Key features:

- Location (Latitude, Longitude)
- Capacity and Occupancy of each lot
- Queue Length
- Vehicle Type (Car, Bike, Truck)
- Nearby Traffic Congestion
- Special Day Indicator (holiday, event)
- Timestamp

Project Objective

To implement a dynamic pricing system that:

- Starts from a base price of $10
- Adjusts pricing based on real-time conditions
- Responds to factors like queue length, traffic congestion, and event days
- Remains smooth and explainable
- Optionally suggests rerouting when nearby competitor lots are more favorable

---

Model 1: Linear Pricing Model

This baseline model uses a linear relation between occupancy and price:

Price = BasePrice + α * (Occupancy / Capacity)

Where $\alpha = 5$, and price is bounded between $5 and $20. This model captures basic supply-demand logic and provides a reference point for advanced models.

Model 2: Demand-Based Pricing Model

This advanced model improves on Model 1 by incorporating multiple real-world features to calculate demand more realistically. These features include:

- Occupancy rate (Occupancy / Capacity)
- Queue length
- Traffic congestion level
- Special day indicator (e.g., holidays, events)
- Type of incoming vehicle (car, bike, truck)

Demand Function:

Demand = α * (Occupancy / Capacity) + β * QueueLength - γ * Traffic + δ * IsSpecialDay + ε_vehicle

- $\alpha = 5$ (weight for occupancy rate)
- $\beta = 0.1$ (weight for queue length)
- $\gamma = 0.05$ (penalty for traffic)
- $\delta = 0.2$ (boost for special day)
- ε_vehicle = vehicle type weight (car = 1.0, bike = 0.5, truck = 1.5)

The raw demand is normalized to ensure output values fall within a consistent range.

Price Calculation:

Price = BasePrice * (1 + λ * NormalizedDemand)

- $\lambda = 0.3$ controls the sensitivity of price to demand
- Price is clipped between 0.5× and 2× the base price to prevent erratic jumps

This model ensures smooth, realistic price transitions and responds to real-time demand changes effectively.

Real-Time Simulation using Pathway

- Implemented real-time streaming using Pathway's pw.io.csv.read() in streaming mode
- Used pw.temporal.tumbling() windows to smooth daily pricing updates
- Applied UDFs for both Model 1 and Model 2 price generation
- pw.run() executed live updates continuously

Visualization with Bokeh

- Used Panel + Bokeh for interactive real-time line plots
- Plotted Model 2 price per lot across timestamps
- Used dropdowns for lot selection
- Optional competitor pricing overlay based on geographic proximity

---

Outputs

- Final streaming output saved to parking_stream.csv
- Includes timestamps, LotID, Occupancy, Model 1 price, and Model 2 price

Sample:

Timestamp	LotID	Occupancy	Capacity	Model2_Price
2025-07-01 08:00AM	12.9716_77.5946	45	60	$12.84

Justification of Pricing Behavior

- Occupancy increases lead to higher prices
- Special days cause noticeable spikes
- Queue length and vehicle type further influence pricing levels
- Smooth changes achieved using windowing functions

Conclusion

This project demonstrates the ability to:

- Design rule-based dynamic pricing systems from scratch
- Simulate real-time ingestion and response using Pathway
- Visualize and explain price shifts effectively

Future scope includes:

- Enhancing the competitor pricing logic
- Integrating predictive ML models for price forecasting
- Rerouting logic for overflow conditions

Prepared by: Dhatri Tatavarthi
Summer Analytics 2025