Dynamic Pricing for Urban Parking Lots

Capstone Project – Summer Analytics 2025

By: Anurag Pandey
Date: July 2025

---

1. Problem Statement

Urban parking resources are limited, and inefficient pricing often leads to either underutilized lots or excessive congestion. This project aims to implement a **real-time dynamic pricing engine** for 14 parking lots using live data streams and open-source tools.

Using pandas, numpy, and Pathway, we built three pricing models that increase in complexity — starting with a baseline and evolving into competition-aware pricing. This system is designed to scale to real-time urban infrastructure and help optimize occupancy, revenue, and driver satisfaction.

---

2. Dataset Overview

- **Observations**: 18,368
- **Features**: 12
- **No missing data or duplicates**

Key Variables:

Feature	Description
ID	Unique parking lot identifier
Occupancy, Capacity	Used to determine fullness ratio
QueueLength	Number of vehicles waiting
TrafficConditionNearby	Proxy for surrounding congestion
VehicleType	Categorical: car, bike, truck
IsSpecialDay	Boolean indicating holidays/events
Latitude, Longitude	Used to calculate spatial proximity
Timestamp	Merged from LastUpdatedDate + Time

3. Models Used

🧮 Model 1: Baseline Linear Pricing

Formula:

$$\text{Price}(t+1)=\text{Price}(t+\alpha)\ .\ (\text{CapacityOccupancy})$$

Base price: $10

- **α (sensitivity)**: 2.0
- **Behavior**: Price increases gradually with lot fullness.

This model is simple and reactive but does not consider external conditions like traffic or queue length.

📈 Model 2: Demand-Based Dynamic Pricing

We define a **custom demand function**:

Demand =
$$\alpha\cdot\text{CapacityOccupancy}+\beta\cdot\text{QueueLength}-\gamma\cdot\text{Traffic}+\delta\cdot\text{SpecialDay}+\epsilon\cdot\text{VehicleTypeWeight}$$

Weights:

- α = 2.0 (Occupancy ratio)
- β = 0.5 (Queue length)
- γ = 1.0 (Traffic level)
- δ = 2.0 (Special day boost)
- ε = 1.2 (Vehicle type)

Vehicle Weights:

- Car: 1.0
- Bike: 0.5
- Truck: 1.5

Price Mapping:

- Pricet = BasePrice·(1+λ·NormalizedDemand) λ = 1.0

- Prices are **bounded between $5 and $20**

Normalization:
Demand is normalized per lot across all timestamps to ensure fair comparison.

🧭 **Model 3: Competitive Pricing with Geo-Intelligence**

This model uses **spatial intelligence** to adjust price competitively.

- If a lot is **over 90% full** and **nearby lots (within 1km)** are cheaper and less full → **reduce price**
- If nearby lots are **full and expensive** → **increase price**

We used geopy.distance.geodesic() to compute proximity and adjusted Model2Price ±10%:

- If reducing: price = max(0.9 × model2_price, 5)
- If increasing: price = min(1.1 × model2_price, 20)

This model mimics real-world driver behavior and market-based responses.

4. Assumptions

- Prices must be bounded between $5 and $20.
- Time intervals between records are consistent (~30 mins).
- Lot locations are fixed; no changes in lat-long.
- Nearby lots are defined within **1 km radius**.
- Vehicle types affect demand: larger vehicles are weighted higher.

5. Real-Time Streaming with Pathway

We embedded pricing logic into a custom Transformer using Pathway, allowing:

- Live price recalculation per event
- Streaming ingestion from CSV-like connectors
- Real-time routing through the UDF

Example UDF Structure:

```
def compute_price_row(...):
  raw_demand = ...
  normalized = ...
  base_price = 10
```

```
    return clip(base_price * (1 + normalized), 5, 20)
```

We deployed this via:

```
pw.run()
```

The output table streams time-stamped prices for each lot.

6. Visualizations (Bokeh)

We created interactive Bokeh plots comparing all three models:

- **X-axis**: Timestamp
- **Y-axis**: Price
- **Legend**: Model 1, Model 2, Model 3
- **Tools**: Hover to inspect exact price/time
- **Lots**: Plots shown for 3 sample lots

Sample Chart Insights:

- Model 1 always increases, even during low queue/traffic.
- Model 2 adapts well to special days and peak traffic.
- Model 3 smooths pricing under pressure and avoids overpricing when competition exists.

7. Results & Observations

- **Model 2** is a good balance of flexibility and interpretability.
- **Model 3** provides the best real-world adaptation but adds computational overhead.

8. Future Work

- **Reinforcement learning** for pricing strategy optimization
- **Driver rerouting suggestions** using real-time demand heatmaps
- **Integration with city-wide traffic APIs**
- **UI dashboard using Dash or Streamlit**
- **Multi-objective optimization** (e.g., revenue + fairness + availability)