Capstone Report

Dynamic Pricing for Urban Parking Lots

Submitted for: Summer Analytics 2025
By: Brick Bose

Problem Statement

Urban parking spaces are underutilized or overcrowded due to static pricing. The challenge is to implement real-time pricing using live sensor and traffic data to optimize parking usage across city lots.

Solution Overview

We developed a dynamic pricing engine using **Pathway** to process real-time data and compute parking rates based on occupancy, traffic, queue length, and special events.

Models Implemented

1. Model 0 – Daily Window (Baseline)

Function Chosen:

Price=10+ (Max Occupancy−Min Occupancy)/Capacity

Why this model was chosen:

- It provides a simple daily signal of **demand volatility**.
- It uses **temporal aggregation (daily window)** which smooths out hourly noise.
- The difference between peak and low occupancy reflects **scarcity and variation**.
- This model sets a **baseline** to compare more dynamic pricing strategies.

Limitations:

- Doesn't react to real-time changes.
- Doesn't factor in queue, traffic, or event-based pressure.

✅ It serves as a control/reference model.

2. Model 1 – Row-wise Occupancy + Queue Model

Function Chosen:

$$\text{Price} = \text{Base} + \alpha \cdot (\text{OccupancyCapacity}) + \beta \cdot \text{QueueLength}^{1.5}$$

With:

- `base_price = 10`
- `alpha = 0.6`, `beta = 0.4`
- Price bounded between ₹5 and ₹20

Why this model was chosen:

- Designed to **respond in real time** (row-by-row streaming).
- Occupancy ratio gives a **direct proxy for demand pressure**.
- Queue length reflects **immediate congestion or excess demand**.
- We tested **nonlinear scaling** of queue (using exponent 1.5) to reflect rising urgency as queues grow.

Why the function structure:

- Linear + power-based formula was **interpretable and scalable**.
- We tested and visualized multiple α–β combinations to get smooth output within bounds.
- This model reacts quickly to **spikes or dips** in lot usage.

✅ It's effective for lots where **queue buildup is a major factor**.

3. Model 2 – Demand-Based Sigmoid Function

Function Chosen:

$$x = w1 \cdot \text{OccupancyRatio} + w2 \cdot \text{Queue} + w3 \cdot \text{Traffic} + w4 \cdot (\text{IsSpecialDay} \cdot \text{Traffic}) + w5 \cdot \text{VehicleRisk}$$

$$\text{Demand} = 1 / (1 + e^{-k(x-\mu)}), \quad \text{Price} = 10 \cdot (1 + \text{Demand})$$

Why this model was chosen:

- This model was developed after testing Model 1 and realizing that:
  - More **factors needed to be considered**: e.g. `TrafficConditionNearby`, `IsSpecialDay`, `VehicleType`
  - Linear pricing functions started to **saturate** and couldn't capture **nonlinear interactions**

Sigmoid Function Rationale:

- Converts a weighted sum of features (`x`) into a **smooth bounded value** (0 to 1)
- Allows us to **control sensitivity** via hyperparameters:
  - `k = steepness`
  - `μ = threshold center`
- Scales final price to stay within a **natural range**, with strong influence from real-time demand signals

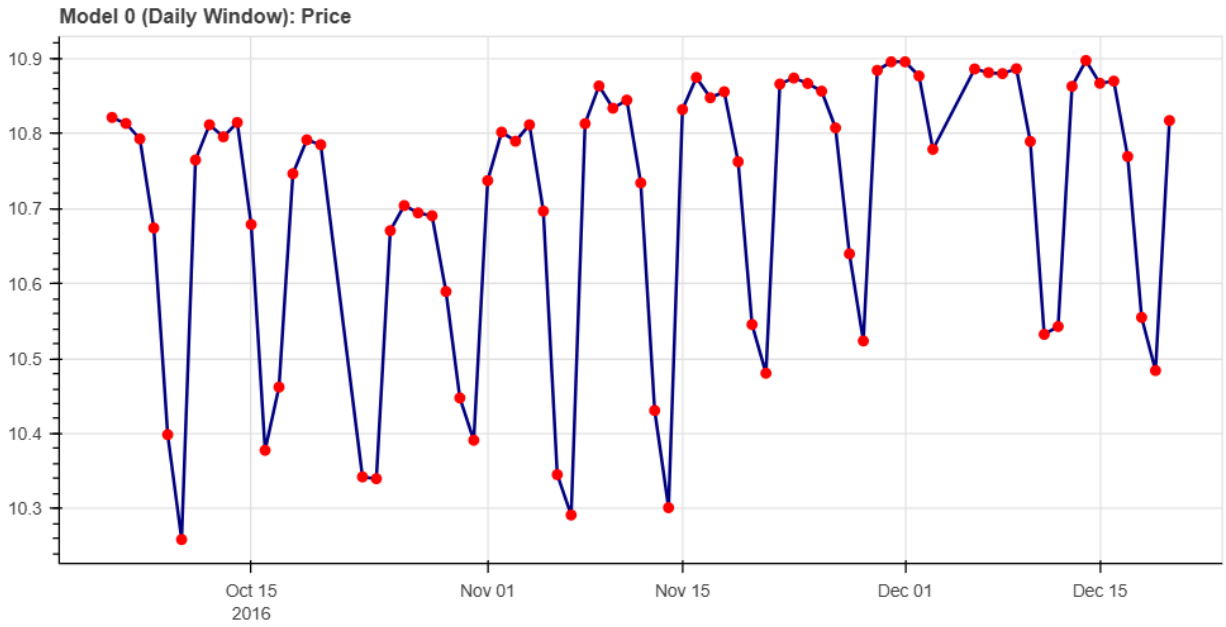
Why it's superior:

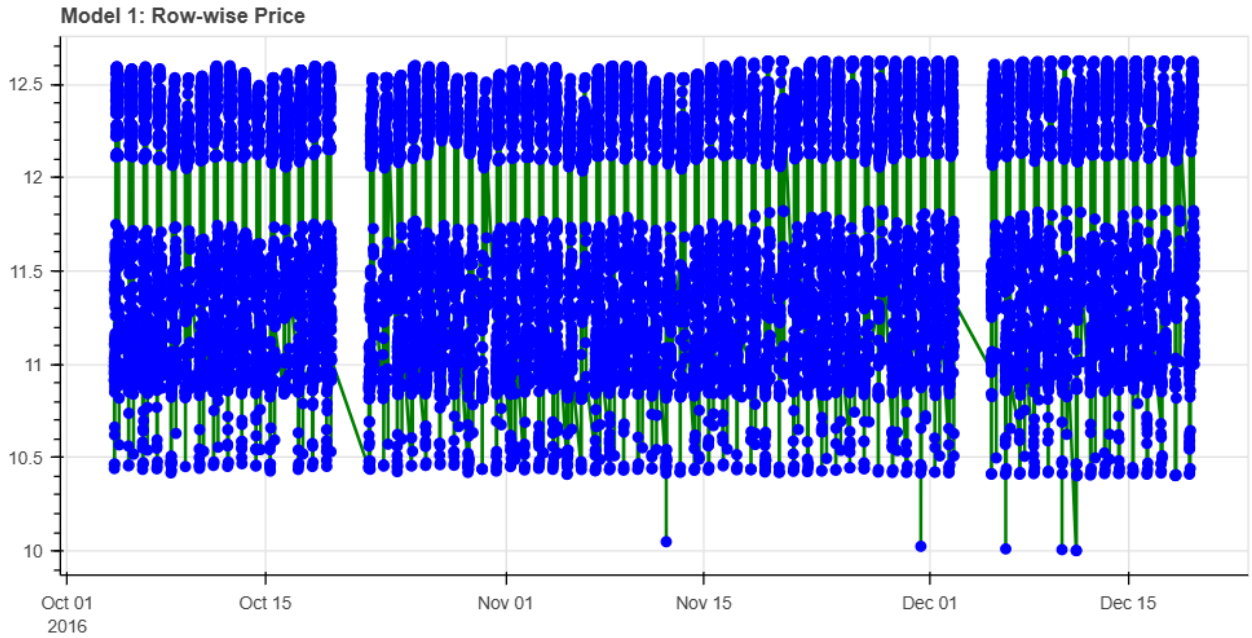
- It was the most **flexible and extensible** model
- Pricing rises sharply under multiple high-pressure features (traffic + queue + event)
- Works well even with noisy streaming data

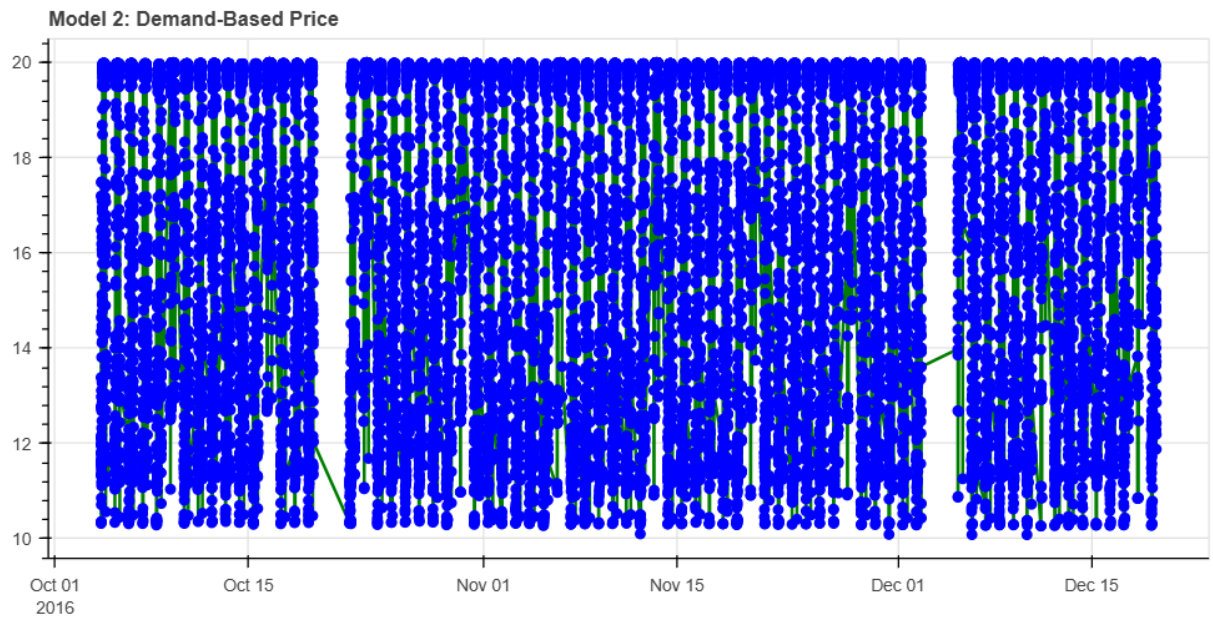
✅ Model 2 represents a real-world **adaptive pricing engine** and performed best in plot visualizations.

Visualization

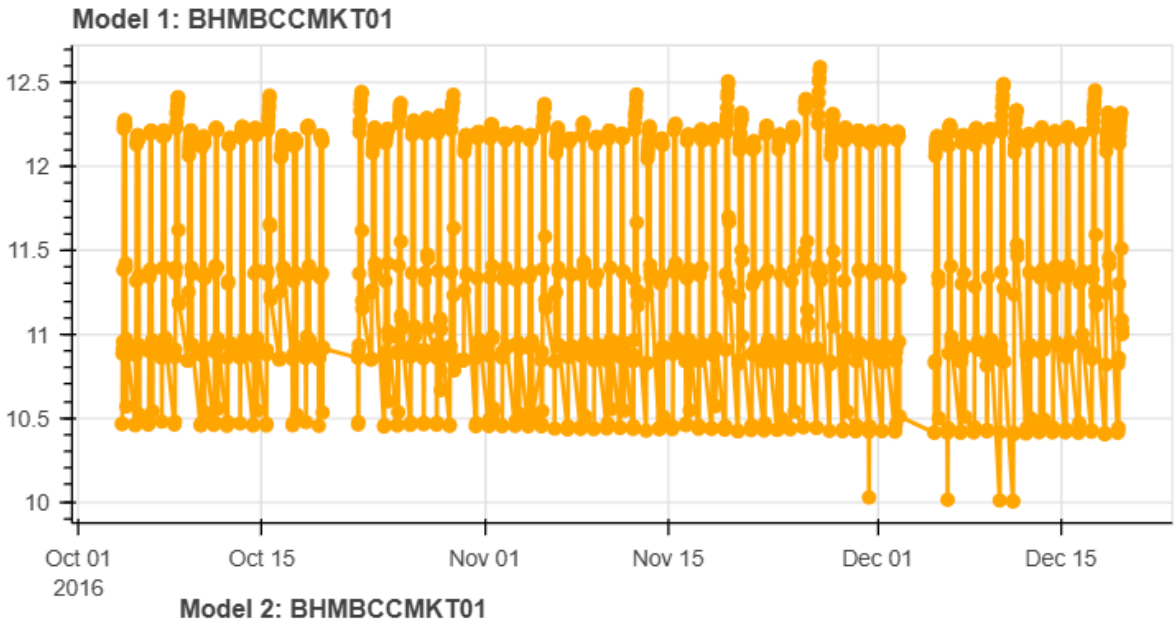
Bokeh plots were created to visualize pricing output in real time across lots and models.

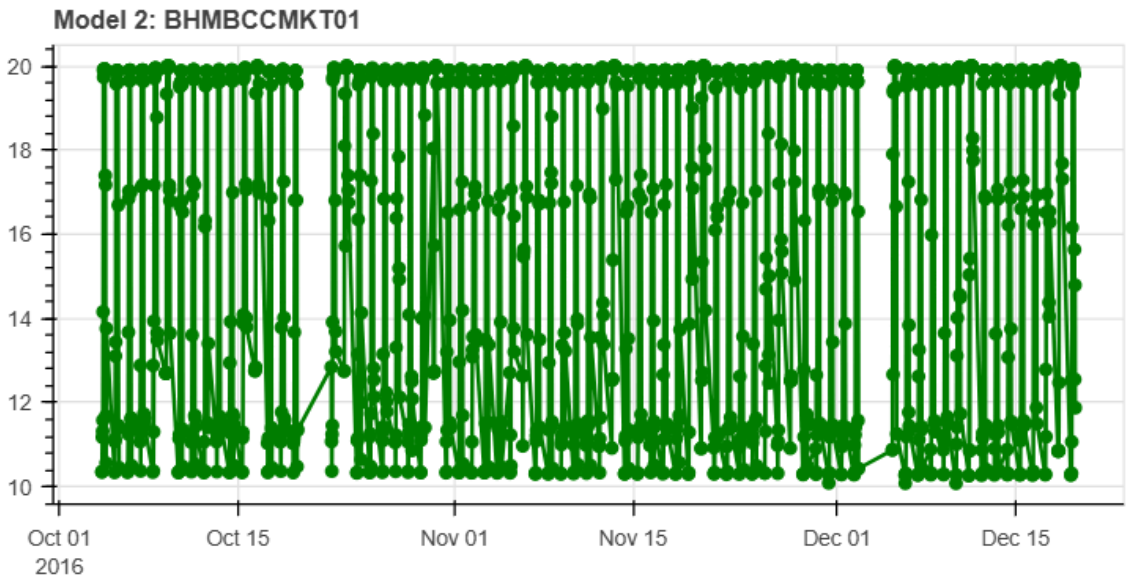






Each model was validated on the same lot for comparative analysis.





Tech Stack

- `Pathway` for real-time data processing
- `Bokeh + Panel` for visualization
- `Python (pandas, numpy)` for preprocessing and modeling
- `Google Colab` as the development environment

Outcome

The models provide scalable, real-time pricing outputs. Model 2 proved most robust, factoring in multiple features to reflect demand volatility.

This architecture can be extended to optimize revenue and traffic flow in real urban deployments.