Dynamic Pricing for Urban Parking Lots

Capstone Project – Summer Analytics 2025

Shaik Nowshin Farhana

Project Objective

To develop a real-time, intelligent, and data-driven pricing engine for 14 urban parking lots, adapting dynamically to fluctuating demand, competition, and environmental conditions. Our goal is to:

- Ensure efficient utilization of parking spaces
- Avoid overcrowding or underutilization
- Enable real-time streaming using Pathway
- Visualize pricing behavior using Bokeh

Tech Stack

Layer	Technology
Programming	Python
DataManipulation	Pandas,NumPy
Real-TimeEngine	Pathway
Visualization	Bokeh
Documentation	Markdown,LATEX

Models Implemented

Model 1: Baseline Linear Pricing Model

Formula:

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

Parameters:

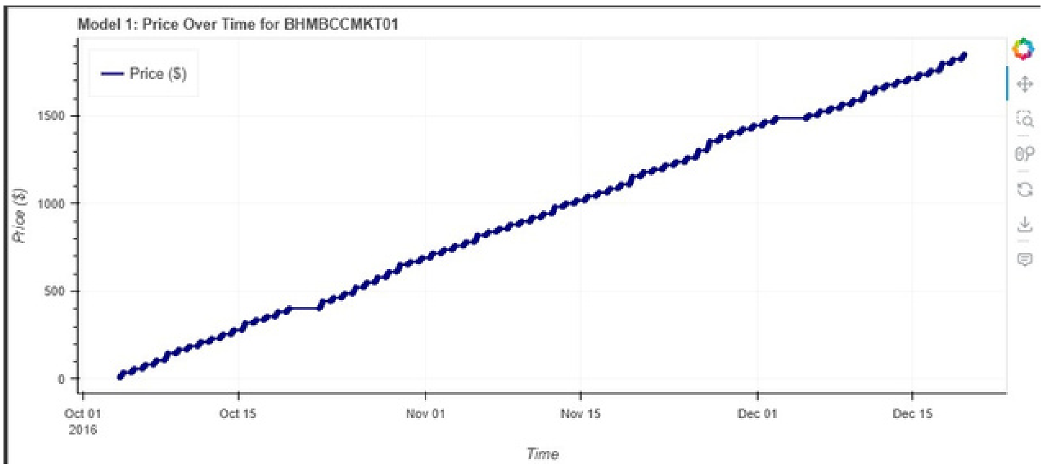
- Base price: $10
- $\alpha=5$

Characteristics:

- Simple, explainable price growth
- Based only on occupancy

Limitations:

- Ignores queue length, traffic congestion, special days, or vehicle type
- No competition modeling



Model 2: Demand-Based Pricing Model

Demand Function:

$$\text{Demand} = 0.5 \cdot \text{OccRate} + 0.3 \cdot \text{QueueLength} - 0.2 \cdot \text{TrafficLevelNum} + 0.5 \cdot \text{IsSpecialDay} + 0.7 \cdot \text{VehicleWeight}$$

Price Function:

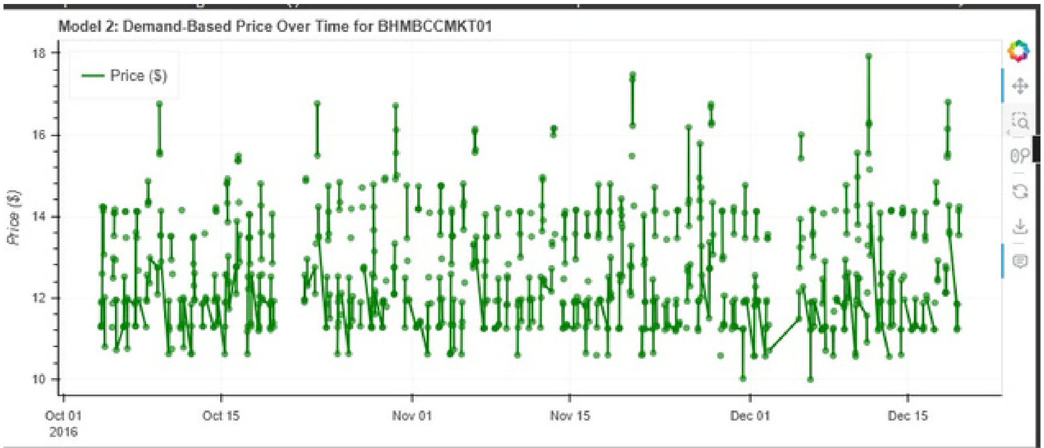
$$\text{Price}_t = 10 \cdot (1 + \lambda \cdot \text{NormalizedDemand}), \quad \lambda = 1.0$$

Feature Mapping:

- TrafficConditionNearby: {low=1, medium=2, high=3 }
- VehicleTypeWeight: {bike=0.5, car=1.0, truck=1.5 }
- Price clipped: $5 to $20

Benefits:

- Accounts for queue, congestion, vehicle type Enables
- dynamic pricing based on real-world features



Model 3: Competitive Pricing Model

Competition Logic:

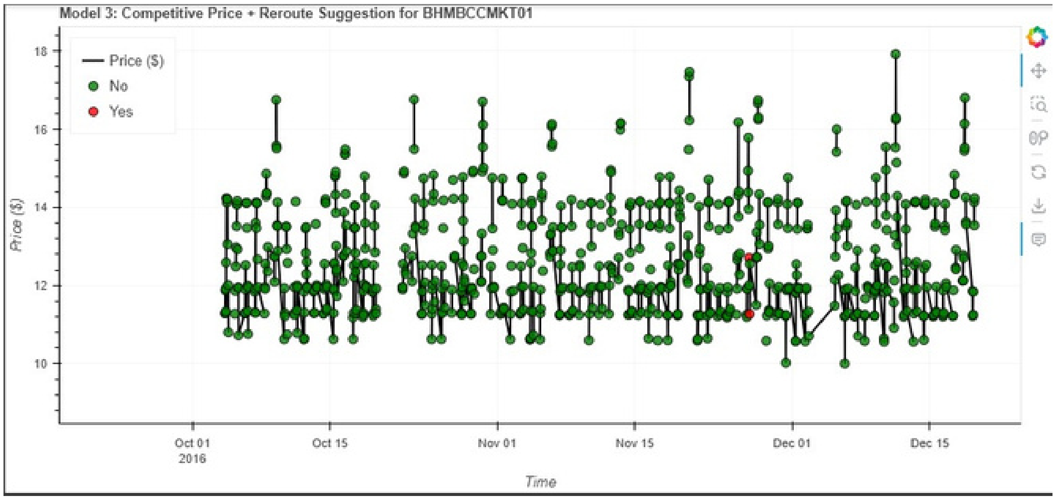
- If current lot is over 90% full and nearby lots are cheaper: reduce price or reroute
- If nearby lots are full and expensive: increase price to capture demand

Implementation Steps:

1. Calculate Haversine distance to all other lots
2. Select neighbors within 1 km
3. Calculate average neighbor price and occupancy
4. Adjust price using a custom function

Outputs:

- Price Model3 Real-time competitive price
- SuggestReroute Boolean rerouting flag



Demand Function Derivation (Expanded)

Demand = 0.5 ·OccRate+0.3·QueueLength+0.2·TrafficLevelNum+0.5·IsSpecialDay+0.7·VehicleWeight

- OccRate: Real-time utilization percentage
- QueueLength: Higher queues increase demand score
- TrafficLevelNum: Congestion penalizes price
- IsSpecialDay: Boosts demand
- VehicleWeight: Heavier vehicles imply more space usage

Real-Time Implementation in Pathway

Pipeline Steps

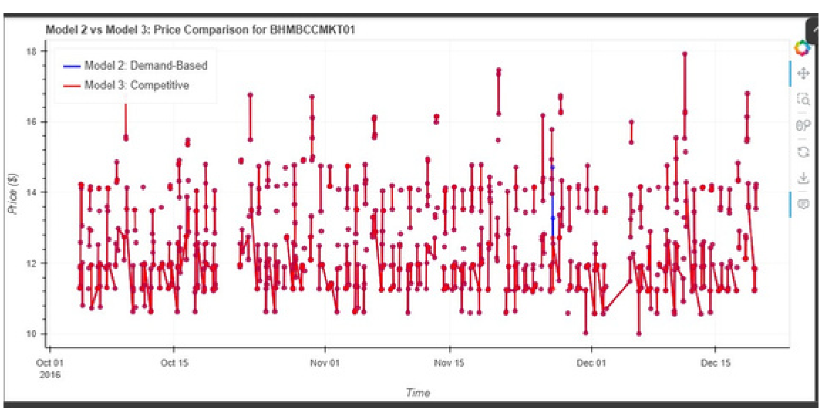
1. Read stream from parking stream.csv
2. Apply schema with capacity, occupancy, traffic, queue
3. Join stream with itself (proximity logic)
4. Apply UDFs for haversine and price adjustment

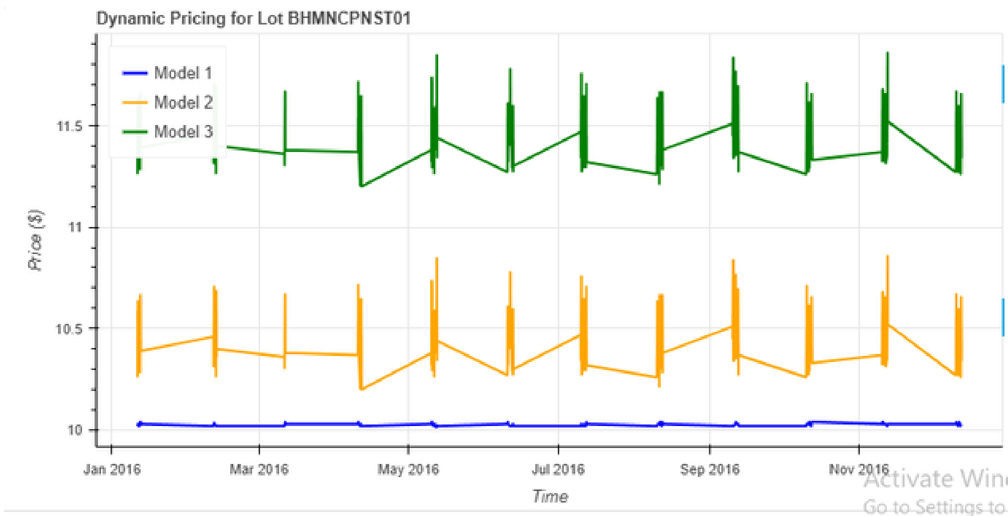
Schema Definition

```
class   ParkingSchema(pw.Schema):
        Timestamp: str
        SystemCodeNumber: str
        Capacity: int
        Latitude: float
        Longitude: float
        Occupancy: int
        QueueLength: int
        TrafficConditionNearby: str
        IsSpecialDay: int
        VehicleType: str
        Price_Model2: float
```

Bokeh Visualization Summary

- Model 1: Smooth linear curve
- Model 2: Interactive hover, color-coded dots
- Model 3: Comparison graph with reroute indicators





- Stream CSV data into Pathway engine Apply
- three models sequentially Reroute and price-
- adjust based on competition Visualize in real-
- time with Bokeh

Final Deliverables

- Python + Pathway-based real-time engine
- Bokeh-based visualization dashboard
- GitHub Repo Link for detailed explanation

Resources Used

1. Pathway: From Jupyter to Deploy
2. Pathway: First Real-Time App
3. Summer Analytics 2025 Portal