Dynamic Pricing for Urban Parking Lots

Consulting & Analytics Club × Pathway

Vikrant Singh

Project Overview

Urban parking lots often suffer from inefficiencies caused by static pricing models that fail to adapt to fluctuating demand throughout the day. This project aims to develop a real-time dynamic pricing engine for **14 urban parking** lots by utilizing historical occupancy data, real-time demand indicators, and competitive pricing insights. The solution integrates various features such as queue length, vehicle type, traffic congestion levels, and special event indicators to calculate demand and adjust prices accordingly. **A three-tiered modelling** approach is adopted, starting with a **baseline linear model**, progressing to a **demand-based pricing function**, and extending to a **competition-aware pricing model** that incorporates geographic proximity and competitor rates.

Dataset Description

Data includes 73 days of real-time records for 14 locations, captured 18 times per day.

Location (Latitude, Longitude)

These geographical coordinates help determine the **exact position** of each parking lot. They are crucial for calculating **proximity to other lots**, enabling competitive pricing logic and rerouting recommendations.

Parking Metrics

- **Capacity** defines the total vehicle holding limit of a parking lot.
- **Occupancy** is the current count of parked vehicles at any time.
- **Queue Length** shows how many vehicles are waiting to enter.

Vehicle Information

The type of incoming vehicle **car, bike, or truck** which can be used to **assign different price weights** or preferences based on space usage or vehicle size.

Environmental Factors

- **Special Event Day Indicator:** A binary or categorical flag signalling whether the day includes **events, holidays, or public gatherings**, which typically lead to **surge demand**.
- **Traffic Conditions:** Reflects the **level of congestion** near the parking area.

Timestamp (Date + Time)

A unified column combining both **date and time of observation**, essential for **time-series modelling** and aligning records chronologically in real-time simulations.

Tools and Technologies

Programming Language: Python

Libraries: Pandas & Numpy (for data manipulation and calculations)

Real-Time Processing: Pathway (for streaming and time-order preservation)

Visualization: Bokeh (for live pricing dashboards)

Model 1

Baseline Linear Model

Strategy

Parameters

- BASE_PRICE = 10.0
- α = 2.0

Formula Used

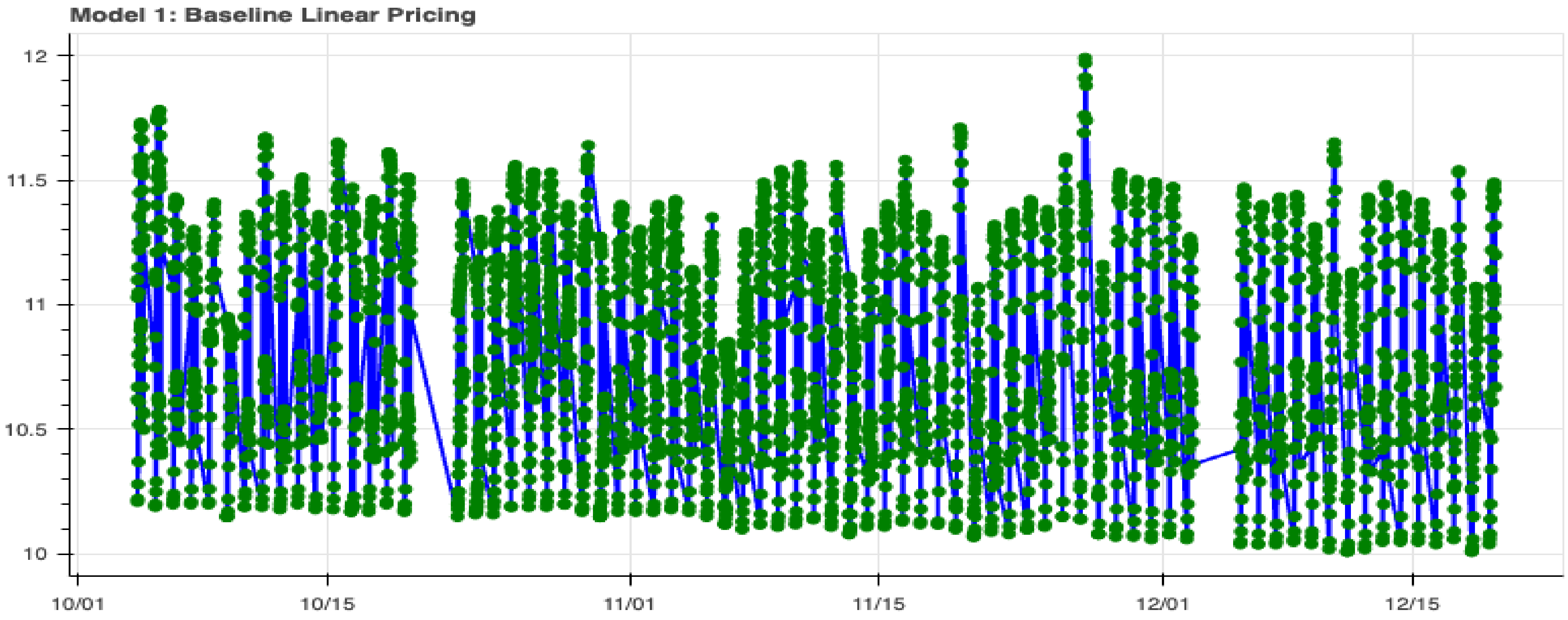
price = BASE_PRICE + α * (occupancy / capacity)

Assumptions

- Price increases with occupancy variation
- Simple, smooth variation

Key Takeaways

- Operational Transparency
- Efficient Processing
- Scalability and Monitoring
- Streaming Model Suitability



Visualization for Model

Model 2

Demand Based Pricing

Strategy

Parameters

- **Constants**
  BASE_PRICE = 10.0 | $\alpha = 0.5$ | $\beta = 0.2$ |
  $\gamma = 0.1$ | $\delta = 1.0$ | $\varepsilon = 0.8$ | $\lambda = 0.6$
- **Vehicle Weights**
  Bike = 0.5 | Car = 1.0 | Truck = 1.5

Formula Used

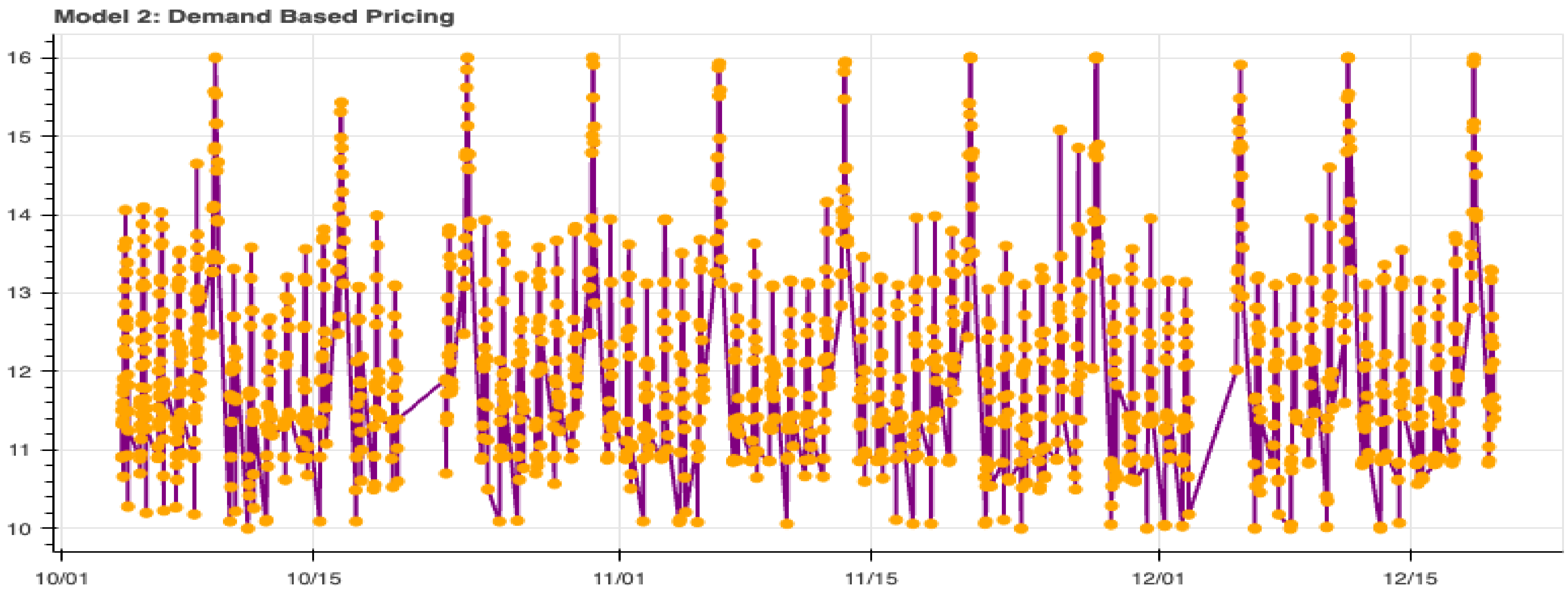
- **Demand Calculation**
  $D = \alpha \cdot Occ + \beta \cdot Queue - \gamma \cdot Traffic + \delta \cdot SpecialDay + \varepsilon \cdot VehicleWeight$
- **Normalized Demand**
  $D\_norm = clamp((D - 0.5) / 3, 0, 1)$
- **Final Price**
  $Price = BASE\_PRICE \times (1 + \lambda \cdot D\_norm)$

Assumptions

- Queue increases urgency
- Traffic increases external wait cost
- Cars/trucks have higher willingness to pay than bikes

Key Takeaways

- **Multi-factor demand-driven pricing**
- **Normalized demand ensures price stability**
- **Dynamic price scaling using λ**
- **Easily tunable for urban scenarios**



Visualization for Model

Model 3

Demand Based Pricing

Strategy

Parameters

- **Constants**
  BASE_PRICE = 10.0 | $\alpha = 0.5$ | $\beta = 0.2$ |
  $\gamma = 0.1$ | $\delta = 1.0$ | $\varepsilon = 0.8$ | $\lambda = 0.6$

- **Vehicle Weights**
  Bike = 0.5 | Car = 1.0 | Truck = 1.5

Formula Used

- **Competitive Price Adjustment (Latitude-based)**
  If **lat > 18.6** and **price > 12** → price -= 1.5
  If **lat < 18.5** → price += 1.0

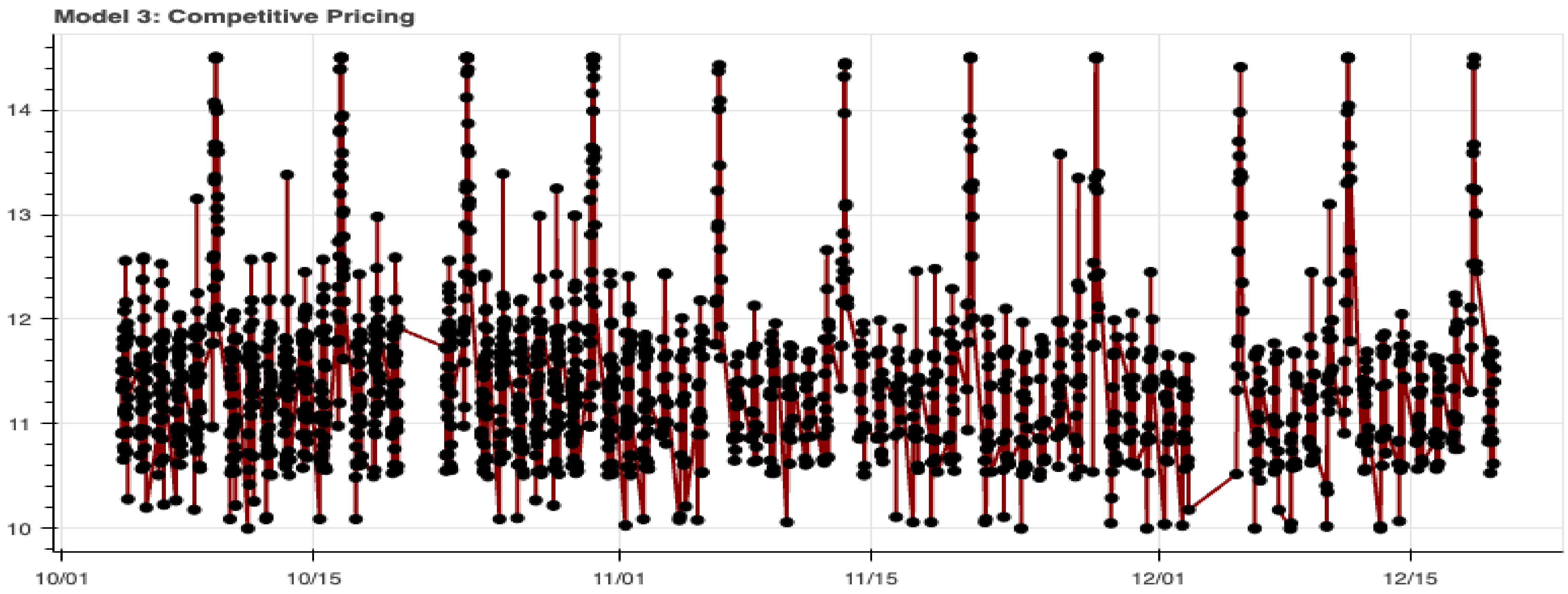
- **Final Price Bound**
  price = clamp(price, 5, 20)

Assumptions

- **Calculate haversine/geodesic distance from other lots**
- **Compare prices of nearby lots within 500m**
- **Adjust current lot's price as:**
  - Decrease if nearby lots are cheaper
  - Increase if nearby lots are more expensive and lot isn't full

Key Takeaways

- **Dynamic pricing adapts to real-time demand factors**
- **Vehicle type, occupancy, and special days influence price**
- **Haversine distance enables location-aware competition check**
- **Nearby lot prices drive competitive adjustments**
- **Final price is normalized and bounded for fairness**



Visualization for Model

Thank You

Template Credits: Microsoft Template