Dynamic Pricing for Urban Parking Lots: Project Report

Introduction

This project addresses the challenge of optimizing urban parking lot utilization through **dynamic, data-driven pricing**. By leveraging real-time data and advanced modeling, the goal is to ensure efficient use of parking spaces, reduce congestion, and provide fair pricing for users. The solution is implemented in Python using Pandas, Numpy, Pathway for real-time simulation, and Bokeh for interactive visualization.

Problem Statement

Urban parking spaces face fluctuating demand throughout the day. Static pricing often leads to either overcrowding or underutilization. The objective was to:

- Develop a dynamic pricing engine for 14 parking spaces.
- Update prices in real time based on:
  - Occupancy history
  - Queue length
  - Traffic level
  - Special events
  - Vehicle type
  - Competitor prices
- Ensure price changes are smooth, explainable, and bounded.
- Visualize results in real time.

Dataset Overview

Each record in the dataset contains:

Feature	**Description**
ID	Unique row index
SystemCodeNumber	Parking lot identifier
Capacity	Maximum vehicles allowed
Latitude, Longitude	Lot location
Occupancy	Current parked vehicles
VehicleType	Type of incoming vehicle (car, bike, truck)
TrafficConditionNearby	Traffic level (low, average, high)
QueueLength	Vehicles waiting
IsSpecialDay	1 if special day/event, else 0
LastUpdatedDate/Time	Timestamp

Implementation Summary

1. Data Ingestion and Preprocessing

- Loaded the dataset into a Pandas DataFrame.
- Ensured all columns matched expected types and names.
- Used Pathway for simulating real-time data ingestion.

2. Feature Engineering

- Converted categorical variables (traffic, vehicle type) into numerical values for modeling.
- Created helper functions for feature mapping.

3. Pricing Models

Model 1: Baseline Linear Model

- **Logic:** Price increases linearly with occupancy rate.
- **Formula:**

$$Price_{t+1} = BasePrice + \alpha \cdot \left(\frac{Occupancy}{Capacity} \right)$$

- **Purpose:** Establishes a simple, explainable baseline.

Model 2: Demand-Based Model

- **Logic:** Incorporates occupancy, queue length, traffic, special days, and vehicle type into a demand function.
- **Formula:**

$$Demand = \alpha \cdot \left(\frac{Occupancy}{Capacity} \right) + \beta \cdot QueueLength - \gamma \cdot Traffic + \delta \cdot IsSpecialDay + \epsilon \cdot VehicleTypeWeight$$

$$Price_t = BasePrice \cdot (1 + \lambda \cdot NormalizedDemand)$$

- **Features:** Demand is normalized; price is bounded (not less than 0.5× and not more than 2× base).

Model 3: Competitive Pricing Model

- **Logic:** Adjusts price based on competitor lots' prices and proximity.
- **Rules:**
  - If the lot is full and competitors are cheaper, suggest rerouting or lower price.
  - If nearby lots are expensive, allow price to increase.
- **Implementation:** Used Euclidean distance for proximity and compared demand-based prices across lots.

4. Batch Processing and Output

- Computed all three prices for each record in the dataset.
- Saved all results to a CSV file for further analysis.

Example Output Table:

ID	SystemCodeNumber	Timestamp	Price_Linear	Price_Demand	Price_Competitive
0	BHMBCCMKT01	04-10-2016 07:59:00	10.53	12.18	12.18
1	BHMBCCMKT01	04-10-2016 08:25:00	10.55	12.24	12.24
2	BHMBCCMKT01	04-10-2016 08:59:00	10.69	12.56	12.56
3	BHMBCCMKT01	04-10-2016 09:32:00	10.93	13.09	13.09
4	BHMBCCMKT01	04-10-2016 09:59:00	11.30	13.87	13.87

Visualization

- Used Bokeh to create interactive line plots of price evolution over time for each parking lot.
- Plots compare all three pricing models, providing visual justification for price changes.
- Addressed Bokeh-specific issues (e.g., duplicate x-axis labels) by deduplicating or aggregating timestamps.
- Additional visualizations included occupancy and queue length over time, price distributions, heatmaps of occupancy, and geographic maps of lots colored by occupancy or price.

Key Challenges and Solutions

- **Slow Processing:** The competitive pricing model's nested looping was slow for large datasets. Optimization suggestions included vectorization and precomputing competitor groups.
- **Pathway Integration:** Used CSV reading for Pathway ingestion, as direct DataFrame conversion was unsupported.
- **Plotting Errors:** Resolved Bokeh's duplicate factor errors by ensuring unique timestamps for categorical axes.

Deliverables

- Well-commented, modular code in notebook cells.
- CSV file containing all computed prices.
- Interactive visualizations for real-time price trends.
- Comprehensive report documenting logic, assumptions, and results.

Conclusion

This project successfully demonstrates a scalable, modular approach to dynamic pricing for urban parking. The models developed are transparent, tunable, and ready for further extension or deployment in real-world scenarios. The workflow, from data ingestion to advanced modeling and visualization, provides a robust foundation for intelligent parking management systems.