Summer Analytics 2025

Capstone Project Report

**Introduction**

Given the task to implement a dynamic pricing model for 14 urban parking spaces, using a dataset collected over 73 days at 18 half-hour intervals from 8:00 AM to 4:30 PM daily. The goal was to simulate real-time price adjustments based on demand and other factors using the Pathway streaming framework.

**Dataset Overview**

* **Location Information**
  + Latitude, Longitude: Geospatial coordinates to assess proximity and clustering of parking spaces.
* **Parking Lot Features**
  + Capacity: Maximum parking slots per space (constant over time).
  + Occupancy: Current vehicles parked (varies per timestamp).
  + QueueLength: Number of vehicles waiting to enter.
* **Vehicle Information**
  + VehicleType: Encoded as numerical values (e.g., bike=0.25, car=0.5, truck=1.0) to represent different space usage and willingness to pay.
* **Environmental Conditions**
  + TrafficConditionNearby: Numeric indicator of congestion level.
  + IsSpecialDay: Binary flag for holidays or special events.
  + fill\_percent = Occupancy / Capacity (normalized demand proxy).
  + Demand: Composite index combining fill\_percent, QueueLength, VehicleType, and IsSpecialDay.

**Model 1**

It is a simple model where the next price is a function of current occupancy and the parking lot’s capacity.

Filtered the dataset to include only one parking lot for focused analysis.

Used Pathway and Bokeh Plot to Visualize real time data

**Pricing Model**

*Price = 10 + 2\*((Occupancy/Capacity)-0.25)*

Base price is set to 10. The price can fluctuate between 9.5 and 11.5 depending on the Occupancy/Capacity ratio.

**Model 2**

**A more advanced model where I have constructed a mathematical demand function by creating new features.**

**1. Data Preprocessing**

* **Timestamp Parsing**: Combine date and time strings into a parsed t column (datetime).
* **Encoding**: Convert VehicleType into numeric scale to reflect differential pricing impact.
* **Feature Engineering**:
  + Calculate fill\_percent for direct occupancy ratio.

**2. Data Analysis**

* **Exploration**
  + Spatial Analysis of each parking space
  + Correlation between features
  + Popularity of vehicles
  + Variation of Queue Length and Fill Percent over time
  + Impact of Special day
* **Inference**
  + Some parking spaces had a lot of outliers for fill percent.
  + Queue Length and Traffic were very highly correlated with each other so I dropped one of the feature while creating the model.
  + Cars were the most frequently parked vehicle type.
  + Periodic variation over each day
  + No impact of Special day seen so didn’t use this feature.

**3. Demand Calculation**

Construct Demand using a weighted combination of normalized features for one parking space.

*Demand = w1·f1(Fill Percent) + w2·f2(Queue Length) + w3·f3(Traffic) + w4·f4(IsSpecialDay) + w5·f5(Vehicle Type)*

Description of Weights and functions

* Used exponential function for Fill Percent and Queue Length; and a linear function for Vehicle Type
* Priority of features: Fill Percent > Queue Length > Traffic
* Didn’t use Is Special Day and Traffic

Calculated Rolling demand for smoother price variation.

**4. Scaling**

Scaled both demand and rolling demand from -0.5 to 4 to ensure desired minimum and maximum values

**5. Pathway Streaming Pipeline**

* **Connector**: Read the pre-processed Data Frame into a Pathway Python Reader.
* **Transform**: Use .with\_columns() on the table to compute price as a new column in real time.
* **Windowing**: For timestamp-level updates, no window aggregation is needed; pricing is row-wise.
* **Output**: The resulting delta\_window table contains (t, SystemCodeNumber, price) for each time point and parking space.

**6. Pricing Model**

*The pricing model was applied to one parking space at a time..*

* **Base Formula**:

Price = BasePrice \* (1 + λ \* Demand)

* + Base Price is set to 10 units
  + λ (lambda): Sensitivity coefficient (0.1) controlling price volatility.

Created both Price and Rolling Price functions using Demand and Rolling Demand

**6.1 Common Pricing**

The common price function is used to plot the price. No price separation for different vehicle types.

**6.2 Specific Pricing for Vehicles**

* Made separate demand function for cycle, bike, car and truck.
* Interpolated all the missing values.
* Standard Prices:
  + Cycle = 2.5 units
  + Bike = 5 units
  + Car = 10 units
  + Truck = 20 units

**7. Real-Time Simulation and Visualization**

* **Panel + Bokeh** is used to render live-updating plots.
* **Custom Plot Function**:
  + Extracts NumPy arrays for time and price.
  + Plots both raw and smoothed price lines.
* **Interactive Legend**:
  + Legend-based highlighting in Matplotlib to focus on individual prices.

**8. Competitor Analysis**

* **Demand Boxplot:** To check popularity of each parking space
* **Time Series Overlays**: Plot all 14 price curves to compare volatility.
* **Ranking Heatmap**: Compute per-timestamp price ranks across spaces.
* **Rankings by**
  + Mean demand
    - Parking with 387 capacity has the most average demand and parking with 2803 capacity has the least average demand.
  + Standard deviation of demand
    - Parking with 387 capacity has the most standard deviation in demand and parking with 2803 capacity has the least.
* **Spatial Plotting**: How far away are parking spaces from each other.
  + Parking with 387 capacity and 577 capacity are very close by however there is a significant difference between their demand. Hence, we can increase the price of the first parking and decrease the price of second to ensure uniform distribution.
  + The same can be said for parking pairs with capacity 3883, 3103 and 1322, 2803

**9. Conclusion**

This pipeline marries a data-driven demand model with a robust streaming architecture and interactive dashboards, providing urban parking operators with actionable, real-time pricing insights and competitive benchmarking.

**Further Comments**

Due to the volume of data across 14 parking lots and 1312 time points, visualizing all lots together would result in overcrowded plots. Therefore, a single parking lot was selected for clearer and more interpretable visual outputs.