**Dynamic Pricing for Urban Parking Lots**

**Summer Analytics 2025 Capstone Project | Consulting & Analytics Club × Pathway**

**✨ Project Overview**

Urban parking lots face inefficiencies due to static pricing. This project aims to build a **real-time dynamic pricing engine** for 14 parking lots based on historical occupancy, demand features, and competitive pressures.

**🌐 Dataset Description**

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

* **Location:** Latitude, Longitude
* **Parking Metrics:** Capacity, Occupancy, Queue Length
* **Vehicle Info:** Type (car, bike, truck)
* **Environment:** Traffic conditions, Special event day indicator
* **Timestamp:** Date + Time (merged into Timestamp column)

**📊 Model 1: Baseline Linear Model**

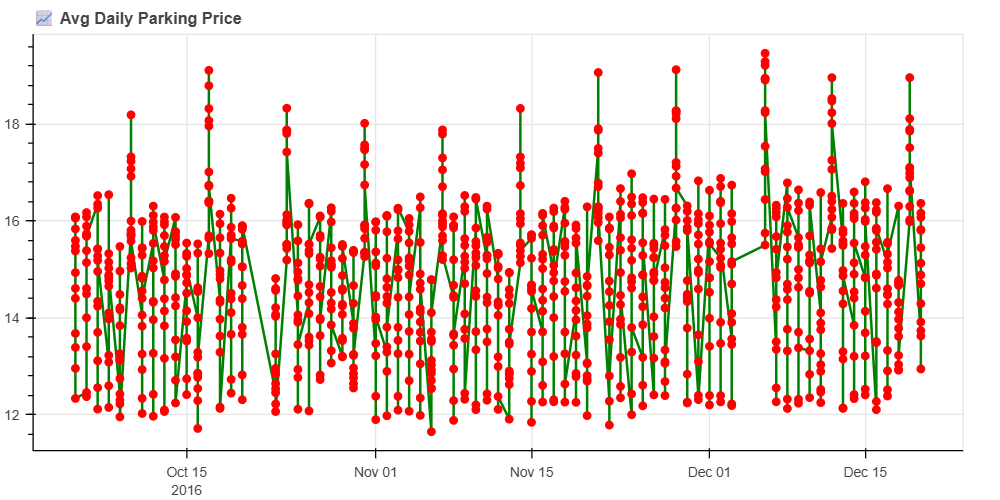
**Pricing Function**

price = 10 + (occ\_max - occ\_min) / capacity

**Assumptions:**

* Price increases with occupancy variation (fluctuation = demand volatility)
* Simple, smooth variation

**Output Visualization:**



### 🌐 Model 2: Demand-Based PricingDemand Function

demand = \

a \* (occupancy / capacity) + \

b \* queue\_length - \

c \* traffic\_level + \

d \* is\_special\_day + \

e \* vehicle\_type\_weight

Where a, b, c, d, e are manually tuned weights.

### Pricing Function

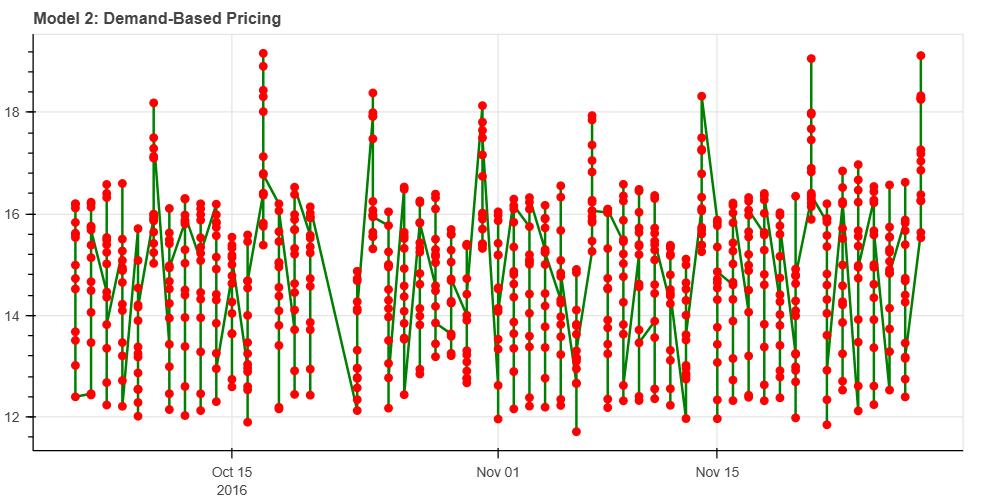
price = base\_price \* (1 + λ \* normalized\_demand)

* Base price = $10
* λ controls sensitivity
* Normalized demand bounds the final price within 0.5x to 2x

### Assumptions:

* Queue increases urgency (positive weight)
* Traffic increases external wait cost (negative weight)
* Cars/trucks have higher willingness to pay than bikes

### Output Visualization:



**🔮 Model 3: Competitive Pricing Model**

**Competitive Logic:**

* 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

**Final Price Logic**

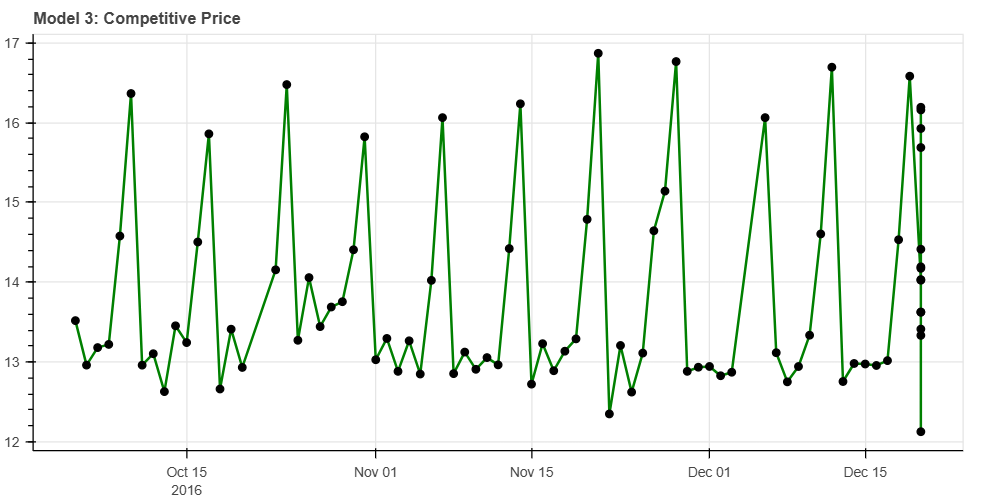
if avg\_nearby < demand\_price:

price = max(avg\_nearby - 1, 5)

else:

price = min(demand\_price + 0.5, 20)

**Output Visualization:**



**✨ Real-Time Execution with Pathway**

* Used pw.demo.replay\_csv to simulate streaming
* Applied tumbling windows for daily aggregates
* All models run and visualize in **Google Colab** using Panel + Bokeh

**🎓 Conclusion**

* **Model 1**: Simple reference based on occupancy
* **Model 2**: Uses interpretable demand features
* **Model 3**: Introduces competitive pricing based on proximity

**Recommendation:**

Model 2 offers the best balance between accuracy and interpretability. Model 3 is effective where parking lots are densely distributed.

**🔧 Technical Notes**

* All models built from scratch using only: **pandas, numpy, Pathway**
* No ML libraries were used (per constraints)
* Compatible with Colab using dynamic file upload

**🗕️ Submission Checklist**

* Well-commented Google Colab notebook
* Real-time data simulation via Pathway
* Three pricing models implemented from scratch
* Visualizations using Bokeh
* Markdown report justifying pricing strategies and logic
* Demand assumptions and tuning parameters explained
* Competitive pricing logic implemented
* Dataset handled via Colab file upload block
* Final .ipynb and .pdf/.md submitted