Project Report

on

ParkSense:

Dynamic Pricing for Urban Parking

Capstone Project of Summer Analytics 2025

Hosted by Consulting & Analytics Club, IIT Guwahati & Pathway

By:

Shanvi Vats

Electronics and Communications Engineering

Birla Institute of Technology, Mesra

Date: July 5, 2025

Table of Contents

1. Introduction

Urban parking spaces represent a finite and highly sought-after resource. Traditional static pricing models often lead to significant inefficiencies, manifesting as either severe overcrowding during peak hours or underutilization during off-peak times. These inefficiencies result in lost revenue for parking operators and frustration for drivers.

The ParkSense project addresses this challenge by simulating an intelligent, data-driven dynamic pricing system. The core objective is to develop a pricing engine that adjusts parking rates in real-time based on a comprehensive set of factors, including current demand, competitive landscape, and environmental conditions. This system aims to improve parking space utilization, optimize revenue, and enhance the overall urban mobility experience.

This report details the design, implementation, and justification of the ParkSense dynamic pricing system, including its architecture, the underlying machine learning models built from scratch, and its real-time visualization capabilities.

2. Data Description and Preprocessing

The project utilizes a dataset collected from 14 urban parking spaces over 73 days, with data points sampled at 30-minute intervals from 8:00 AM to 4:30 PM daily. Each record in the dataset.csv includes the following information:

- **Location Information:** Latitude and Longitude (for calculating proximity to competitors).
- **Parking Lot Features:** Capacity (maximum vehicles), Occupancy (current vehicles), and Queue Length (vehicles waiting for entry).
- **Vehicle Information:** Type of incoming vehicle (car, bike, truck).
- **Environmental Conditions:** Nearby traffic congestion level, Special Day indicator (e.g., holidays, events).

Data Preprocessing Steps:

Before feeding the data into the real-time processing pipeline, the following preprocessing steps were applied using Pandas and NumPy:

1. **Timestamp Creation:** The LastUpdatedDate and LastUpdatedTime columns were combined and converted into a single Timestamp column of datetime objects, enabling chronological sorting and time-based analysis.
2. **Data Sorting:** The DataFrame was sorted by the newly created Timestamp to ensure data integrity for sequential processing.
3. **Categorical Feature Encoding:**
    - VehicleType (cycle, bike, car, truck) was label encoded into numerical values (0, 1, 2, 3).
    - TrafficConditionNearby (low, average, high) was label encoded into numerical values (0, 1, 2). This conversion is necessary for numerical models.
4. **Feature Selection for Streaming:** Relevant columns were selected and saved to parking_data_stream.csv, which serves as the input for the Pathway real-time simulation.

3. Technology Stack

The ParkSense project is built upon a robust set of Python libraries and tools:

- **Python**
- **NumPy**
- **Pandas**
- **Pathway**
- **Bokeh**
- **Panel**
- **Google Colab**
- **Mermaid.js**

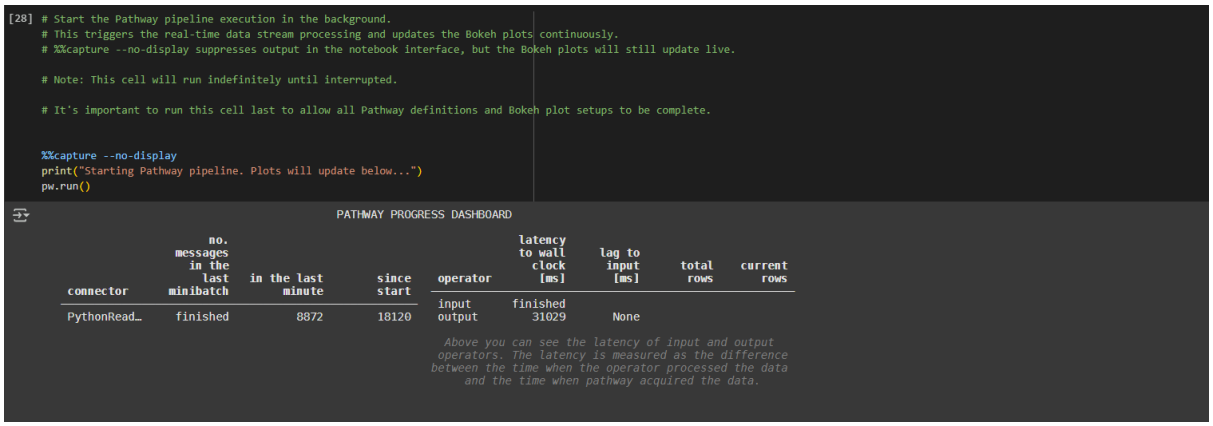
4. System Workflow

1. **Data Preparation (Offline):**
    - The dataset.csv is initially processed using Pandas and NumPy to clean, transform, and feature-engineer the historical data. This includes parsing timestamps, encoding categorical variables, and normalizing numerical features.
    - A critical step in this phase is the **offline training of Model 2's demand function coefficients** using linear regression on the historical data. This ensures the demand model is data-driven.
    - The prepared data is then saved as parking_data_stream.csv.
2. **Real-Time Data Ingestion with Pathway:**
    - Pathway's pw.demo.replay_csv function simulates a real-time data stream by replaying parking_data_stream.csv. This mimics a continuous flow of live parking updates, preserving timestamp order.
    - A Pathway Schema (ParkingSchema) is defined to ensure correct data type interpretation and structure for the incoming stream.
3. **Pathway Real-time Processing Engine:**
    - As data records flow into Pathway, they are processed in real-time. The data_with_time Pathway table is created, adding a proper datetime object and a day identifier for temporal analysis.
    - This stream then feeds into the dynamic pricing models.
4. **Dynamic Pricing Models:**
    - For each incoming record, three distinct pricing models calculate a proposed price:
        - **Model 1 (Baseline Linear):** Applies a simple linear adjustment based on occupancy.
        - **Model 2 (Demand-Based):** Calculates a demand score using the pre-trained coefficients and adjusts the price accordingly, ensuring smoothness and boundedness.

- **Model 3 (Competitive):** Extends Model 2's price by applying adjustments based on the lot's occupancy and a hypothetical competitor's average price.

- The outputs of all three models, along with key input features, are combined into a single combined_prices Pathway table.

5. **Real-time Visualization with Bokeh and Panel:**
   - The combined_prices Pathway table is linked to Bokeh through Panel.
   - For each unique parking spot, a dedicated Bokeh plot is generated, displaying the real-time price trends from all three models.
   - Panel aggregates these plots into an interactive dashboard, which is served directly within the Google Colab environment, providing live visual justification of the pricing behavior.
6. **Pipeline Execution:**
   - The pw.run() command initiates the Pathway pipeline, enabling continuous data processing and real-time updates to the Bokeh/Panel dashboard.



5. Dynamic Pricing Models

The project implements three pricing models, each increasing in complexity and intelligence, all built from scratch using NumPy and Pandas for their core logic. The base price for all models is set at **$10.0**.

5.1. Model 1: Baseline Linear Model

This is a simple linear model serving as a reference point. It adjusts the next price based on the current occupancy rate.

Formula:

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

- BASE_PRICE: The initial base price ($10.0).
- alpha **(alpha)**: A fixed coefficient that determines the sensitivity of the price increase to the occupancy rate.
- Occupancy/Capacity: The current occupancy rate of the parking space.

Justification for alpha=5.0:

The value alpha=5.0 was chosen as a simple, illustrative constant. It implies that for a 100% occupied lot, the price would increase by $5.0 above the base price. This provides a clear, predictable baseline behaviour against which more complex models can be compared. It is a manually tuned parameter reflecting a straightforward linear response to occupancy.

5.2. Model 2: Demand-Based Price Function

This model constructs a mathematical demand function using key features and then uses this demand value to adjust prices.

Demand Function Formula:

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

Price Adjustment Formula:

$$\text{Price}_t = \text{BasePrice} \cdot (1 + \lambda \cdot \text{NormalizedDemand})$$

- BASE_PRICE: The initial base price ($10.0).
- alpha_occ,beta_queue,gamma_traffic,delta_special,epsilon_vehicle: Coefficients representing the impact of each feature on demand. These are **learned from historical data**.
- QueueLength_Normalized,TrafficLevel_Normalized,VehicleType_Normalized: Normalized values of the respective features (scaled between 0 and 1).
- lambda **(lambda)**: A factor controlling the sensitivity of the price to the NormalizedDemand.

Determining Coefficients (alpha_occ,beta_queue,gamma_traffic,delta_special,epsilon_vehicle**):** These coefficients were determined using a **Linear Regression model implemented from scratch** using NumPy's Normal Equation method on the historical dataset.csv.

1. **Demand Proxy Definition:** Since "demand" is not directly available, a Demand_Proxy was defined as:

   Demand_Proxy=(Occupancy+QueueLength)/Capacity

   This proxy assumes that higher occupancy and longer queues indicate higher demand for the parking space.

2. **Feature Preparation:** The following features were prepared for the regression model:
   - Occupancy_Rate: Occupancy / Capacity
   - QueueLength_Normalized: QueueLength divided by its maximum observed value (15.0).

- Traffic_Normalized: Traffic_Encoded divided by its maximum value (2.0).
- IsSpecialDay: Binary indicator (0 or 1).
- Vehicle_Normalized: Vehicle_Encoded divided by its maximum value (3.0).

3. **Linear Regression (Normal Equation):** The coefficients (beta) were calculated using the formula: $\text{beta}=(X^T X)^{-1} X^T y$ where X is the design matrix (features plus an intercept term) and y is the Demand_Proxy.

Learned Coefficients:

Occupancy Rate (alpha_occ): 1.0076

Queue Length (beta_queue): -0.0082

Traffic Condition (gamma_traffic): 0.0078

Special Day (delta_special): 0.0033

Vehicle Type (epsilon_vehicle): 0.0000

Interpretation of Learned Coefficients:

- **Occupancy Rate (**alpha_occ **= 1.0076):** This positive and relatively large coefficient indicates a strong direct relationship between occupancy rate and the demand proxy. As the parking lot fills up, the model predicts a significant increase in demand, which is highly intuitive.
- **Queue Length (**beta_queue **= -0.0082):** This coefficient presents a counter-intuitive negative sign. While a longer queue typically signifies higher demand, the linear regression model suggests a slight *decrease* in the demand proxy as queue length increases, when other factors are held constant. This could be attributed to:
    - **Proxy Definition:** The Demand_Proxy includes QueueLength directly, but its interaction with Occupancy and Capacity in the historical data might lead to this.
    - **Behavioral Deterrence:** It's plausible that beyond a certain point, excessively long queues deter potential customers, leading to a perceived drop in demand.

    - **Multicollinearity:** High correlation between QueueLength and other features (e.g., high occupancy often leads to queues) can sometimes result in unstable or unexpected coefficient signs in linear models. Further investigation into feature correlations and potential non-linear relationships would be beneficial.

- **Traffic Condition (**gamma_traffic **= 0.0078):** A positive coefficient, indicating that higher nearby traffic congestion is associated with a slight increase in the demand proxy. This aligns with expectations, as more traffic might mean more drivers looking for parking.
- **Special Day (**delta_special **= 0.0033):** A positive coefficient, suggesting that special days are associated with a slight increase in the demand proxy. This is intuitive, as such days typically draw more people and thus more parking demand.
- **Vehicle Type (**epsilon_vehicle **= 0.0000):** A coefficient very close to zero, implying that the encoded vehicle type has negligible linear impact on the demand proxy within this model. This could mean that, for the given dataset, the specific type of vehicle does not significantly influence the overall parking demand when other factors are considered.

How Price Changes with Demand: The NormalizedDemand (clipped between 0 and 2, then scaled by 0.5) directly influences the price. A higher NormalizedDemand leads to a higher price, and vice-versa. The lambda_factor controls the sensitivity.

Justification for lambda=0.8**:** The lambda_factor = 0.8 is a hyperparameter chosen to balance price responsiveness with stability. A value of 0.8 allows for a significant price adjustment based on demand (up to 80% increase over base price if NormalizedDemand is ensuring the system reacts dynamically to market conditions without causing excessively erratic price swings. This value can be further optimized through A/B testing or more advanced simulations in a production environment.

5.3. Model 3: Competitive Pricing Model

This model incorporates location intelligence to simulate real-world competition, building upon the price generated by Model 2.

Competitive Logic: The model adjusts the Model 2 price based on the current occupancy rate and a comparison with a hypothetical average competitor price.

- If the parking lot is **nearly full** (occupancy_rate > 0.8) AND its Model 2 price is **higher** than the hypothetical_competitor_avg_price, the price is **slightly reduced** by $0.50 to remain competitive and avoid losing customers. This strategy aims to capture demand even at high occupancy by offering a more attractive price relative to competitors.
- If the parking lot is **very empty** (occupancy_rate < 0.3) AND its Model 2 price is **lower** than the hypothetical_competitor_avg_price, the price is **slightly increased** by $0.50 to capture more revenue, as there's room to increase without deterring customers, given that nearby options are more expensive.
- Otherwise, no competitive adjustment is made.

Justification for hypothetical_competitor_avg_price = 12.0: The value 12.0 for the hypothetical_competitor_avg_price is an illustrative placeholder. It's chosen to be slightly higher than the BASE_PRICE ($10.0), simulating a competitive environment where other parking lots might generally charge more. This allows the competitive adjustment logic to be demonstrated. In a full implementation, this would be a dynamically calculated value from real-time competitor data.

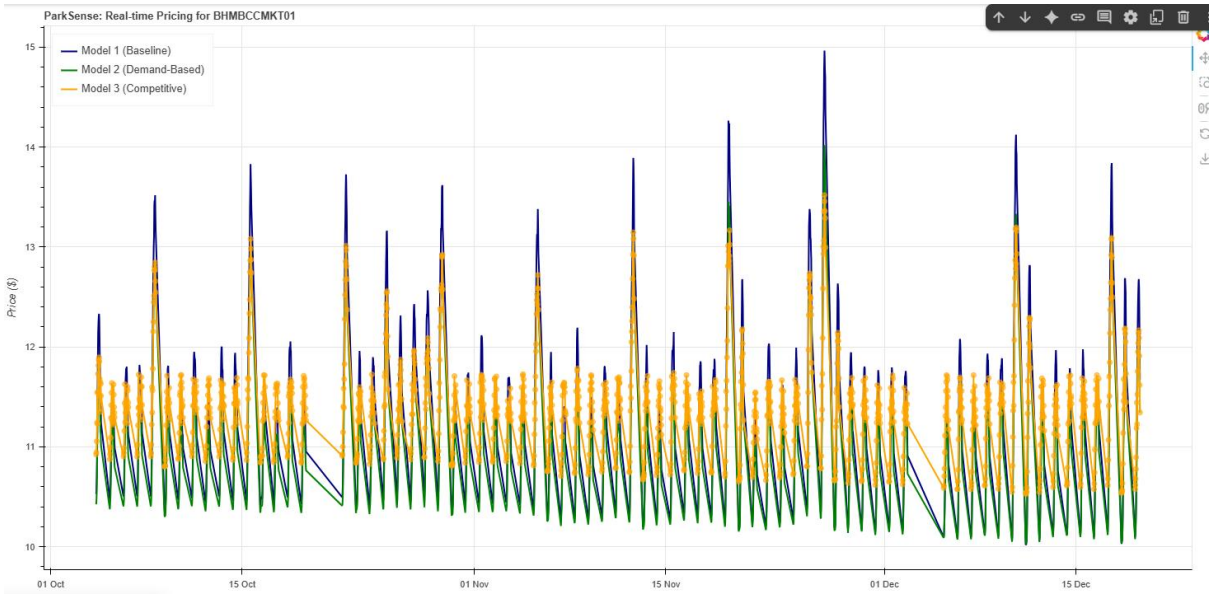
How Price Changes with Competition: This model introduces a reactive element to pricing based on external market conditions. It aims to:

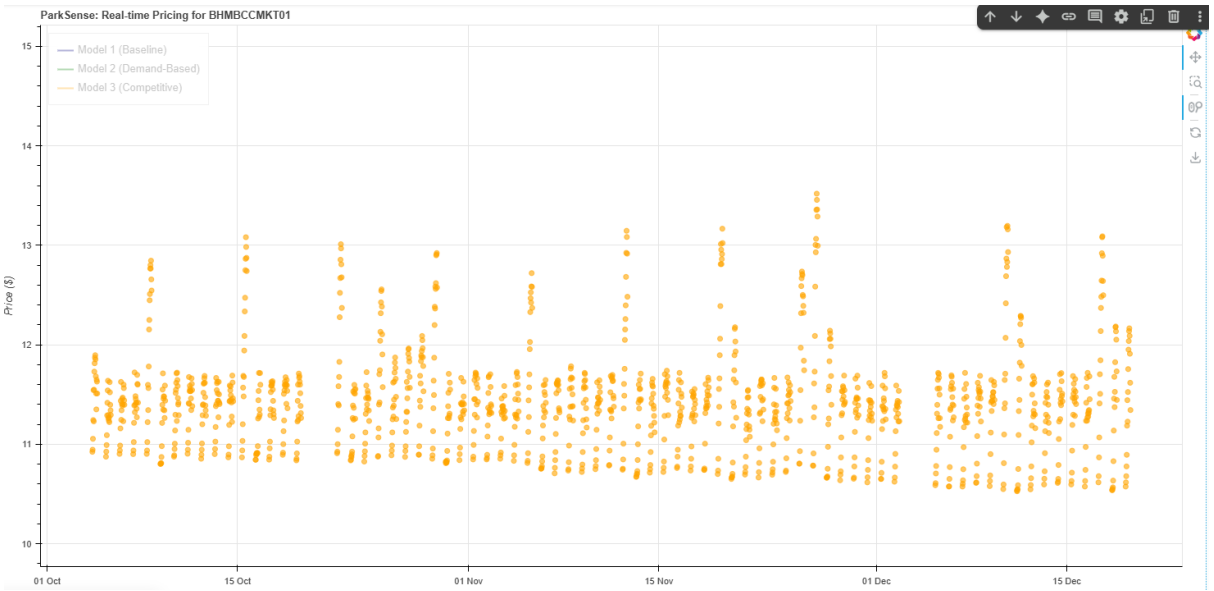
- **Prevent Underpricing:** Increase prices when the lot is empty but still cheaper than competitors, maximizing revenue potential.
- **Prevent Overpricing (when full):** Reduce prices slightly when full but more expensive than competitors, ensuring continued customer flow and preventing complete diversion to competitors. This demonstrates a basic strategic response to the competitive landscape.

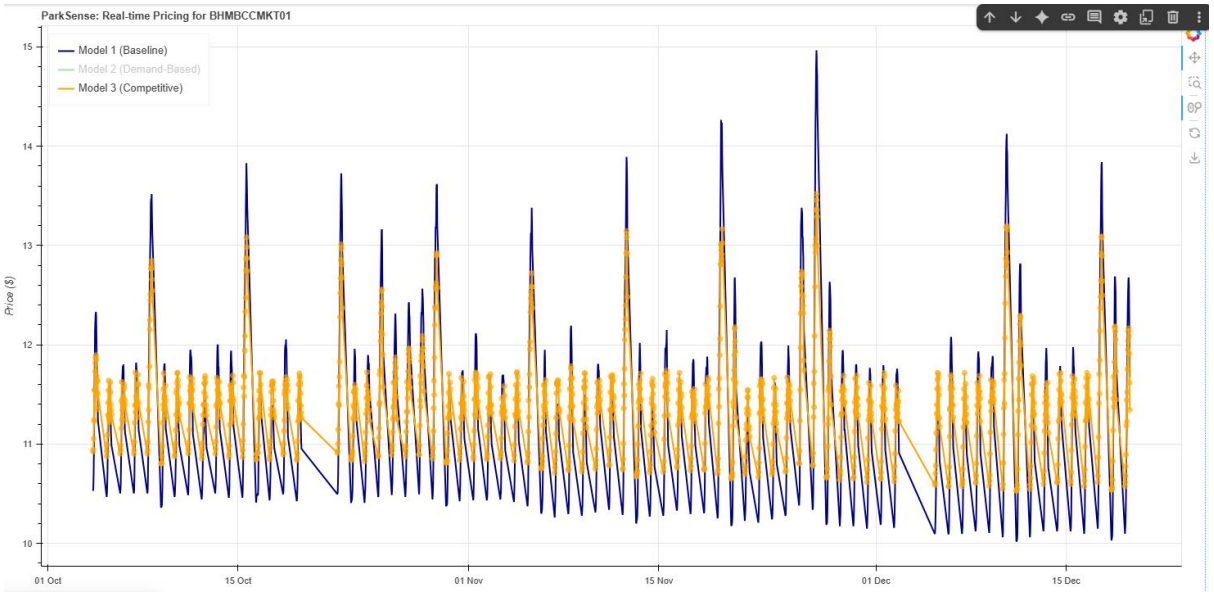
6. Real-Time Simulation and Visualization

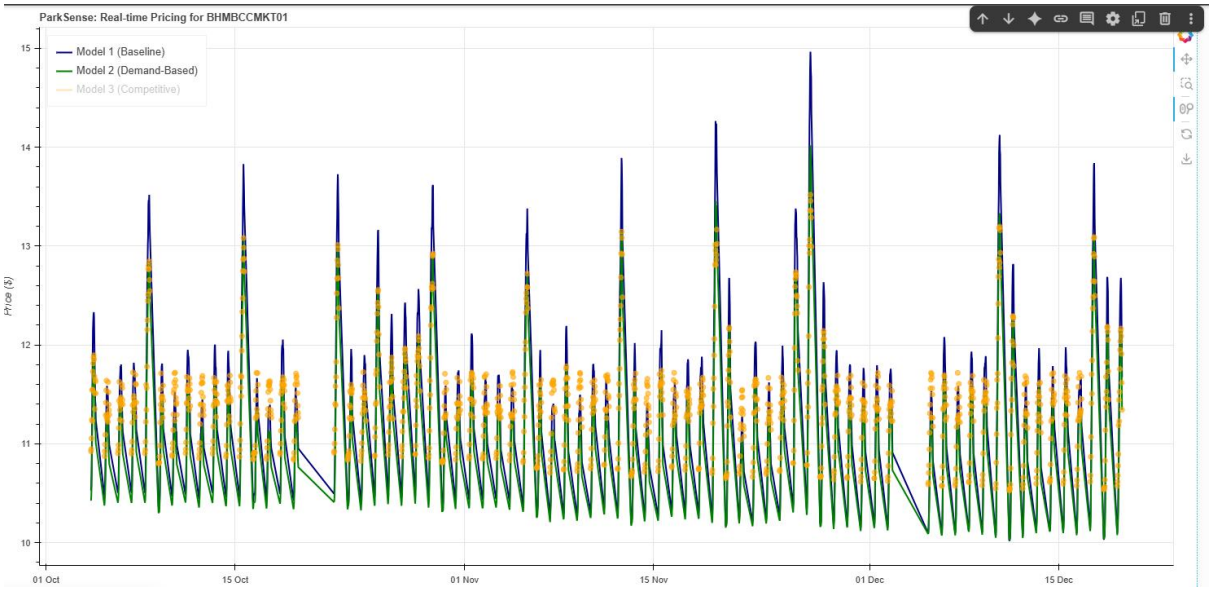
The project simulates real-time data ingestion and provides continuous visual feedback.

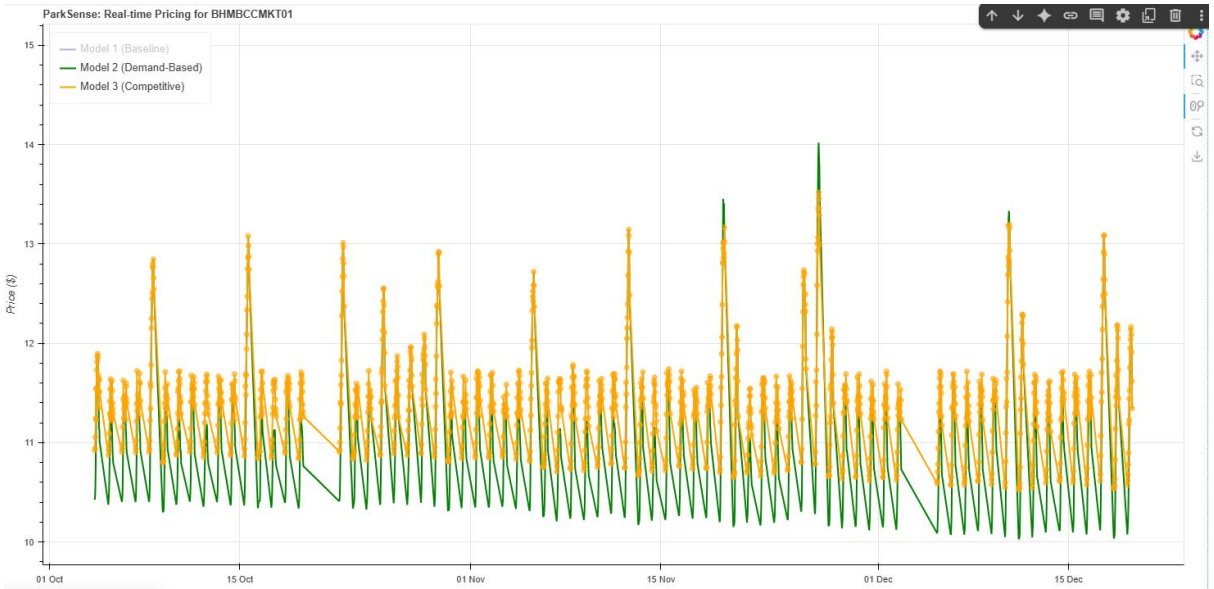
- **Pathway for Real-Time Simulation:** Pathway is central to simulating the real-time environment. It ingests the parking_data_stream.csv with a controlled delay (input_rate=1000), preserving the timestamp order. It continuously processes features and emits pricing predictions as data arrives.
- **Bokeh for Real-Time Visualizations:** Bokeh is used to create interactive line plots that display the real-time prices generated by each of the three models for every parking space.
- **Panel for Dashboarding:** Panel integrates these Bokeh plots into a dynamic dashboard, which is served directly within the Google Colab environment. This allows for live monitoring and visual justification of the pricing behavior across all 14 parking spaces.


ParkSense: Real-time Pricing for BHMBCCMKT01
Model 1 (Baseline)
Model 2 (Demand-Based)
Model 3 (Competitive)
Price ($)
15
14
13
12
11
10
01 Oct
15 Oct
01 Nov
15 Nov
01 Dec
15 Dec


ParkSense: Real-time Pricing for BHMBCCMKT01
Model 1 (Baseline)
Model 2 (Demand-Based)
Model 3 (Competitive)
Price ($)
15
14
13
12
11
10
01 Oct
15 Oct
01 Nov
15 Nov
01 Dec
15 Dec


ParkSense: Real-time Pricing for BHMBCCMKT01
Model 1 (Baseline)
Model 2 (Demand-Based)
Model 3 (Competitive)
Price ($)
15
14
13
12
11
10
01 Oct
15 Oct
01 Nov
15 Nov
01 Dec
15 Dec

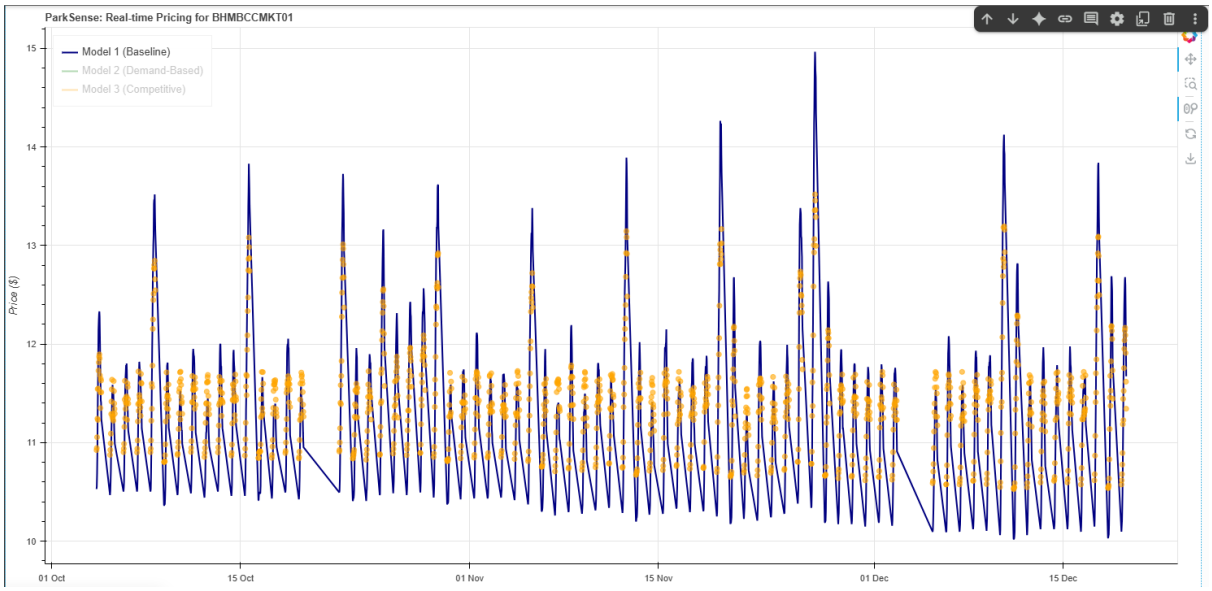

ParkSense: Real-time Pricing for BHMBCCMKT01
Model 1 (Baseline)
Model 2 (Demand-Based)
Model 3 (Competitive)
Price ($)
15
14
13
12
11
10
01 Oct
15 Oct
01 Nov
15 Nov
01 Dec
15 Dec


ParkSense: Real-time Pricing for BHMBCCMKT01
Model 1 (Baseline)
Model 2 (Demand-Based)
Model 3 (Competitive)
Price ($)
15
14
13
12
11
10
01 Oct
15 Oct
01 Nov
15 Nov
01 Dec
15 Dec

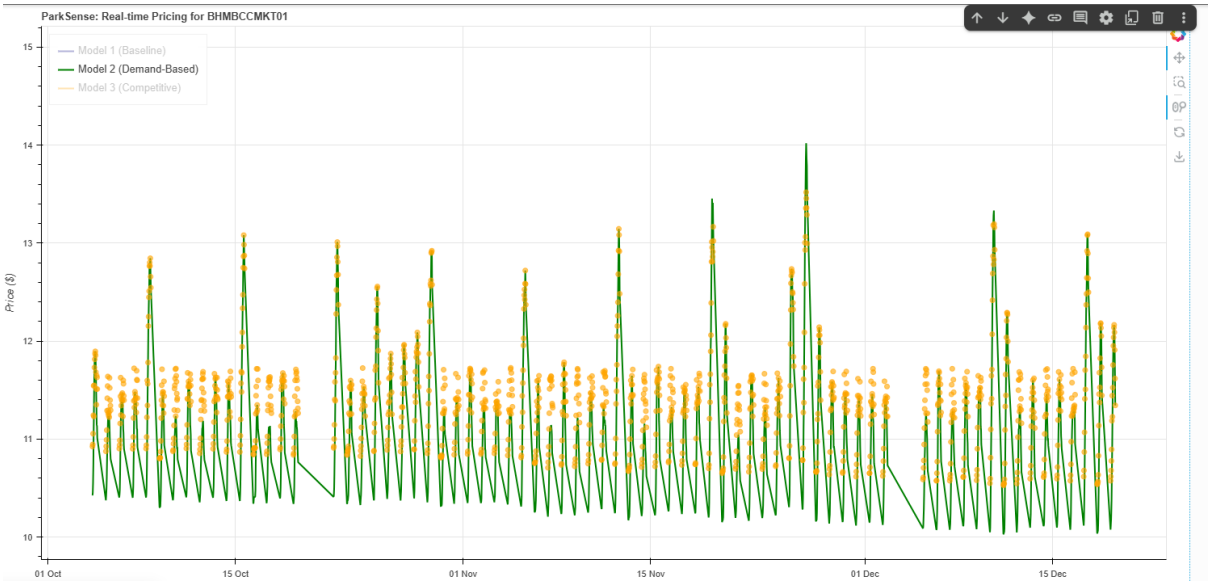
6.1. Visual Justification of Pricing Behavior

The Bokeh plots visually demonstrate how prices fluctuate in response to the incoming data and the logic of each pricing model. Based on the implemented models, the visualizations are expected to show:

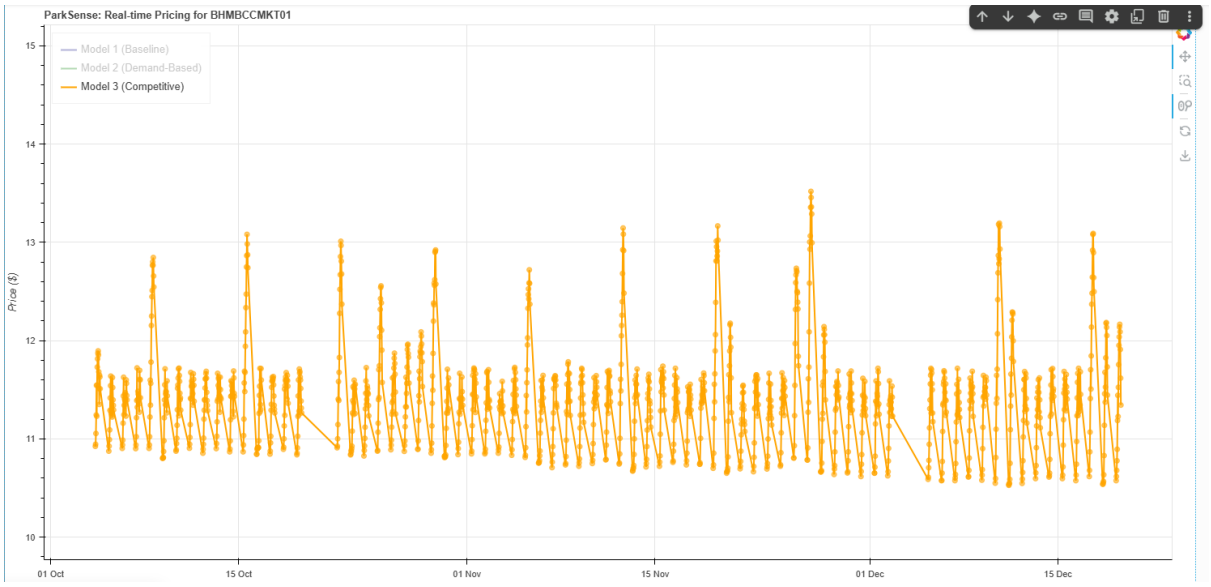
- **Model 1 (Baseline Linear):** A relatively smooth price curve that directly correlates with the occupancy rate of the parking lot. As occupancy increases, the price will rise linearly, and as it decreases, the price will fall. This provides a clear, predictable baseline.



- **Model 2 (Demand-Based):** A more dynamic price curve compared to Model 1. Prices will generally be higher during periods of high demand (e.g., peak morning/afternoon hours, high traffic congestion, special days) and lower during off-peak times. The influence of QueueLength might be subtle or counter-intuitive as per the learned coefficient, but the overall demand proxy will drive these changes.



- **Model 3 (Competitive):** This price curve will largely follow Model 2's trend but will exhibit subtle adjustments. You would observe small price reductions when the lot is highly occupied and its Model 2 price is above the hypothetical competitor's price. Conversely, you might see small price increases when the lot is under-occupied but still cheaper than the hypothetical competitor. This demonstrates the model's attempt to strategically position its price within the market.



7. Assumptions Made and Their Implications

Throughout the development of the ParkSense system, several assumptions were made, each with specific implications for the model's behavior and applicability:

- **Demand Proxy Definition:**
    - **Assumption:** The Demand_Proxy was defined as (Occupancy + QueueLength) / Capacity.
    - **Implication:** This is a simplified representation. It assumes that demand is a direct function of how full the lot is and how many cars are waiting. It might not capture other aspects like booking trends, customer willingness-to-pay, or external factors not in the dataset. A more sophisticated proxy might involve predicting future occupancy or actual vehicle entry rates.
- **Linear Relationships:**
    - **Assumption:** The demand function in Model 2 and the price adjustment in Model 1 assume linear relationships between features and demand/price.
    - **Implication:** Real-world relationships are often non-linear. For example, the impact of queue length on demand might not be linear; a very long queue could deter customers more significantly than a moderately long one. Using linear models might limit the accuracy and responsiveness of the pricing in complex scenarios.
- **Fixed Parameters for Models:**
    - **Assumption:** Model 1's alpha (5.0), Model 2's lambda_factor (0.8), and Model 3's hypothetical_competitor_avg_price (12.0) are fixed, illustrative values.
    - **Implication:** These parameters are not dynamically learned or optimized. In a production system, these would ideally be determined through:
        - **Optimization Algorithms:** Solving for values that maximize specific business objectives (e.g., revenue,

utilization). Their fixed nature means the system's adaptability is limited to the initial tuning.

- **Normalization Max Values:**
    - **Assumption:** The maximum values used for normalizing QueueLength, TrafficConditionNearby_Encoded, and VehicleType_Encoded (15.0, 2.0, 3.0 respectively) are based on dataset inspection and typical ranges.
    - **Implication:** If future data contains values exceeding these assumed maximums, the normalization will be incorrect, potentially distorting the feature's impact on demand. In a robust system, these maximums would be dynamically updated or derived from a more comprehensive statistical analysis of the entire data distribution.
- **No External Data Sources:**
    - **Assumption:** The simulation relies solely on the provided dataset.csv and does not integrate with actual live APIs for real-time traffic updates, external event calendars, or competitor pricing feeds.
    - **Implication:** The system's real-time intelligence is limited to the features available in the historical dataset. A true dynamic pricing system would benefit significantly from real-time external data sources for greater accuracy and responsiveness to unforeseen events.
- **Simplified Competitive Logic:**
    - **Assumption:** Model 3's competitive logic is a basic illustration using a static hypothetical competitor price.
    - **Implication:** This significantly limits its real-world applicability. It does not account for the actual number of competitors, their precise locations, their real-time prices, or their current occupancy levels. It also doesn't consider dynamic rerouting suggestions based on real-time availability across multiple lots.

8. Limitations

Based on the assumptions and the scope of this capstone project, the current ParkSense system has the following limitations:

- **Static Competitive Data:** Model 3 uses a hypothetical competitor price, which does not reflect real-time market dynamics. This prevents truly adaptive competitive pricing.
- **No External API Integration:** The system does not connect to live external data sources (e.g., Google Maps for real-time traffic, event APIs) that would significantly enhance its predictive power and real-time responsiveness.
- **Linear Model Constraints:** The demand function is based on linear regression, which may not fully capture complex, non-linear relationships between pricing factors and demand in a real-world scenario.
- **Limited Optimization:** The parameters (alpha, lambda_factor, hypothetical_competitor_avg_price) are manually set or derived once. There is no continuous optimization loop to refine these parameters based on the system's performance (e.g., revenue generated, utilization rates).
- **No Predictive Demand Forecasting:** The demand proxy is based on current conditions. A more advanced system would include predictive models to forecast future demand, allowing for proactive pricing adjustments.
- **No Multi-Lot Coordination:** Model 3's competitive logic is per-lot and doesn't involve a centralized decision-making process that coordinates pricing across all 14 parking spaces to optimize the entire network.
- **Scalability:** While Pathway handles streaming, the current model training (NumPy/Pandas) is an offline batch process. For extremely large datasets or more complex models, a more scalable training infrastructure would be needed.

9. Future Work

To evolve ParkSense into a more robust and commercially viable dynamic pricing solution, the following enhancements are recommended:

- **Advanced Competitive Intelligence:**
    - Integrate real-time data from actual competitor parking lots (e.g., via web scraping or APIs).
    - Implement spatial analysis using haversine_distance to identify and factor in prices of *nearest* competitors dynamically.
    - Develop strategies for price matching, undercutting, or premium pricing based on real-time competitive analysis.
- **External Data Integration:**
    - Connect to real-time traffic APIs (e.g., Google Maps Traffic API) for more accurate congestion data.
    - Incorporate external event calendars or local news feeds to dynamically identify and react to special events.
    - Explore weather data integration, as weather can significantly impact parking demand.
- **Non-Linear Demand Modeling:**
    - Experiment with more sophisticated machine learning models (e.g., Gradient Boosting Machines, Neural Networks) that can capture non-linear relationships between features and demand, potentially using libraries like scikit-learn (if allowed by project constraints) or implementing simpler non-linear transformations with NumPy.
- **Reinforcement Learning for Price Optimization:**
    - Implement a reinforcement learning agent that learns optimal pricing strategies over time by interacting with a simulated environment or A/B testing in a live setting. This would allow the system to continuously adapt and maximize long-term objectives (e.g., revenue, utilization).
- **Predictive Demand Forecasting:**
    - Develop time-series forecasting models to predict future occupancy and queue lengths, enabling proactive price adjustments rather than purely reactive ones.

- **Network-Wide Optimization:**
    - Extend Model 3 to a centralized optimization framework that considers all 14 parking spaces simultaneously, coordinating prices to balance demand across the entire network and maximize overall revenue or utilization.
    - Implement vehicle rerouting suggestions based on real-time availability and optimal pricing across the network.
- **User Interface Enhancements:**
    - Develop a more comprehensive and user-friendly dashboard with additional metrics (e.g., historical revenue, utilization rates, predicted demand).
    - Add controls for manual override or parameter tuning.

10. Conclusion

The ParkSense project successfully demonstrates a dynamic pricing engine for urban parking lots using real-time data streams and custom-built machine learning models. By leveraging Pathway for efficient data processing and Bokeh/Panel for interactive visualizations, the system provides a clear and justifiable approach to adjusting parking prices based on demand and competitive factors.

The implementation of three progressive models highlights how increasing complexity can lead to more intelligent pricing strategies. While certain aspects, particularly the competitive model, are simplified for this capstone, the project lays a strong foundation for developing a comprehensive and robust dynamic pricing solution for real-world urban parking challenges. The real-time visualizations offer valuable insights into the system's behavior, allowing for transparent analysis and justification of pricing decisions.

This project serves as a compelling proof-of-concept for the application of streaming analytics and machine learning in optimizing urban infrastructure.