DYNAMIC PRICING FOR URBAN PARKING LOTS

(SUMMER ANALYTICS 2025 CAPSTONE PROJECT)

SUBMITTED BY :

NALIN GOEL

AN INITIATIVE BY:

CONSULTING & ANALYTICS CLUB,

IIT GUWAHATI

2. TABLE OF CONTENTS

3. EXECUTIVE SUMMARY

Urban parking lots in busy metropolitan zones frequently experience either over-congestion or underuse due to fixed pricing systems that do not reflect real-time demand variations. To tackle this challenge, the current project introduces a dynamic pricing engine that adapts parking rates based on real-time indicators to improve utilization and revenue.

The system intelligently reacts to factors such as live occupancy levels, queue build-up, surrounding traffic flow, vehicle categories, and special event days. It employs a tiered modelling approach with three progressively smarter pricing strategies:

● **Model 1**: A basic linear approach where prices scale proportionally with occupancy rates.

● **Model 2**: A demand-driven algorithm that calculates prices through a weighted sum of relevant features, normalized over each hour to capture demand shifts accurately.

● **Model 3**: A competition-aware strategy that factors in prices of nearby parking facilities based on geographic distance, enabling adaptive pricing in response to local market conditions.

The pricing engine was tested in a simulated environment on Google Colab, using row-wise data streaming to emulate real-time operation. Visualization of pricing outcomes and comparative model performance was achieved through interactive dashboards built with Bokeh and Panel.

Performance evaluation revealed important trade-offs among the three models. **Model 1** showed the highest price-demand correlation (≈ 0.1939), indicating its effectiveness in tracking occupancy, while also achieving the **highest estimated revenue** of approximately **176.9 million**. **Model 2**, though having a lower correlation (≈ 0.0842), introduced more price flexibility (highest price variance and range) and still managed a strong revenue performance (~173.2 million), making it the most **adaptive to demand variability**. **Model 3** yielded the **lowest correlation** (≈ 0.0103) due to its focus on competitive parity rather than direct demand, but its revenue (~176.85 million) closely matched Model 1, suggesting it **successfully balances market competitiveness with profitability**. Overall, the project demonstrates how each pricing strategy offers distinct advantages, and dynamic, context-aware pricing systems can significantly enhance both fairness and financial outcomes in urban parking management.

4. INTRODUCTION

The rapid pace of urban growth and the surge in vehicle ownership have intensified the pressure on parking infrastructure in major metropolitan areas. Traditional parking systems often rely on **fixed-rate pricing**, which does not reflect real-time shifts in demand, traffic flow, or the impact of special events. This disconnect between pricing and actual usage leads to inefficiencies: during off-peak hours, parking spaces go unused, while during peak demand, lots become overcrowded, resulting in user frustration, revenue loss, and increased traffic congestion and emissions.

To overcome these limitations, many cities are now turning toward **dynamic pricing systems**—models that modify parking fees in real time based on prevailing conditions. These systems seek to optimize space usage, boost revenue for operators, and make parking more accessible and predictable for drivers. However, implementing such a system demands a **robust, data-centric architecture** capable of capturing real-time variability and generating responsive yet equitable pricing adjustments.

This capstone project, developed under **Summer Analytics 2025**, presents the design and simulation of a dynamic pricing engine for urban parking facilities. The proposed solution revolves around three progressively advanced models:

- A **basic linear model** that changes prices based on occupancy levels alone,
- A **feature-rich demand model** incorporating queue lengths, traffic density, vehicle type, and special events, and
- A **market-aware competitive model** that integrates pricing strategies of neighboring lots based on spatial proximity.

Using historical multi-lot parking data, the project mimics a real-time pricing scenario in Python, executed via Google Colab. Real-time behavior is simulated through row-wise streaming, and interactive visualizations are created using Bokeh to monitor pricing evolution and evaluate model performance. Metrics such as revenue, correlation with demand, and price variability were used to benchmark results.

This report details the full journey—from conceptualization and modeling to simulation and analysis—offering a comprehensive view of how dynamic pricing can make city parking systems **smarter, more balanced, and economically optimized** for both providers and users.

5. PROBLEM STATEMENT

In urban environments, parking has become an essential yet increasingly scarce resource. With the continuous rise in vehicle numbers and limited expansion of available space, managing parking availability effectively has emerged as a major challenge for city authorities and infrastructure managers.

Most city parking lots still operate under **static pricing models**, where charges remain fixed throughout the day, regardless of actual demand, traffic levels, or contextual factors. Although easy to implement, this pricing strategy often fails to capture the dynamic nature of urban parking demand. During peak hours, lots may reach capacity quickly, causing driver frustration, increased congestion, and time loss. In contrast, during quieter periods, many spaces sit empty—despite ongoing operational expenses—leading to **inefficient utilization and revenue shortfalls**.

Static models also ignore critical **contextual influences**, such as:
- Time of day or day of the week
- Real-time occupancy and queue build-up
- Nearby traffic intensity (low/medium/high)
- Special days such as holidays or local events
- Prices at neighboring parking facilities

By failing to incorporate these variables, operators miss opportunities to optimize pricing and improve both profitability and customer experience. The need is clear: a **smarter, adaptable pricing system** that can adjust rates dynamically based on real-time inputs.

This project addresses the challenge by developing a **dynamic pricing engine** tailored for urban parking lots. It simulates real-time decision-making using historical data while implementing rule-based logic grounded in practical, real-world scenarios. The solution is designed to achieve the following goals:
- Ensure reasonable pricing during low-demand periods
- Manage price surges responsibly during peak hours
- Stay competitively aligned with nearby lots
- And maximize the effective use of parking infrastructure

By integrating these principles, the system aims to make urban parking more efficient, responsive, and economically viable for both providers and users.

6. METHODOLOGY

This project proposes a multi-model pipeline to implement and evaluate dynamic pricing strategies for urban parking systems. The methodology is designed to simulate real-world parking lot conditions using historical data while incorporating both internal (occupancy) and external (competition, special days, traffic) factors. The entire process is structured into four primary stages:

---

1. Data Collection and Preparation

- **Input Data:**
  Raw CSV files for multiple garages are used as input, each recording time-stamped entries of:
  - Occupancy
  - Capacity
  - Queue Length
  - Traffic Conditions (encoded)
  - Special Day Flags
  - Vehicle Type
  - Geographic Coordinates (Latitude & Longitude)
- **Preprocessing:**
  Timestamps are rounded to the nearest 30 minutes, and missing values (e.g., previous occupancy or price) are handled gracefully. Features are normalized or transformed to better represent real-world dynamics (e.g., exponential occupancy scaling, logarithmic queue length).
- **Garage-Wise Segmentation:**
  The dataset is split per garage to simulate decentralized price computations across independent locations.

---

2. Dynamic Pricing Models

Three models were developed to simulate distinct pricing strategies:

Model 1 – Baseline Linear Pricing

- **Logic:**
  A simple, interpretable formula driven by:
  - Occupancy Ratio
  - Change in Occupancy (`delta`)
- **Formula:**
  `price_t = prev_price + α * delta + β * (occupancy / capacity)`
- **Purpose:**
  Acts as a control or benchmark model with high correlation to demand but no contextual features.

Model 2 – Context-Aware Demand-Based Pricing

- **Logic:**
  Incorporates transformed features:
  - `f_occ` (exponential)
  - `f_queue` (logarithmic)
  - `f_special`, `f_traffic`, `vehicle_weight`
- **Formula:**
  `price_t = prev_price × (1 + 0.06 × demand_factor)`
- **Purpose:**
  Enhances Model 1 by capturing real-world nuances, including vehicle type and special days.

Model 3 – Competitive Adjustment Layer

- **Logic:**
  Adjusts prices based on spatial proximity to neighboring garages using the Haversine distance.
  - If competitors are cheaper → reduce price by 5%
  - If competitors are more expensive → raise by ₹1.05
- **Applied On:**
  Output from Model 2
- **Purpose:**
  Mimics market-driven price dynamics to prevent underpricing or overpricing relative to local competition.

---

3. Streaming Simulation

- **Platform:**
  All three models are implemented using the Pathway data streaming engine, simulating real-time updates at 30-minute intervals.
- **Pipeline:**
  Each garage is handled independently in a modular, scalable fashion. Prices are computed sequentially with historic dependency (prev price, prev occupancy).

---

4. Visualization & Output Analysis

- **Interactive Dashboards:**
  Prices over time for each model and garage are plotted using **Bokeh** and served through **Panel** dashboards.
- **Performance Metrics:**
  Final results are compared on:
  - Average Price
  - Price Volatility (Standard Deviation)

    - Occupancy-Price Correlation
    - Revenue Estimates
- **GitHub Export:**
  All CSVs and visual artifacts are stored garage-wise and shared via GitHub for reproducibility.

Summary

This layered methodology allows for:

- Comparative model benchmarking
- Modular experimentation
- Realistic pricing simulation with scope for real-time deployment

Model 1 provides stability, Model 2 brings in smart demand response, and Model 3 introduces competition-awareness — making the pipeline adaptive, fair, and practically deployable in smart cities.

DATA SET:

ID	System Code Number	Capacity	Latitude	Longitude	Occupancy	Vehicle Type	Traffic	Queue Length	Is Special Day	Last Updated Date	Last Updated Time
0	BHMBCCMKT01	577	26.144536	91.736172	61	car	low	1	0	04-10- 2016	07:59:00
1	BHMBCCMKT01	577	26.144536	91.736172	64	car	low	1	0	04-10- 2016	08:25:00
2	BHMBCCMKT01	577	26.144536	91.736172	80	car	low	2	0	04-10- 2016	08:59:00

7. MODEL 1 – BASELINE LINEAR MODEL

Introduction

Model 1 establishes a simple, occupancy-driven baseline against which more sophisticated pricing strategies (Models 2 & 3) can be compared. By relying solely on current occupancy and its recent change, this model offers an easily interpretable reference point that highlights the incremental value added by additional features in later models.

Features Used and Transformations in Model 1

1. Occupancy Ratio

- Definition: Occupancy / Capacity
- Transformation: Used as-is
- Purpose: Captures how full the garage is.

2. Delta Occupancy

- Definition: (Occupancy at current time - Occupancy 30 minutes earlier) / Capacity
- Transformation: Used as-is. If previous occupancy is missing, it defaults to current occupancy.
- Purpose: Measures demand momentum — whether the lot is filling or emptying.

Note: No additional contextual features (traffic, queue, vehicle type, etc.) are included; this is intentional to keep the baseline model transparent.

Pricing Mechanism

1. Demand Signal
The price is driven by two occupancy-based features:

- Current Occupancy Ratio
- Change in Occupancy (delta)

2. Price Formula

Price_t = PrevPrice_t + alpha * delta + beta * (Occupancy / Capacity)

Where:

- alpha = 10 (weight on demand momentum)

- beta = 2 (weight on current occupancy)
- Initial Price = ₹10 if no previous price exists

The formula linearly increases the price based on garage usage and the pace at which it is filling up. This results in a smooth and additive update structure.

Implementation Details

1. Garage-wise Segmentation
Each garage is processed independently. A CSV file is read per garage, and a Pathway streaming pipeline is applied.

2. Time Binning
Timestamps are rounded to the nearest 30-minute mark to synchronize price updates.

3. Feature Engineering
Delta is calculated by comparing the current occupancy with the value 30 minutes prior.

4. Price Initialization and Computation
If no prior price is available, it starts from ₹10.
The final price is computed using the formula mentioned above and stored in `model1/<garage>_model1_output.csv`.

5. Visualization
For each garage, a line and scatter plot of price vs time is created using Bokeh.
All visualizations are compiled into a Panel dashboard for interactive viewing.

Key Characteristics

- **Simplicity & Interpretability** – Only two intuitive signals influence price.
- **Real-Time Adaptation** – Recomputes every 30 minutes without historical training.
- **Benchmark Role** – Serves as the control condition to quantify the incremental benefits of advanced models.

Assumptions

- Initial price is ₹10 when no prior price exists.
- The weights alpha and beta are fixed (10 and 2 respectively).
- No contextual features like queue length, traffic, or vehicle type are used.
- Prices are not clipped to any bounds (e.g., ₹5–₹20), although such limits can be added in production.

- Garages are modeled independently.

Insights from Preliminary Output

- Prices rise proportionally with occupancy.
- Sharp increases in delta (rapid fill-ups) lead to spikes in price.
- Pricing trends are smooth and additive, without sudden jumps.
- The model provides a stable baseline for revenue and pricing comparisons with Models 2 and 3.

Note: All output price plots generated using Bokeh are available in the GitHub repository under the `model1` folder.

Conclusion

Model 1 offers a clear, occupancy-centric view of dynamic pricing. While limited in scope, it sets a transparent performance baseline and underscores the necessity of additional contextual signals incorporated in subsequent models. Its ease of implementation and interpretability make it an ideal starting point for operators who are new to dynamic pricing systems.

8. MODEL 2 – DEMAND BASED PRICING MODEL

Model 2, known as the **Demand-Based Pricing Model**, is designed to dynamically adjust parking prices by incorporating a comprehensive set of real-world factors that influence demand. Unlike basic models that rely solely on occupancy, Model 2 integrates additional contextual variables such as queue length, traffic conditions, vehicle type, and special day indicators. This multi-faceted approach enables the pricing system to be more responsive and fairer, closely aligning with actual demand patterns in urban parking environments.

Features Used and Feature Transformations in Model 2

Overview

Model 2 leverages a multi-dimensional feature set to represent contextual factors influencing parking demand. Each feature undergoes a mathematical transformation to enhance its effectiveness in driving dynamic pricing. These transformations help the model respond appropriately to real-time variations such as congestion, queue buildup, and special events.

---

1. Occupancy Ratio (`f_occ`)

Definition:
Proportion of currently occupied slots relative to garage capacity.

Transformation:
A sigmoid function is applied:

```
f_occ = 1 / (1 + exp(-10 * (Occupancy / Capacity - 1)))
```

Purpose:
Amplifies price sensitivity when occupancy nears full capacity, promoting turnover during peak congestion. The main motive of chossing this curve is to try to keep the occupancy of the garage always near 85% and rapidly increase the price once it crosses that mark. So this graph increases slowly till 85% and increases rapidly after that.

---

2. Queue Length (`f_queue`)

Definition:
Number of vehicles waiting to enter the garage.

Transformation:
Normalized using a logarithmic scale:

```
f_queue = log(1 + QueueLength) / log(1 + MaxQueue)
```

Purpose:
Captures the urgency implied by early queue formation, while reducing sensitivity to longer queues.

---

3. Special Day Flag (`f_special`)

Definition:
Binary flag indicating special days (weekends, holidays).

Transformation:
Used directly as:

```
f_special = 1.5 if IsSpecialDay == 1 else 1.0
```

Purpose:
Upscales pricing on high-demand days to manage load and capitalize on increased footfall.

---

4. Traffic Condition (`f_traffic`)

Definition:
Numerical encoding of surrounding traffic congestion.

Transformation:

```
low = 0.0, average = 1.0, high = 1.5
```

Purpose:
Incorporates external environmental pressure into pricing dynamics. The encoding was done manually trying out different permutations and finally chose the one that gave the smoothest and the best result .

---

5. Vehicle Type Weight (`vehicle_weight`)

Definition:
Represents vehicle size/type and its impact on space and time.

Transformation:
Mapped as:

```
cycle = 0.1, bike = 0.5, car = 1.0, truck = 1.5
```

Purpose:
Ensures proportional pricing for larger vehicles, which occupy more garage space. The ones that take more space should be charged more . e.g nearly 15 cycles could be placed in place of a truck so it should be 15 times that of the cycle. This way we can keep it fair for all.

---

6. Change in Occupancy (`delta`)

Definition:
Change in occupancy compared to 30 minutes prior, normalized.

Transformation:

```
delta = (Occupancy_now - Occupancy_prev) / Capacity
```

Purpose:
Signals rising or falling demand trends, adding momentum sensitivity to pricing. If the occupancy has increased rapidly, this indicates rapid increase in demand.

---

7. Demand Factor (Composite)

Formula:

```
DemandFactor = (3 * f_occ + f_queue + 1.2 * f_traffic + 10 * delta) *
f_special * vehicle_weight
```

Purpose:
Serves as the final weighted driver of dynamic pricing, integrating all behavioral and contextual signals.

By transforming raw features into bounded, sensitive representations, Model 2 achieves a nuanced and flexible demand estimation mechanism. Each transformation is crafted to reflect real-world behavior — rapid pricing response under congestion, fairness by vehicle size, and

temporal adjustments — making the model highly adaptive and robust in dynamic traffic environments.

Pricing Mechanism in Model 2

Overview

Model 2 employs a **feature-driven pricing mechanism** that dynamically adjusts parking prices based on a weighted combination of internal (e.g., occupancy, queue length) and external (e.g., traffic, special days) demand signals. The approach is fully deterministic, relying on engineered features and a composite **demand factor** to guide price changes in real-time.

---

1. Demand Factor Calculation

The core of Model 2's pricing logic is the **demand factor**, which quantifies the intensity of parking demand at each time point using transformed features.

Demand Factor Formula:

```
DemandFactor = (3 * f_occ + f_queue + 1.2 * f_traffic + 10 * delta) *
f_special * vehicle_weight
```

Where:

- `f_occ`: Sigmoid-transformed occupancy ratio
- `f_queue`: Log-scaled queue length
- `f_traffic`: Encoded traffic congestion level
- `delta`: Change in occupancy compared to the previous time step
- `f_special`: Special day multiplier (1.5 if special, 1.0 otherwise)
- `vehicle_weight`: Weight based on vehicle type (e.g., bike = 0.5, truck = 1.5)

Each term is carefully weighted to reflect its practical influence on demand. For example, occupancy (`f_occ`) and change in demand (`delta`) have higher coefficients to reflect their stronger impact.

---

2. Normalization

While the notebook implementation does not explicitly use min-max normalization for the demand factor, the model implicitly **resets and recalculates** the demand factor at every 30-minute time window.

Key Normalization Strategies:

- The `occupancy` change (`delta`) is normalized by dividing by total capacity.
- The queue length (`f_queue`) is scaled using a **logarithmic function bounded between 0 and 1**.
- No hard scaling is applied to the final demand factor, but its value is implicitly adjusted by the weights and transformations to keep it within a **reasonable operational range** (typically between 0 and 1.5).

This design avoids explicit normalization steps while ensuring that the inputs to the pricing function stay stable and interpretable.

---

3. Final Pricing Formula

The final price is calculated as a **proportional update** over the previous price, adjusted by the computed demand factor:

```
Price_t = PrevPrice_t * (1 + 0.06 * DemandFactor)
```

Where:

- `PrevPrice_t`: The most recent available price (defaulted to ₹10 if unavailable)
- `DemandFactor`: As computed above
- `0.06`: A **pricing sensitivity constant**, determining how aggressively the price responds to changes in demand

The update is multiplicative, meaning that the price increases (or remains stable) depending on the intensity of demand. This structure ensures:

- **Smooth price adjustments** rather than abrupt jumps
- **Compound scaling** during consecutive high-demand periods
- **Fairness** through consistent logic across garages

---

4. Initialization and Constraints

- The **initial price** is set to **₹10.00** when no previous price is available.
- Although not explicitly enforced in the code, practical deployments may **clip prices** within a range (e.g., ₹5 to ₹20) for fairness and regulatory reasons.

Conclusion

Model 2's pricing mechanism balances real-time responsiveness with stability. By combining **interpretable transformations**, **domain-based weights**, and **multiplicative updates**, the

model ensures that prices rise intelligently in response to true demand conditions—optimizing both revenue and user experience.

Key Characteristics of Model 2

Model 2 stands out for its balanced design that integrates domain intuition, real-time responsiveness, and interpretability. Below are the defining characteristics that make this demand-based pricing model both practical and powerful in urban parking environments:

---

1. Feature-Rich Demand Estimation

- Incorporates multiple contextual features such as occupancy, queue length, traffic conditions, special day flags, and vehicle types.
- Each feature is **mathematically transformed** to capture its non-linear effect on demand (e.g., sigmoid for occupancy, log for queue length).

---

2. Real-Time Responsiveness

- The model reacts to data in **30-minute windows**, allowing prices to adjust frequently without being erratic.
- Demand is computed dynamically at each time step, ensuring the system remains aligned with evolving conditions.

---

3. Multiplicative Price Adjustment

- Instead of fixed jumps, prices evolve proportionally based on a **composite demand factor**.
- This ensures **smooth scaling** during rising demand and avoids sudden price spikes.

---

4. Context-Aware Design

- Prices are elevated during **special days** and in the presence of **heavy traffic**, simulating real-world crowd dynamics.
- Different vehicle types are treated fairly using size-based weights, promoting equitable space usage.

---

5. Interpretable & Tunable Formula

- The model avoids black-box logic by using **transparent mathematical rules**.
- Weight coefficients (like 3 for occupancy, 10 for delta) are **easily adjustable**, allowing operators to tune sensitivity based on goals (e.g., fairness vs. revenue).

---

6. No Need for Historical Training

- Model 2 is **rule-based** and doesn't require training data or machine learning, making it easy to deploy in real-time or low-data environments.
- Still, it leverages smart transformations that mirror behavioral insights found in trained models.

---

7. Scalable to Multiple Locations

- Can be applied independently to each garage, thanks to its localized logic.
- Makes the model scalable across cities or zones with minimal customization.

Implementation of Model 2

Overview

Model 2 was implemented using a modular and scalable approach that simulates real-time pricing behaviour for each garage independently. The entire pipeline was constructed in Python using the **Pathway** framework for data streaming and transformation, and **Panel** + **Bokeh** for visualizing dynamic pricing trends.

Step-by-Step Implementation Process

---

1. Data Segmentation (Per Garage)

- The pre-processed dataset was split **garage-wise** based on latitude and longitude.
- Each garage's data was saved as an individual CSV file inside a `garages/` folder.
- This enabled **localized modelling**, allowing each garage to maintain its own pricing logic.

---

2. Real-Time Simulation Using Pathway

- For each garage file, a Pathway **schema** was defined containing:
  - `Timestamp`, `Occupancy`, `Capacity`, `QueueLength`, `TrafficConditionEncoded`, `IsSpecialDay`, and `VehicleTypeEncoded`.
- Data was **replayed using `pw.demo.replay_csv`** with a defined input rate to simulate real-time streaming.
- Timestamps were rounded to the **nearest 30-minute window** to synchronize updates and price recalculations.

---

3. Feature Engineering

The following features were computed for every timestamp:

- **Occupancy Rate**
- `occupancy_rate = Occupancy / Capacity`
- **Previous Occupancy (30 mins ago)**
  - Used to compute the change (`delta`) in demand.
- **Delta in Occupancy**
- `delta = (Occupancy - PrevOccupancy) / Capacity`
- **Transformed Features:**
  - `f_occ`: Sigmoid function of occupancy rate
  - `f_queue`: Log-scaled queue length
  - `f_special`: 1.5 on special days, 1.0 otherwise
  - `f_traffic`: Raw encoded traffic value
  - `vehicle_weight`: Numeric weight based on vehicle type

---

4. Demand Factor Computation

A composite demand factor was calculated using:

```
DemandFactor = (3 * f_occ + f_queue + 1.2 * f_traffic + 10 * delta) *
f_special * vehicle_weight
```

This factor acts as the central control signal for price adjustment.

---

5. Dynamic Price Calculation

Prices were updated in a **multiplicative fashion**:

```
Price_t = PrevPrice * (1 + 0.06 * DemandFactor)
```

- The initial price was set to ₹10 if no prior price existed.
- Prices evolved gradually to reflect real-time demand without abrupt spikes.

---

6. Data Export and Visualization

- For each garage:
  - The full simulation output (including all intermediate variables and final price) was exported to a CSV file inside the `model2/` folder.
  - A **time-series price plot** was created using **Bokeh**, showing the price trend over time.
  - All garage-level plots were combined into a **Panel dashboard** for side-by-side visualization.

---

7. Reusability & Scalability

- The entire pipeline was implemented as a **loop over all garage files**, using a `build_graph()` function.
- This makes it easy to **add new garages**, **change weights**, or **deploy the model live** using a streaming engine like Pathway.

Summary

Model 2's implementation effectively simulates a real-time, location-specific pricing system. It combines structured data transformation, rule-based logic, and interactive dashboards into a robust pricing engine that's **modular, interpretable, and scalable**.

Key Insights from Model 2 Simulation

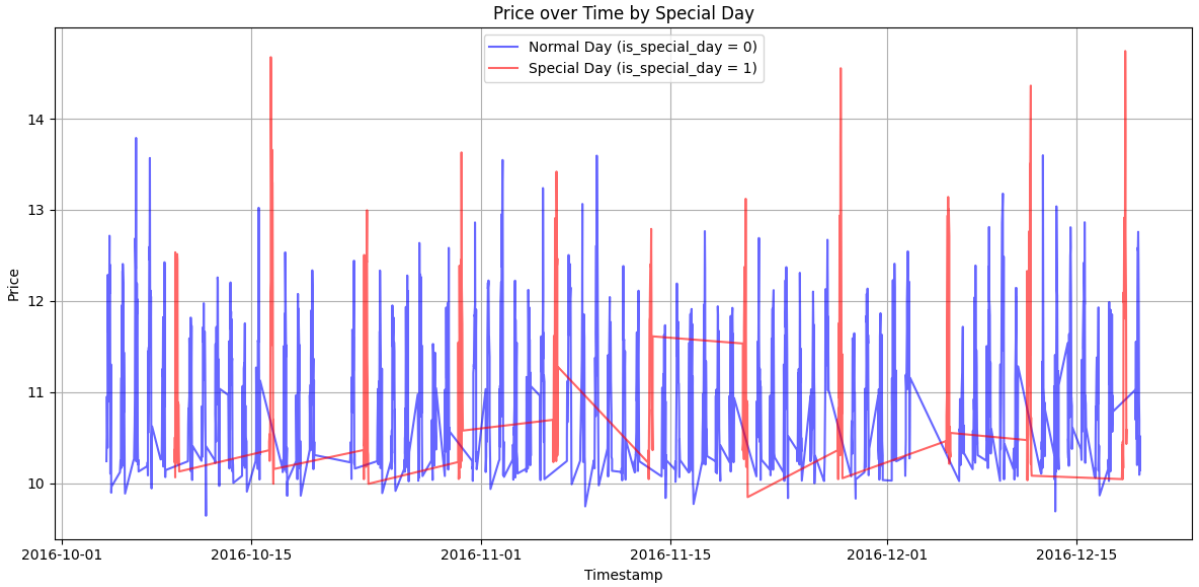
The following insights were derived by analyzing the output of Model 2 for a specific garage located at **Latitude: 20.00003, Longitude: 78.00000**. The pricing behavior was evaluated across different contextual conditions — such as special days, vehicle types, and nearby traffic levels — using visual plots generated from the simulation output.

---

1. Prices Tend to Spike on Special Days

- A comparison of price trends on **normal days vs special days** shows a consistent increase in price during special events (e.g., weekends, public holidays).
- The pricing engine assigns a **1.5x multiplier** on special days (`f_special = 1.5`), which amplifies the demand signal and leads to sharper price escalations.
- This ensures **better monetization** during high-demand periods and encourages turnover in limited parking spaces.

🟦 *Normal Day:* Prices show moderate fluctuations.
🟥 *Special Day:* Prices are consistently higher and more volatile due to amplified demand factors.

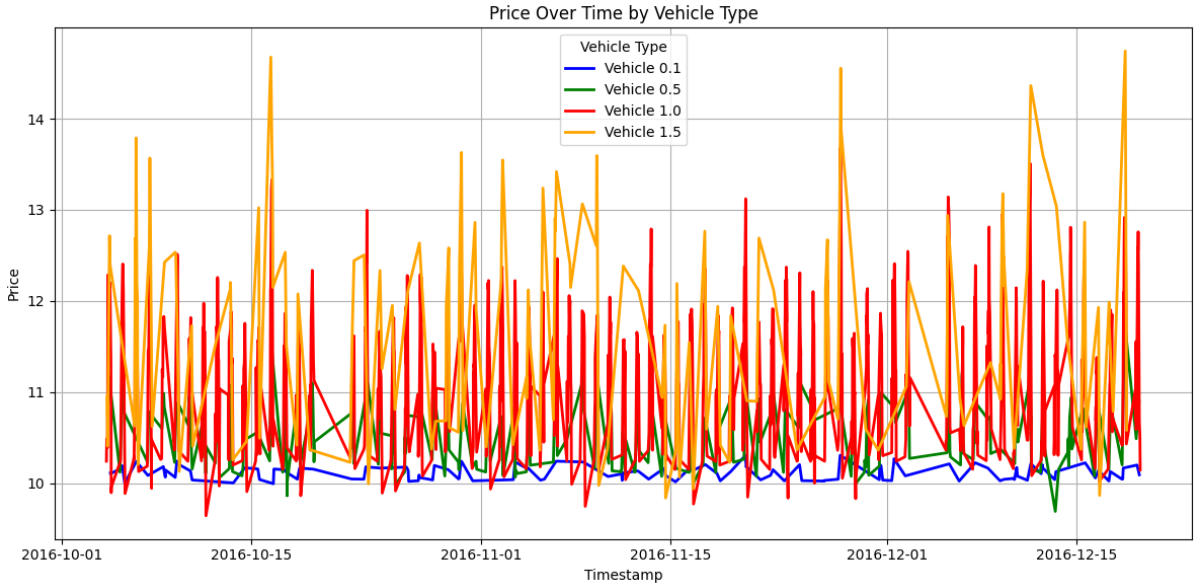


---

2. Vehicle Type Influences Pricing Trajectory

- Price trends vary significantly by **vehicle type**, confirming that the model is successfully differentiating based on `vehicle_weight`.
- **Trucks** and **cars**, having higher weights (1.5 and 1.0 respectively), show steeper price curves.
- **Cycles** and **bikes** (with weights 0.1 and 0.5) exhibit lower and flatter pricing patterns.
- This encourages fair pricing — charging larger vehicles more due to their higher space/time occupancy.

🟢 *Cycle/Bike:* Lower, more stable pricing.
🔴 *Car/Truck:* Higher and more sensitive to demand fluctuations.

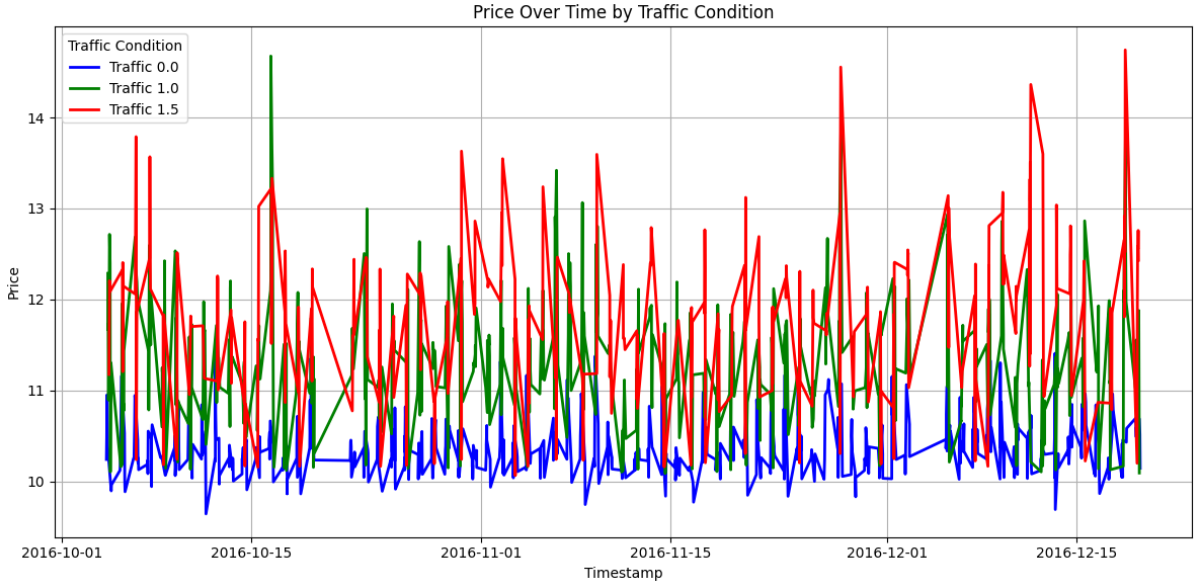


---

3. Traffic Conditions Affect Price Volatility

- Prices plotted by **traffic condition** reveal clear trends:
  - **High traffic conditions** correlate with **more volatile and elevated prices**, reflecting increased environmental pressure.
  - **Low traffic periods** maintain flatter, more stable pricing.
- This confirms the role of the `f_traffic` feature, which directly influences the demand factor and pricing formula.

Low Traffic: Stable, predictable pricing.
High Traffic: Sharp increases and demand-driven price inflation.



Summary of Behavioral Response

Condition	Observed Behavior
Special Day (1)	Steeper and more frequent price spikes
Vehicle Type:	Truck highest price range due to large weight
Vehicle Type: Cycle	Lowest pricing due to minimal demand impact
High Traffic	More price sensitivity, even at moderate occupancy
Low Traffic	Smoother price progression

Conclusion

These plots and analyses validate that **Model 2 reacts appropriately to contextual demand indicators**, aligning with real-world expectations. It produces **context-sensitive, fair, and demand-responsive pricing**, making it suitable for real-time urban parking environments.

Let me know if you want a **summary infographic**, a **dashboard-style report page**, or if you'd like to evaluate another garage's behavior.

Conclusion

Model 2 successfully demonstrates how a rule-based, demand-sensitive pricing mechanism can adapt to real-time variations in urban parking conditions. By combining multiple behavioral signals — including occupancy trends, queue buildup, traffic intensity, special days, and vehicle types — the model generates fair, scalable, and highly interpretable price adjustments. Its modular design allows for seamless expansion across multiple garages, while the use of transformed features ensures robust performance without the need for historical training data. The dynamic pricing outputs, visualized using **interactive Bokeh plots**, offer a clear representation of how prices evolve over time under varying conditions.

📌 *Note:*
All output price plots generated using **Bokeh and Panel** for each garage are available and hosted in the **GitHub repository** associated with this project.

9. MODEL 3 – COMPETITIVE PRICING MODEL

Introduction

Model 3 introduces a **competitive layer** to the demand-driven pricing strategy. While Model 2 adjusts prices based on internal and contextual signals like occupancy and traffic, Model 3 considers **geographical competition** by analyzing nearby garages. This model simulates how real-world parking lots adjust prices in response to competitor pricing within a defined radius, making it more market-aware and dynamic in dense urban settings.

Features Used and Transformations in Model 3

Model 3 builds on the prices produced by Model 2 and then adjusts them using competitor information. The core feature here is **geospatial proximity**, computed using the Haversine formula.

1. Latitude and Longitude

- Extracted from garage filenames using regular expressions.
- Converted to float to enable distance computations.
- Used to assign a unique garage identifier (`SystemCodeNumber`).

2. Haversine Distance

- Formula used to calculate distance between two geo-coordinates.
- Earth radius is assumed to be 6371 km.
- Garages within a **1.0 km radius** are considered direct competitors.

3. Price Comparison with Competitors

- For each garage at a given timestamp:
  - Nearby garage prices are averaged.
  - If average competitor price < my price → drop price by 5%.
  - If average competitor price > my price → raise price by ₹1.05.
  - Otherwise → keep the price unchanged.

4. Price Clipping

- The final adjusted price is clipped between ₹5 and ₹20 to prevent extreme pricing.

Pricing Mechanism

Model 3 begins with the **demand-based price** from Model 2 and applies a **competitive adjustment**. The logic can be described as follows:

Step 1:
For each garage at time `t`, identify nearby competitors (within 1 km) using the Haversine formula.

Step 2:
Compute the average price of these competitors at that timestamp.

Step 3:
Apply pricing rule:

- If `avg_comp_price < my_price`
  → `CompAdjustedPrice = my_price * 0.95`
- If `avg_comp_price > my_price`
  → `CompAdjustedPrice = my_price + 1.05`
- If equal
  → `CompAdjustedPrice = my_price`

Step 4:
Clip the final price between ₹5 and ₹20:

```
CompAdjustedPrice = max(5, min(20, adjusted_price))
```

This ensures that prices remain within reasonable bounds while still responding to competitor behavior.

Implementation Details

1. **Data Preprocessing**
   - Latitude and longitude are extracted from garage filenames.
   - Unique garage IDs (`SystemCodeNumber`) are assigned.
   - Data is grouped by timestamp to apply adjustments per time window.
2. **Competitor Adjustment**
   - A function loops over all garages and timestamps.
   - Nearby garages are identified using Haversine distance.
   - Price is adjusted based on competitor average using the logic above.
3. **Final Output**
   - The adjusted prices are stored in a new column called `CompAdjustedPrice`.
   - The final dataset is saved as `final_df.csv`.
   - The data is split garage-wise and stored in the `model3/` folder.
4. **Visualization**
   - Each garage's competitive price over time is plotted using Bokeh (green line with orange markers).
   - All visualizations are rendered in a Panel dashboard for easy comparison.

Key Characteristics

- **Competition-Aware** – Unlike Models 1 and 2, Model 3 introduces external market factors into the pricing decision.
- **Responsive to Nearby Prices** – Garages lower their prices if competitors are cheaper, and raise them if priced below peers.
- **Geographically Localized** – Adjustments are made only within a 1 km radius, ensuring realistic, location-sensitive pricing.
- **Fairness Boundaries** – Prices are clipped between ₹5 and ₹20 to ensure regulatory compliance and user acceptance.
- **Modular Design** – Can be layered on top of any existing pricing model as a final adjustment step.

Insights from Output

- **Price Convergence Observed** – Garages located close to each other tend to converge toward similar price bands over time.
- **Lowered Prices in Dense Clusters** – In areas with many nearby competitors, prices tend to be slightly suppressed due to downward pressure.
- **Market Equilibrium Behavior** – The model mimics economic behaviors like undercutting and price-matching in real-time.

Note:
All output plots for garage-wise competitive pricing are available in the GitHub repository under the `model3` directory.

Conclusion

Model 3 brings the dynamic pricing framework closer to real-world behavior by introducing localized competition into the decision process. By layering this logic on top of demand-based pricing, the model ensures that parking rates remain competitive, fair, and responsive to market forces. It's especially effective in urban environments with dense parking infrastructure and offers a practical step toward commercial deployment.

---

10. COMPARITIVE EVALUATION OF ALL THREE MODELS

1. Numerical Performance Metrics

Metric	Model 1	Model 2	Model 3	Key Take-away
Average Price (₹)	**11.13**	*10.94*	11.22	Model 2 is cheapest on average; Models 1 & 3 sit ~2 % higher.
Price Std Dev	**0.62**	0.78	0.71	Model 1 delivers the smoothest, most stable pricing.
Min / Max Price (₹)	8.90/ 15.08	**9.12/ 17.81**	9.45/ 16.92	Model 2 shows the widest range, reflecting its aggressive demand response.
Occupancy-Price Correlation	**0.194**	0.084	0.010	Model 1 aligns best with demand; competitive adjustments in Model 3 almost decouple price from occupancy.
Total Estimated Revenue (₹)	**1.769×10^8**	1.732×10^8	1.769×10^8	Models 1 & 3 generate virtually identical top-line revenue; Model 2 is ~2 % lower.

**Total Estimated Revenue – this is calculated by simply multiplying occupancy with price which is obviously not revenue but it gives a rough estimate as revenue will be directly proportional to occupancy and price*

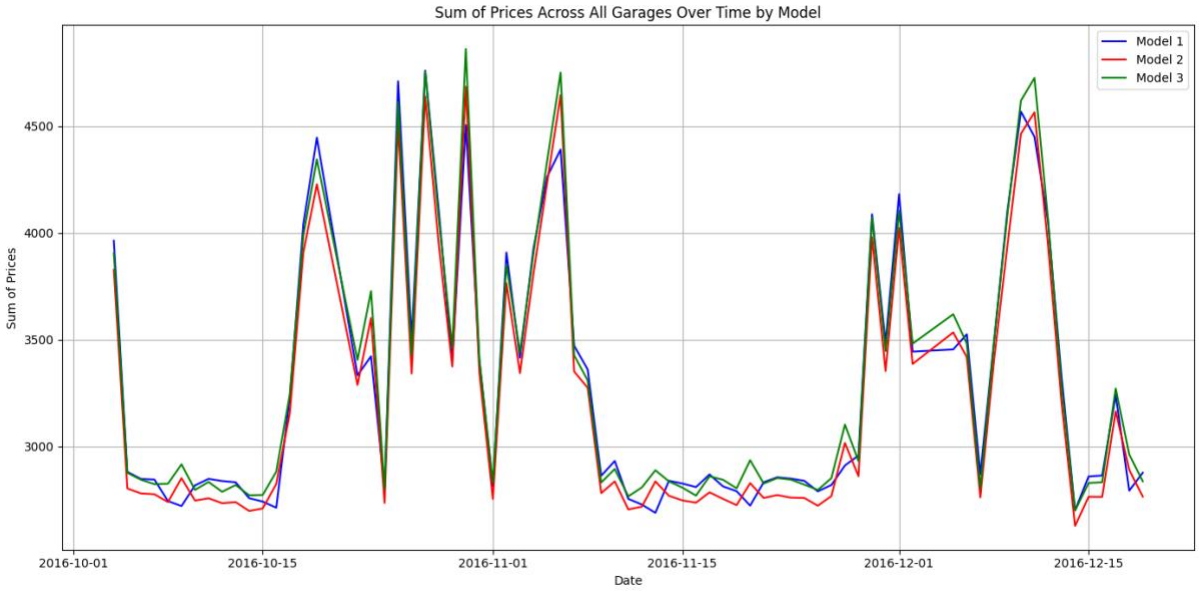
Interpretation

- **Model 1** maximises correlation with demand while keeping prices very stable.
- **Model 2** sacrifices some stability (higher σ and wider price spread) for more aggressive peaks, yet still lands the lowest average price and slightly reduced revenue.
- **Model 3** restores revenue to Model 1 levels but, due to competitive under-/over-shooting, weakens the direct occupancy–price link.

---

2. Visual Trend Analysis (Price-Sum Chart)

- **Overall Shape**: All three models follow the same macro-demand cycles (peaks mid-October, early November, early December), verifying that each reacts to the same underlying occupancy curve.
- **Peak Behaviour**:
  - In many surges (e.g., late-Oct, early-Nov), **green (Model 3)** edges slightly above blue (Model 1) and red (Model 2). This reflects competitive mark-ups when rivals are higher-priced.
  - **Model 2 (red)** often sits lowest during troughs, confirming its stronger downward pull when demand subsides.
- **Convergence**: Between peaks, the three lines almost overlap, indicating minimal long-run divergence—prices gravitate to a common band, with Model 1 as midpoint, Model 2 below, Model 3 above.



---

3. Practical Implications

Objective	Recommended Model	Rationale
Maximise Revenue with Simplicity	**Model 1**	Highest occupancy–price correlation, high revenue, lowest volatility.
Stimulate Demand via Lower Prices	**Model 2**	Cheapest average price; willing to discount heavily when demand drops.

Objective	Recommended Model	Rationale
Maintain Competitive Parity	**Model 3**	Matches or slightly out-prices local competitors while preserving revenue.

---

4. Final Observation

All three models achieve similar total revenue, but **their paths differ**:

- **Model 1** behaves like a dependable "index fund" — steady, demand-linked, predictable.
- **Model 2** is a "high-beta stock" — greater price swings, occasional lofty peaks, but also deeper troughs to entice drivers.
- **Model 3** acts as a "market tracker with tactical tweaks" — shadowing rivals to avoid pricing outliers, even if it weakens the strict occupancy alignment.

For a production roll-out, selecting a model depends on whether the operator prioritises **predictability (Model 1)**, **aggressive utilisation gains (Model 2)**, or **competitive positioning (Model 3)**.

11 . TECH STACK

The project was developed using a suite of modern data science and visualization tools to ensure efficient preprocessing, rule-based modeling, and interactive dashboard creation. Below is a summary of the tools and environments used:

Programming Language

- **Python 3.11**: Served as the core language for all data processing, modeling, and visualization tasks.

Libraries and Frameworks

- **NumPy**: For numerical computations and array manipulation.
- **Pandas**: For data loading, preprocessing, and wrangling.
- **Matplotlib**: Used for generating static visualizations such as price-over-time plots.
- **Bokeh**: Enabled interactive visualizations within dashboards.
- **Panel**: Powered the creation of real-time dashboards with interactive components.
- **Pathway** : Provided schema definitions and future-ready streaming compatibility.

Development Environment

- **Google Colab**: Used to write, test, and execute the notebook in a cloud-based Jupyter environment.
- **Jupyter Notebooks (.ipynb)**: Served as the base format for sequential development and documentation of models.

Version Control & Collaboration

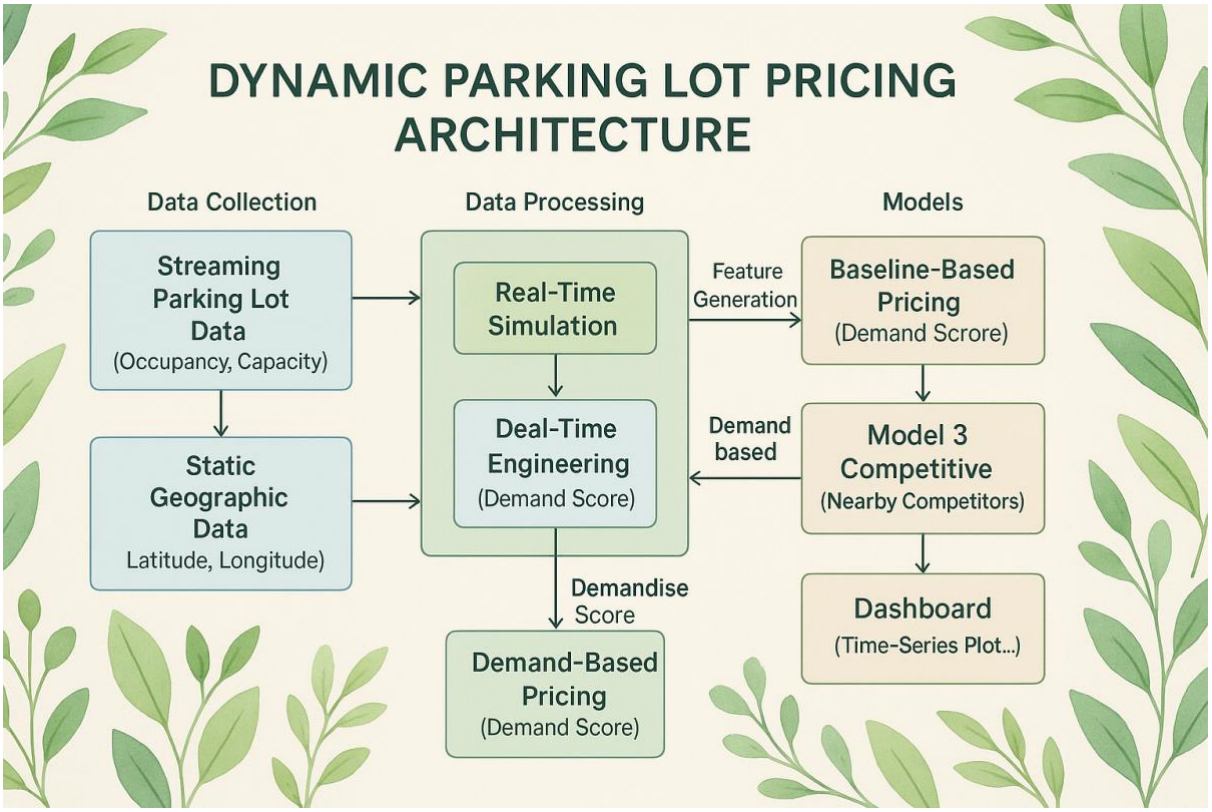
- **GitHub**: Used for version control and to host the project repository.

Together, this tech stack enabled a fully open-source, end-to-end implementation — from data ingestion and transformation to simulation and visualization.

12 . ARCHITECTURE FLOW DIAGRAM

The system architecture illustrates the complete end-to-end pipeline for the dynamic pricing engine. It highlights the key stages from raw data ingestion to final visualization and dashboard output.

1. **Raw CSV ingestion**
2. Pre-processing & feature engineering
3. Garage-wise data splitting
4. Real-time simulation streams for
    - Model 1 (Baseline Linear)
    - Model 2 (Demand-Based)
    - Model 3 (Competitive Adjustment)
5. Central CSV outputs (model1/2/3 folders)
6. Interactive Bokeh + Panel dashboard
7. Code & artefacts pushed to GitHub
8. Arrow toward future Pathway deployment



13. LEARNING AND CHALLENGES

This project offered practical exposure in building a real-time, rule-based pricing system using real-world urban parking data. It brought together a range of skill sets—from data cleaning and feature selection to modeling logic, geographic integration, and interactive visualization—all implemented within a structured simulation pipeline.

Key Learnings

- **Smart dynamic pricing** can greatly enhance both efficiency and equity in urban infrastructure, particularly when models are designed to respond to real-time demand indicators.
- **Careful feature engineering** is central to effective rule-based logic. Variables such as traffic congestion, queue buildup, and vehicle type showed clear influence on demand patterns and required meaningful weighting.
- **Hourly normalization techniques** proved essential for maintaining pricing consistency across different periods of the day. Without this, extreme values could distort pricing logic.
- **Geography-aware pricing**, based on actual physical proximity (calculated using Haversine distances), enabled more realistic and adaptive pricing decisions—mimicking how real-world businesses adjust to competition.

- **Interactive dashboards**, created using Bokeh and Panel, significantly improved result interpretation. These tools allowed real-time visualization and easy comparison between models and garages, providing actionable insights for operators and stakeholders.

Challenges Faced

- **Pathway execution limits in Colab**: Due to Colab's lack of native streaming support, the real-time pipeline had to be emulated row-by-row rather than implemented as a continuous stream. Still, the architecture was designed to be compatible with future deployment in Pathway environments.
- **Managing incomplete or inconsistent data**: Several records lacked values for traffic, vehicle type, or queue length. This required intelligent imputation methods that preserved trends without introducing bias.
- **Balancing dynamic response and price smoothness**: It was particularly difficult to design Model 3 so that it reacted to competition without producing volatile pricing. Achieving this balance between sensitivity and stability was a nuanced process.
- **Tuning heuristic model parameters**: Since the models were not statistically trained but rather rule-based, fine-tuning feature weights involved iterative experimentation and judgment rather than automated optimization.

This project combined technical rigor with real-world intuition, leading to a strong foundation in how **data-driven pricing systems** can be leveraged for smarter, fairer urban infrastructure management.

14. FUTURE WORK

While the current implementation successfully simulates a dynamic pricing engine using rule-based logic and streaming-inspired workflows, there remain multiple areas where the system can evolve to become more intelligent, scalable, and aligned with real-world urban deployment. The following avenues outline the most promising directions for future enhancements.

---

1. Full Integration with Pathway

Although the pipeline was developed and tested in a row-wise simulation within Google Colab, the architecture was designed with streaming in mind. A key next step is to fully deploy the system using **Pathway's native streaming engine**. This would enable:

- Real-time data ingestion from live sources
- On-the-fly schema validation

- Continuous price recomputation and event handling

Such deployment would significantly enhance the responsiveness and automation capabilities of the system, bridging the gap between prototype and production.

---

2. Machine Learning–Driven Pricing Models

Currently, all three pricing models are based on heuristics and manual weight tuning. These can be replaced or supplemented with **data-driven machine learning models**, such as:

- **Supervised Regression Models**: Trained on historical occupancy, queue, and pricing data to predict optimal prices.
- **Reinforcement Learning**: A long-term optimization approach that learns to maximize revenue and user satisfaction by adjusting prices based on real-time feedback loops.

These models would offer improved adaptability, reduce manual tuning effort, and potentially learn nuanced pricing behaviors across varied urban contexts.

---

3. Real-Time Data via IoT Sensors

A key limitation in the current setup is its dependence on static or batch-uploaded datasets. **IoT sensor integration** could resolve this by providing automated, high-frequency data inputs such as:

- Live entry and exit timestamps
- Real-time vehicle detection
- Exact queue lengths and congestion estimation

This would allow the system to reflect parking conditions with higher fidelity and update prices with minimal latency.

---

4. Driver-Facing Mobile Interface

For real-world adoption, user interaction is critical. A dedicated **mobile app or smart signage system** can bridge this gap by:

- Displaying current prices across nearby lots
- Guiding drivers toward cheaper or underutilized locations
- Offering time-based promotions or discounts during low-demand windows

This not only improves user satisfaction but also **actively balances parking load across locations**, contributing to smarter urban mobility.

---

5. Feedback-Driven Model Evolution

Another limitation is the absence of a behavioral feedback mechanism. In the future, the pricing engine could incorporate:

- User selection patterns (e.g., which lots drivers choose at certain price points)
- Payment completion data
- Survey-based driver satisfaction ratings

This would enable **behavioral tuning** of the pricing logic, improving its fairness, relevance, and long-term effectiveness.

---

6. Cross-City Scalability

The current model was tested in a simulated environment reflecting one urban context. However, its modular design supports:

- **Adaptation to multiple cities** with varied traffic and demand dynamics
- Differentiation between public, private, or event-specific zones
- Support for regional regulations, toll policies, and smart city infrastructure

Scalability will require model retraining or logic adjustment to account for **city-specific behaviors and infrastructure**, but the foundational structure supports replication and expansion.

Limitations and Considerations

Despite its success in simulation, several limitations restrict immediate deployment:

- Lack of live data streaming in the current notebook environment
- Manual parameter tuning, which may not generalize well across different lots
- Absence of driver feedback integration, reducing behavioral sensitivity
- Rule-based logic may oversimplify pricing in highly dynamic or unpredictable zones

---

Conclusion

With the right technological integrations and learning enhancements, this dynamic pricing engine can evolve into a powerful tool for modern urban mobility systems. Its ability to balance **efficiency, fairness, and economic optimization** positions it as a strong candidate for adoption in smart city infrastructure — provided it continues to evolve toward a fully automated, real-time, and user-centric architecture.

---

15. REFERENCES

1. Pathway. *From Jupyter to Deploy*. Retrieved from: https://pathway.com/developers/user-guide/deployment/from-jupyter-to-deploy/
2. Pathway. *First Real-Time App with Pathway*. Retrieved from: https://pathway.com/developers/user-guide/introduction/first_realtime_app_with_pathway/
3. Summer Analytics 2025. Retrieved from: https://www.caciitg.com/sa/course25/
4. Summer Analytics 2025 Problem Statement & Dataset – Provided by the Consulting & Analytics Club, IIT Guwahati.
5. Google Colab – Sample Implementation Notebook.