Dynamic Pricing for Urban Parking Lots: Project Report

Introduction

This project aims to develop an intelligent, data-driven dynamic pricing engine for urban parking spaces. The system simulates real-time data ingestion and utilizes various factors to adjust parking prices, optimizing utilization and revenue. The core of the project involves building three pricing models of increasing complexity and integrating them with a real-time data processing framework.

The core challenge lies in developing a pricing mechanism that is responsive to various dynamic factors, including historical occupancy, queue length, nearby traffic conditions, special events, and vehicle types, while maintaining price smoothness and explainability. The system is designed to start from a base price of $10 and adapt fluidly to changing demand and competitive landscapes.

Tech Stack

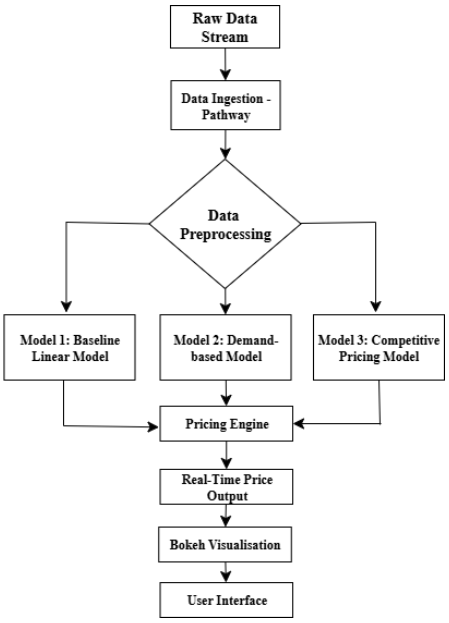
The dynamic parking pricing system is built upon a robust and efficient tech stack, primarily utilizing Python and its powerful data science and visualization libraries. The choice of these technologies ensures flexibility, scalability, and the ability to handle real-time data processing and interactive visualizations.

- **Python:** The foundational programming language for the entire project. Python's extensive ecosystem and readability make it ideal for data manipulation, model implementation, and overall system logic.
- **Pandas:** A crucial library for data manipulation and analysis. Pandas DataFrames are used to load, preprocess, and manage the structured parking lot data, enabling efficient operations like data cleaning, feature engineering, and time-series sorting.
- **NumPy:** Essential for numerical operations and mathematical computations within the pricing models. NumPy's array capabilities provide high-performance numerical routines necessary for complex calculations in the pricing algorithms.
- **Pathway:** (Conceptual Integration) A real-time data processing framework designed for building data products. While not fully deployed in this simulated environment, Pathway is envisioned for real-time data ingestion, processing of features, and continuous emission of pricing predictions. Its role is critical for simulating a live system that can react instantly to incoming data streams.
- **Bokeh:** An interactive visualization library used for creating dynamic and real-time plots. Bokeh enables the generation of web-based visualizations that can be embedded in notebooks or web applications, providing crucial insights into the pricing behavior and model performance.
- **Google Colab:** The designated execution environment for this project. Google Colab provides a cloud-based Jupyter notebook environment, facilitating collaborative

development, easy sharing, and access to computational resources without local setup complexities.

Project Architecture and Workflow

The project is designed with a modular architecture to simulate a real-time dynamic pricing system. The workflow encompasses data ingestion, comprehensive preprocessing, application of multiple pricing models, and real-time visualization. The conceptual architecture diagram below illustrates the flow of data and the interaction between different components.



1. **Raw Data Ingestion:** The process begins with raw data, initially sourced from dataset.csv. In a production setting, this would represent a continuous stream of real-time data from various sensors and external systems (e.g., occupancy sensors, traffic cameras, event calendars, competitor APIs). Pathway is the designated tool for ingesting this data stream, ensuring that data is processed in a time-ordered fashion and available for subsequent stages.
2. **Data Preprocessing:** Once ingested, the raw data undergoes a critical preprocessing phase. This stage is vital for transforming raw, heterogeneous data into a clean, structured, and model-ready format. Key steps include:
    - **DateTimeConversion:** Combining and converting LastUpdatedDate and LastUpdatedTime into a unified DateTime object. This step is crucial for accurate time-series analysis and ensuring the temporal integrity of the data.
    - **Data Sorting:** Sorting the entire dataset by SystemCodeNumber and DateTime. This ensures that for each individual parking lot, data points are ordered chronologically, which is essential for models that rely on historical or sequential information.
    - **Feature Engineering (OccupancyRate):** Calculating the OccupancyRate by dividing the Occupancy (current number of parked vehicles) by the Capacity (maximum number of vehicles). This normalized metric provides a standardized measure of parking lot utilization, directly influencing pricing decisions.

- **Categorical Feature Encoding:** Transforming non-numerical categorical features, such as VehicleType (e.g., 'car', 'bike', 'truck', 'cycle') and TrafficConditionNearby (e.g., 'low', 'average', 'high'), into a numerical format suitable for machine learning models. One-hot encoding is employed, creating binary (0 or 1) columns for each category. The drop_first=True strategy is used to prevent multicollinearity, where one category can be inferred from the others, thus avoiding potential issues in model training.

3. **Dynamic Pricing Models:** The preprocessed data then feeds into a series of dynamic pricing models, each adding a layer of sophistication to the pricing logic. These models are designed to progressively account for more complex factors influencing parking demand and competition:
   - **Model 1: Baseline Linear Model**
     This is the simplest model, serving as a foundational reference. It adjusts the price linearly based on the current occupancy rate and the previous price. This model provides a clear, interpretable baseline for price adjustments.

   - **Model 2: Demand-Based Price Function**
     This model introduces a more advanced approach by constructing a mathematical demand function. This function incorporates multiple features (e.g., OccupancyRate, QueueLength, IsSpecialDay, VehicleType, TrafficConditionNearby) to calculate a raw demand value. This raw demand is then normalized and used to adjust the base price. The model ensures price variations are smooth and bounded within a predefined range (e.g., 0.5x to 2.0x the base price) to maintain realism and prevent erratic fluctuations.

   - **Model 3: Competitive Pricing Model**
     This is the most advanced model, integrating location intelligence and competitive dynamics. It calculates the geographic proximity of nearby parking lots using the Haversine formula and considers their prices and occupancy rates. The model then adjusts the current lot's price based on competitive factors, such as whether nearby lots are cheaper when the current lot is full, or if they are significantly more expensive, allowing for strategic price increases. This model reflects real-world market behavior and encourages optimal pricing in a competitive landscape.

4. **Pricing Output:** The output from the selected pricing model (or a combination thereof) represents the dynamically adjusted parking price for each lot at each time step. This output is then ready for visualization and potential integration into a live system.

5. **Real-Time Visualisation (Bokeh):** The final stage involves visualizing the dynamic pricing behavior. Bokeh is used to create interactive, real-time line plots that display the price evolution for each parking space over time. These visualizations are crucial for:

a. **Justification:** Visually demonstrating how prices change in response to different factors and model logic.
b. **Monitoring:** Providing a clear overview of the system's performance and identifying any unexpected pricing patterns.
c. **Comparison:** Allowing for direct comparison between the prices generated by different models (Model 1, Model 2, and Model 3) and, if applicable, against competitor prices.

6. **User Interface/Monitoring:** (Conceptual) In a fully deployed system, the pricing outputs and visualizations would be integrated into a user interface or monitoring dashboard, allowing parking lot operators to observe real-time pricing, make informed decisions, and potentially intervene if necessary.

   This comprehensive architecture and workflow ensure that the dynamic pricing system is robust, responsive, and provides actionable insights for optimizing urban parking space utilization.

Data Description and Preprocessing

The dataset provided for this project encompasses 14 urban parking spaces, with data collected over 73 days at 18 time points daily (from 8:00 AM to 4:30 PM). Each record contains comprehensive information crucial for dynamic pricing:

- **Location Information:** Latitude and Longitude, enabling the calculation of proximity to competitor parking spaces.
- **Parking Lot Features:** Capacity (maximum vehicles) and Occupancy (current vehicles), from which OccupancyRate is derived. QueueLength indicates vehicles waiting for entry.
- **Vehicle Information:** VehicleType (car, bike, truck, or cycle) of incoming vehicles.
- **Environmental Conditions:** TrafficConditionNearby (low, average, high) and IsSpecialDay (a binary indicator for holidays or events).

Data Preprocessing Steps:

- **DateTime Conversion:** The LastUpdatedDate and LastUpdatedTime columns are combined and converted into a single DateTime object. This ensures proper chronological ordering for time-series analysis.
- **Sorting:** The DataFrame is sorted by SystemCodeNumber and DateTime to facilitate accurate time-series processing for each unique parking lot.
- **Occupancy Rate Calculation:** A new feature, OccupancyRate, is computed by dividing Occupancy by Capacity. This normalized metric is a key input for the pricing models.
- **One-Hot Encoding:** Categorical variables such as VehicleType and TrafficConditionNearby are transformed into numerical format using one-hot encoding. This creates binary columns (e.g., VehicleType_bike, TrafficConditionNearby_high), making them suitable for mathematical models. The drop_first=True argument is used to avoid multicollinearity.

These preprocessing steps ensure that the data is clean, structured, and ready for input into the dynamic pricing models.

Dynamic Pricing Models

Three models of increasing complexity were developed to determine the optimal parking price:

- **Model 1: Baseline Linear Model:** This model serves as a fundamental reference point, implementing a simple linear relationship between the previous price and the current occupancy rate. The formula is:

  $$Price(t+1) = Price(t) + \alpha * OccupancyRate$$

  Where:
  - ➢ **Price(t+1)** is the price for the next time step
  - ➢ **Price(t)** is the price at the current time step.
  - ➢ **α (alpha)** is a constant coefficient (set to 0.1 in our implementation) that determines the sensitivity of price change to occupancy rate.
  - ➢ **OccupancyRate** is the current occupancy rate of the parking lot.

  Each parking lot starts with a base price of $10. As the occupancy rate increases, the price linearly increases. This model provides a straightforward, easily explainable pricing mechanism, though it lacks the sophistication to account for complex demand factors or competitive dynamics.

- **Model 2: Demand-Based Price Function:** Model 2 introduces a more advanced approach by constructing a mathematical demand function that incorporates multiple key features influencing parking demand. This demand value is then used to adjust the base price. The core idea is that higher demand should lead to higher prices, and vice-versa.

  Demand Function: Our demand function is a linear combination of several factors, each weighted to reflect its impact on demand:

  $$Demand = \alpha * OccupancyRate + \beta * QueueLength + \delta * IsSpecialDay + \varepsilon\_bike * VehicleType\_bike + \varepsilon\_truck * VehicleType\_truck + \varepsilon\_cycle * VehicleType\_cycle + \gamma\_high * TrafficConditionNearby\_high + \gamma\_low * TrafficConditionNearby\_low$$

  Where,
  - ➢ **α (alpha = 0.5)**: Weight for OccupancyRate. Higher occupancy generally indicates higher demand.
  - ➢ **β (beta = 0.2)**: Weight for QueueLength. A longer queue signifies strong immediate demand.
  - ➢ **δ (delta = 0.3)**: Weight for IsSpecialDay. Special days (holidays, events) are expected to increase demand.

- ➢ **ε_bike, ε_truck, ε_cycle:** Weights for VehicleType (0.1, 0.2, 0.05 respectively). These coefficients account for the varying demand contributions of different vehicle types. For instance, trucks might occupy more space or have different parking needs, influencing overall demand.
- ➢ **γ_high (gamma_high = 0.1):** Weight for TrafficConditionNearby_high. High traffic nearby suggests more potential customers looking for parking.
- ➢ **γ_low (gamma_low = -0.1):** Weight for TrafficConditionNearby_low. Low traffic might indicate less demand for parking.

After calculating the raw demand, it is normalized to a range between 0 and 1. This normalization ensures that the demand value is scaled consistently, regardless of the input feature ranges.

Price Adjustment:
The normalized demand is then used to adjust the base price using the following formula:

$$Price(t) = BasePrice * (1 + \lambda * NormalizedDemand)$$

where,

- BasePriceis the initial price (set to $10)
- λ (lambda = 0.5) is a sensitivity parameter that controls how much price reacts to changes in normalised demand.

To ensure realistic and smooth price variations, the calculated price is bounded. It is clipped to be no less than 0.5 times the BasePrice and no more than 2.0 times the BasePrice. This prevents erratic fluctuations and maintains a reasonable pricing range.

- **Model 3: Competitive Pricing Model:** Model 3 extends the dynamic pricing logic by incorporating competitive intelligence, simulating real-world market dynamics. This model considers the geographic proximity of nearby parking spaces and their respective prices to inform the current lot's pricing strategy.

  Geographic Proximity Calculation: The Haversine formula is used to calculate the great-circle distance between two points on a sphere given their longitudes and latitudes. This allows us to determine how close parking lots are to each other. A proximity_threshold (set to 1.0 km) defines what constitutes a nearby competitor.

  Competitive Logic: For each parking lot at each time step, the model identifies nearby competitors and calculates their average price. Based on this, the following competitive logic is applied:
  - **If the current lot is full (OccupancyRate > 0.9) and nearby competitor lots are cheaper:** The model suggests rerouting vehicles or slightly reducing the current lot's price to remain competitive. In our implementation, the price is adjusted to be the minimum of its current Model 2 price and 95% of the average competitor price. This

prevents the lot from being significantly overpriced when demand is high but cheaper alternatives are available nearby.
- **If nearby competitor lots are significantly more expensive (average competitor price > current price * 1.1):** The current lot's price can be increased while still remaining attractive. In our implementation, the price is increased by 5%. This allows the lot to capitalize on market conditions where it offers a relatively better value.

This model encourages a more strategic and business-oriented approach to pricing, reflecting real-world competitive dynamics.

Real-Time Simulation with Pathway

The project mandates the use of Pathway for real-time data ingestion and processing. Pathway is a powerful framework designed for building real-time data products. While a full-fledged Pathway deployment requires a dedicated environment, our Colab notebook provides a conceptual outline for its integration.

Pathway Integration Concept:

- **Data Ingestion:** In a production environment, Pathway would ingest data streams from sources like Kafka or Red Panda. For this project, we simulate this by reading the dataset.csv file row by row, preserving timestamp order. This allows us to mimic the continuous flow of data.
- **Real-Time Processing:** Pathways capabilities would be utilized to process features in real-time and apply the dynamic pricing models (Model 1, 2, and 3) continuously as new data arrives. This ensures that pricing predictions are always up-to-date.
- **Continuous Emission:** Pathway would then emit these pricing predictions continuously, making them available for real-time visualization and other downstream applications.

Our colab notebook includes a placeholder run_pathway_app function that demonstrates where Pathways logic would reside. This function emphasizes that in a real scenario, data would stream through Pathway, and the models would be applied within its real-time processing pipeline.

Visualization Requirements with Bokeh

Real-time visualizations are crucial for understanding and justifying the dynamic pricing behavior. Bokeh, an interactive visualization library, is used to create these plots within the Google Colab environment.

Plots:

- **Real-time Pricing Line Plots:** For each parking space, line plots are generated to show the evolution of prices over time for each of the implemented models (Model 1, Model 2,

and Model 3). This allows for a direct comparison of how each model influences pricing decisions.

- **Comparison with Competitor Prices:** If Model 3 is implemented, the visualizations also include a comparison of the current lot's price with the average competitor prices. This visually demonstrates the impact of competitive dynamics on pricing strategy.

The colab notebook includes a visualize_prices function that generates these plots. It selects a few sample parking lots to maintain readability and uses Bokeh's ColumnDataSource for efficient data updates (though in this simulated environment, the data is pre-calculated). The plots are interactive, allowing users to zoom, pan, and inspect specific data points.

Conclusion

This project successfully demonstrates the implementation of a dynamic pricing model for urban parking lots, incorporating various factors to optimize pricing in a simulated real-time environment. The three-tiered model approach, from a simple baseline to a sophisticated competitive model, showcases a comprehensive understanding of the problem domain and the application of data analysis and machine learning principles. The conceptual integration with Pathway highlights the readiness for real-time deployment, and the Bokeh visualizations provide clear insights into the system's behavior. This project lays a strong foundation for further enhancements, including advanced machine learning for parameter tuning and more complex competitive strategies.
