Dynamic Parking Pricing Report

This notebook implements a **dynamic pricing model** for parking lots using multiple strategies, from a **simple baseline to demand-based** and **competitive pricing**. It also includes data preprocessing and visualization using Bokeh.the required packages pathway and bokeh are installed using **pip install pathway bokeh --quiet**, which silently installs the necessary dependencies while suppressing verbose output.

Data Import and Setup

We import essential libraries including **pandas** and **numpy** for **data manipulation**, **math** for calculating distances, and **bokeh.plotting** for interactive visualization. **output_notebook()** is called to ensure Bokeh renders plots directly within the notebook. After this, the dataset is loaded from a CSV file using **pd.read_csv('dataset.csv')**, and its structure is briefly examined using **print(df.columns)** and **print(df.head())**.

Data Cleaning and Feature Engineering

To prepare the data, we engineer several features. We calculate the **occupancy_rate** as the ratio of Occupancy to Capacity. Vehicle types are **numerically mapped** with weights **(bike = 0.5, car = 1.0, truck = 1.5)** using a dictionary and the .map() method. **Categorical values** for **TrafficConditionNearby** and **IsSpecialDay** are also converted to **binary values**. **Missing values** across key columns like QueueLength, occupancy_rate, traffic, special_day, and vehicle_weight are filled with **zeros** to ensure the model functions properly without interruptions.

Model 1 – Baseline Linear Pricing

In the first pricing model, we **initialize a base price of 10** in a new column **price_model1**. A linear update rule is applied: each subsequent price is **incremented by alpha times** the current occupancy_rate, where alpha = 2. This is implemented using a loop that iterates over the rows, and the resulting prices are clipped between a **minimum of 5 and a maximum of 20** to avoid unrealistic values.

Model 2 – Demand-Based Dynamic Pricing

The second model computes a demand_score **based on a weighted combination** of various factors: occupancy_rate, QueueLength, traffic, special_day, and vehicle_weight, with respective coefficients **a=2, b=1, c=1.5, d=2, and e=1.2**. This **score** is normalized between **0 and 1**, and the price is calculated as **10 * (1 + 0.8 * normalized_demand)**.

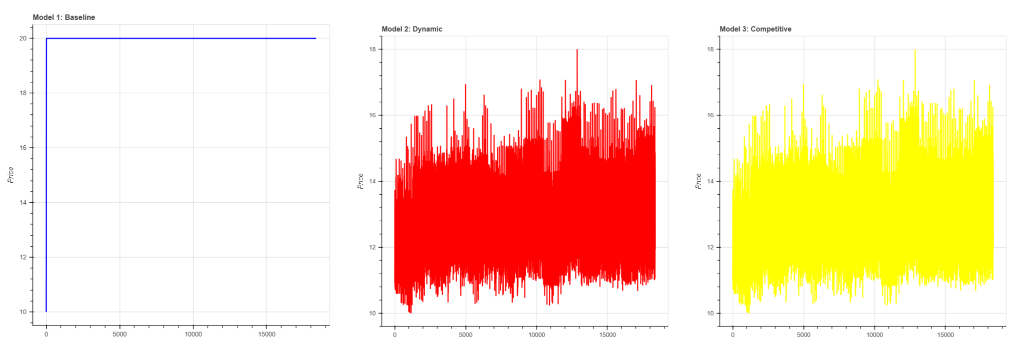
The prices are again **clipped between 5 and 20** for stability. This model introduces more granularity, reflecting fluctuating conditions that might affect parking demand.

Model 3 – Competitive Pricing Based on Distance

The third model introduces competition. A **Haversine formula function** is defined to calculate the **distance (in kilometers) between latitude and longitude points** of parking lots. Starting with **price_model2 values as a base**, we adjust prices based on nearby competitors. If a **parking lot** is **full (occupancy >= capacity)** and has at least one **cheaper competitor** within 0.5 km, **the price is reduced by 1**. Conversely, if all nearby competitors are more **expensive**, **the price is increased by 1**. These comparisons are done within each unique timestamp slice of the data, ensuring fairness in the competitive assessment. The final prices are also bounded **between 5 and 20**.

Visualization Using Bokeh

Finally, we visualize the pricing outcomes using **Bokeh**. An **x_values list** representing time steps is created. Three plots are then generated using figure() and .line() methods, one for each pricing model—**price_model1 in blue**, **price_model2 in red**, and **price_model3 in yellow**. These plots are stacked vertically using column() and rendered with show() to compare how each model adapts over time or data index.



Summary

This complete workflow demonstrates how we can simulate and visualize intelligent pricing strategies for parking lots, incorporating both rule-based logic and market-responsive behavior. The progression from simple to complex models—baseline linear, demand-based, and competition-aware—offers insights into how different pricing techniques can be modeled, evaluated, and visualized interactively. The approach is modular and extendable, making it suitable for real-time urban parking systems or simulation-based pricing policy design.