Dynamic Pricing for Urban Parking Lots

Capstone Project of Summer Analytics 2025

hosted by Consulting & Analytics Club × Pathway

Project Report

By : Krish Sandhu

GitHub : [GIT-KrishSandhu]

Project : [View on Google Colab]

College : KIET Group of Institutions, Uttar Pradesh

Demand Function

The price for each day is calculated via the demand function, based on daily aggregates of values (max, min, average).

- Model 1 : The demand function in Model 1, similar to the one given in the sample notebook, for the exception of the multiplying variable, in model 1, the value of alpha is given by,

  alpha = ((pw.this.occ_max / pw.this.occ_min) ** 0.25)

  Meaning alpha scales accordingly with the occupancy per day.

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

- Model 2 : Demand is given by,

```
daily_agg = daily_agg.with_columns(
    demand = (
        (pw.this.is_special * 1) * (                                # if special day = 1, 1.6, is a scaling feature
            ((pw.this.occ_max/pw.this.occ_min) ** 0.35) *             # alpha constant = ((pw.this.occ_max/pw.this.occ_min) ** 0.35)
             ((pw.this.occ_max - pw.this.occ_min) / pw.this.cap)
            + 0.8 * (pw.this.traffic_mode - 1)                        # constant = 0.8, multiplier for traffic
            + (2/45) * (pw.this.queue_avg)                            # constant = 2/45,  average queue length, normalized (divide by 15)*(2/3)
            + ((pw.this.vehicle_mode - 1)/ 3)                         # formula with normalization, range will be (-0.33, 0.66), to control price spike
        )
        - (pw.this.is_special - 1) * (                                # if special day = 0, note values of constants below are lesser, for a regular day
            ((pw.this.occ_max/pw.this.occ_min) ** 0.25) *             # alpha constant = ((pw.this.occ_max/pw.this.occ_min) ** 0.25)
             ((pw.this.occ_max - pw.this.occ_min) / pw.this.cap)
            + 0.5 * (pw.this.traffic_mode - 1)                        # constant = 0.5, multiplier for traffic
            + (1/45) * (pw.this.queue_avg)                           # constant = 1/45,  average queue length, normalized (divide by 15)*(1/3)
            + ((pw.this.vehicle_mode + 1)/ 3)                         # range (0.33, 1.33)
        )
    )
)
```

Variables used, the Feature Columns and value range-

1. SD = IsSpecialDay (0 or 1),
2. Occ(max and min) = Occupancy
3. Capacity

4. Traffic = TrafficConditionNearby {'low': 0, 'average': 1, 'high': 2}
5. Q = QueueLength (0 to 15)
6. V = VehicleType {'cycle': 0, 'bike': 1, 'car': 2, 'truck': 3}

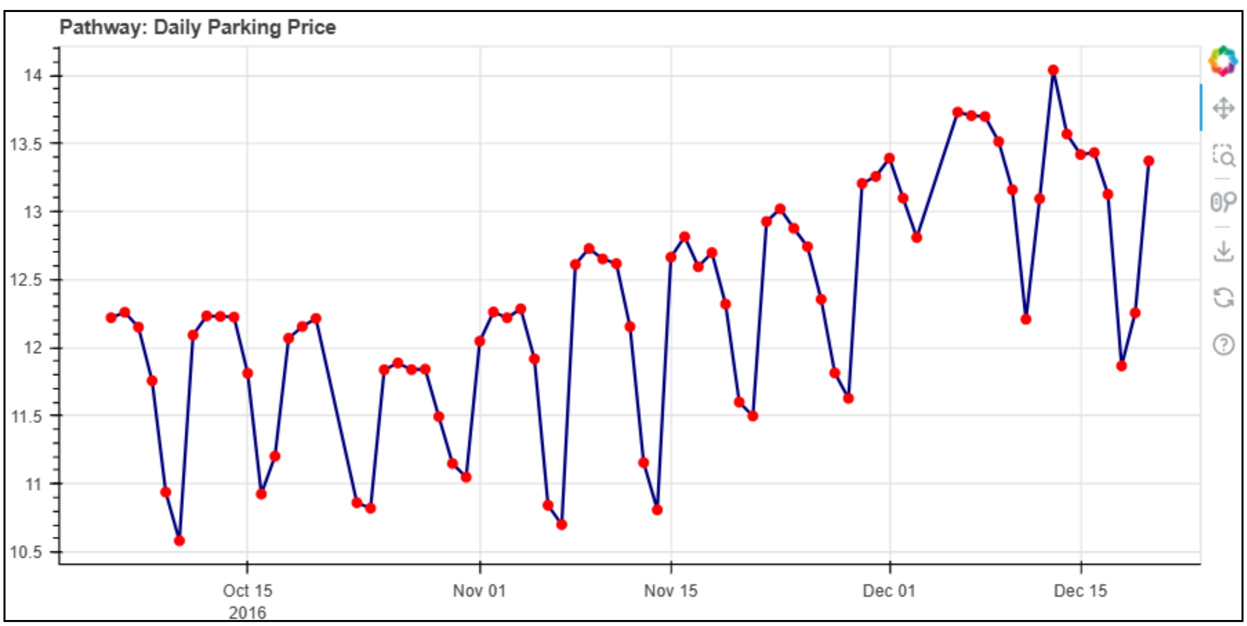
Price for parking at the lot are then calculated from this function,

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

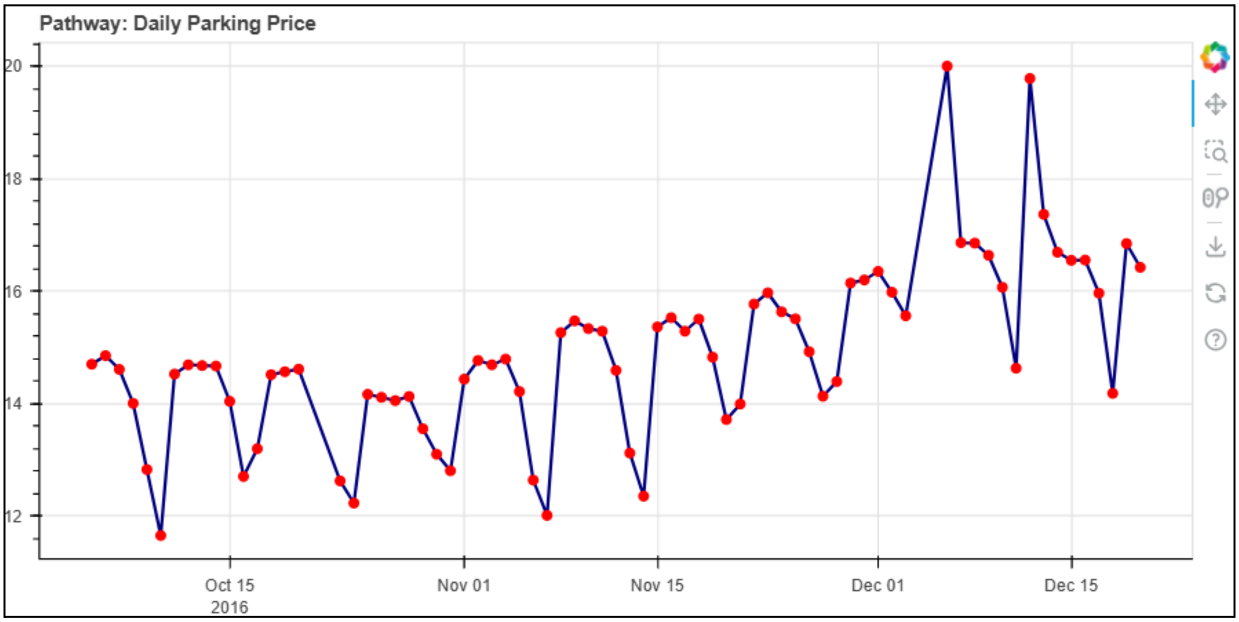
Then, if price is bounded such that if it exceeds (2 x BasePrice) then it is to be fixed at that value, same for lower limit (0.5 x BasePrice), here BasePrice = 10.

Visualizations

- <u>Model 1 :</u>



- <u>Model 2</u> :



Trends & Conclusions

In this study, we assumed that parking demand is influenced by all the variables defined above. The final pricing strategy reflects a bounded, demand-sensitive model: when demand increases — particularly due to spikes in occupancy variability, traffic congestion, or larger vehicle presence — the price rises accordingly. Conversely, on regular days with lighter traffic or fewer parked vehicles, prices are constrained to lower values. This behavior emulates real-world adaptive pricing systems and ensures that the model remains both flexible and economically rational. The curve progression in the graph confirms that the system successfully captures non-linear fluctuations in parking demand.