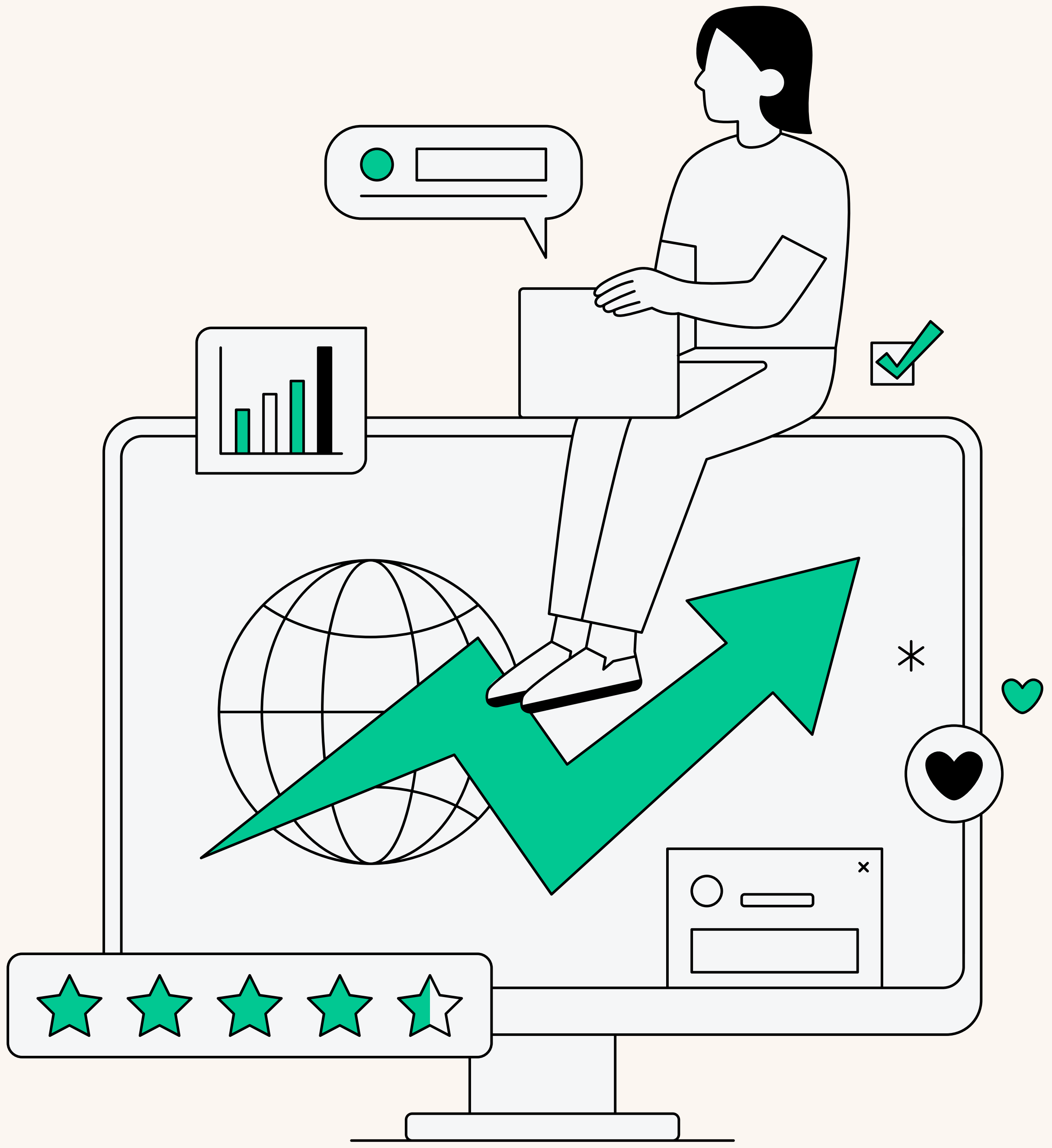
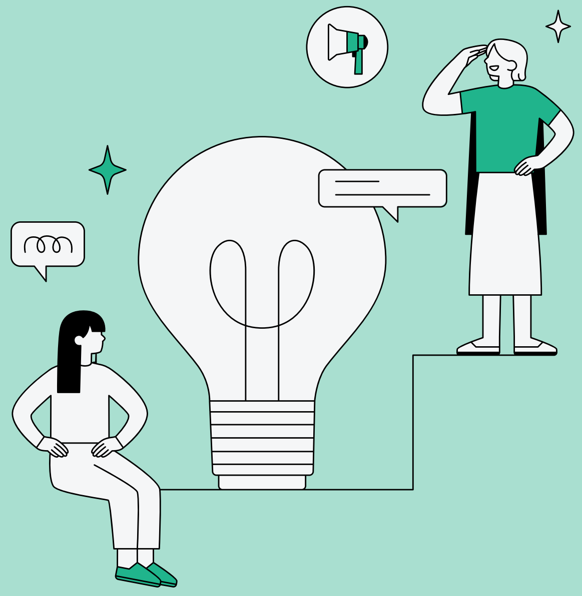
Dynamic Pricing for Urban Parking Lots

Presented by- Tavgun Kaur



Project Overview



- Urban parking spaces are a limited and highly demanded resource. Prices that remain static throughout the day can lead to inefficiencies — either overcrowding or underutilization.
- To improve utilization, dynamic pricing based on demand, competition, and real-time conditions is crucial.
- This project simulates such a system that creates an intelligent, data-driven pricing engine for 14 parking spaces using real-time data streams, basic economic theory and ML models built from scratch, using only numpy, pandas libraries.

Data collected from 14 urban parking spaces over 73 days, sampled at 18 time points per day with 30 minutes of time difference (from 8:00 AM to 4:30 PM the same day).

Data Preprocessing and Feature Engineering

```
LotID
BHMBCCMKT01          1312
BHMNCPHST01          1312
BHMMBMMBX01          1312
BHMNCPNST01          1312
Shopping             1312
BHMEURBRD01          1312
Broad Street         1312
Others-CCCPS8        1312
Others-CCCPS105a     1312
Others-CCCPS119a     1312
BHMBCCTHL01          1312
Others-CCCPS135a     1312
Others-CCCPS202      1312
Others-CCCPS98       1312
Name: count, dtype: int64
```

Handling categorical data like ***Vehicle Type and TrafficConditionNearby*** *so as to optimize faster using numerical techniques.*

```
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   SystemCodeNumber       18368 non-null  object
 1   Capacity               18368 non-null  int64
 2   Latitude               18368 non-null  float64
 3   Longitude              18368 non-null  float64
 4   Occupancy              18368 non-null  int64
 5   VehicleType            18368 non-null  object
 6   TrafficConditionNearby 18368 non-null  object
 7   QueueLength            18368 non-null  int64
 8   IsSpecialDay           18368 non-null  int64
 9   LastUpdatedDate        18368 non-null  object
 10  LastUpdatedTime        18368 non-null  object
dtypes: float64(2), int64(4), object(5)
memory usage: 1.5+ MB
```

Introducing new features like :

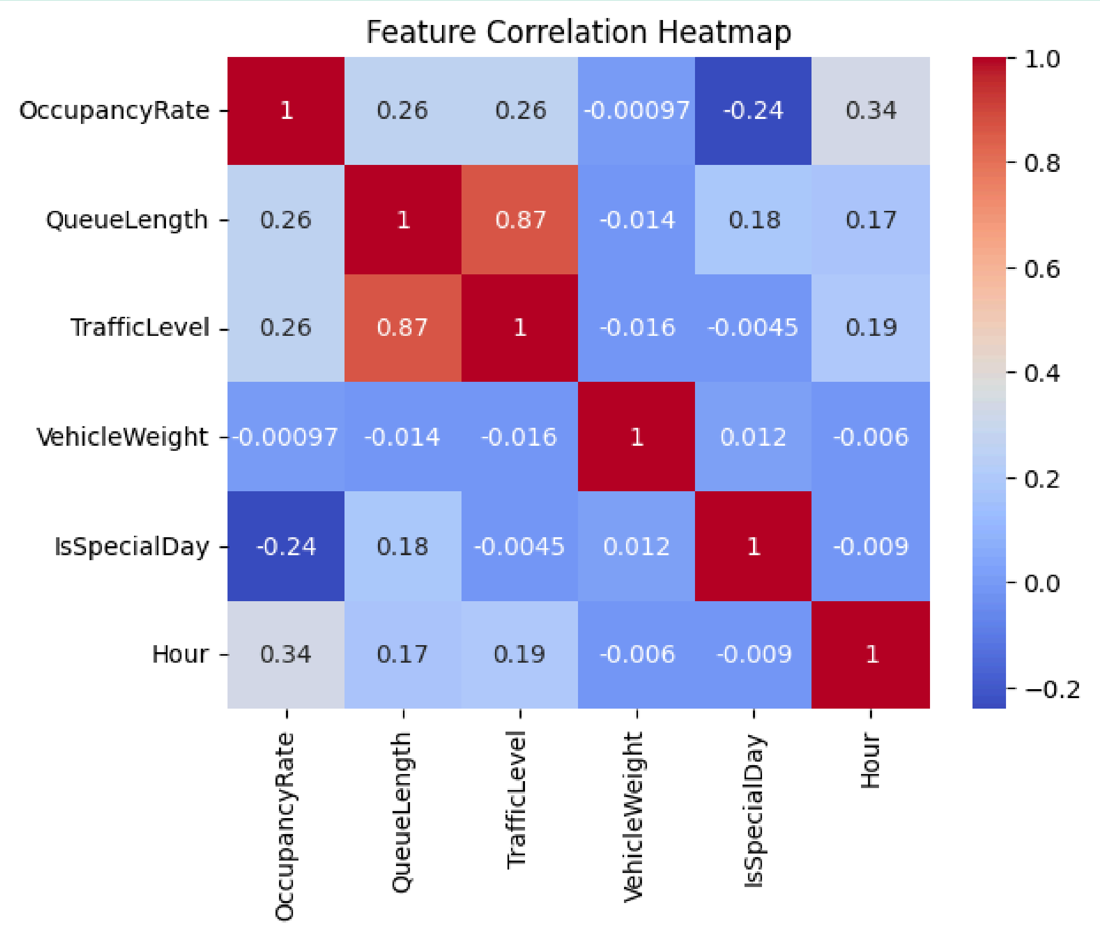
1.Occupancy Rate :

- This feature computes the current usage level of each parking lot as a proportion of its total capacity.
- It normalizes raw occupancy counts, allowing comparison across lots of different sizes.
- High occupancy rates signal higher demand and are crucial for dynamic pricing decisions.

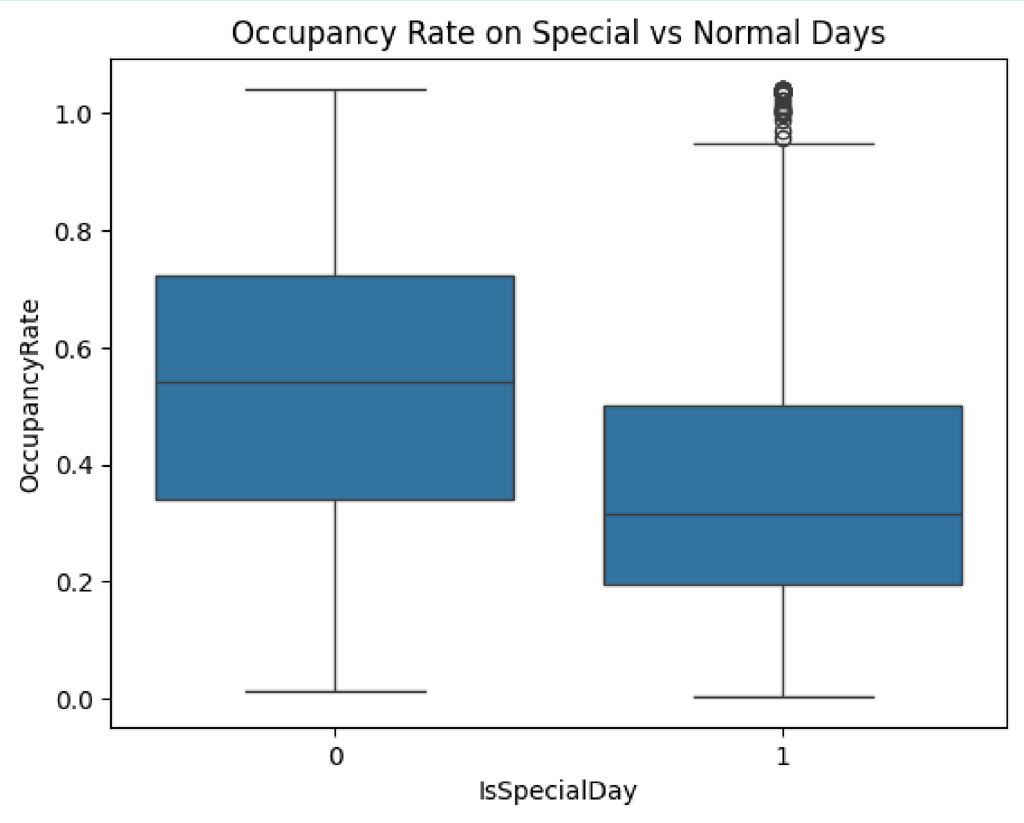
2.IsSpike:

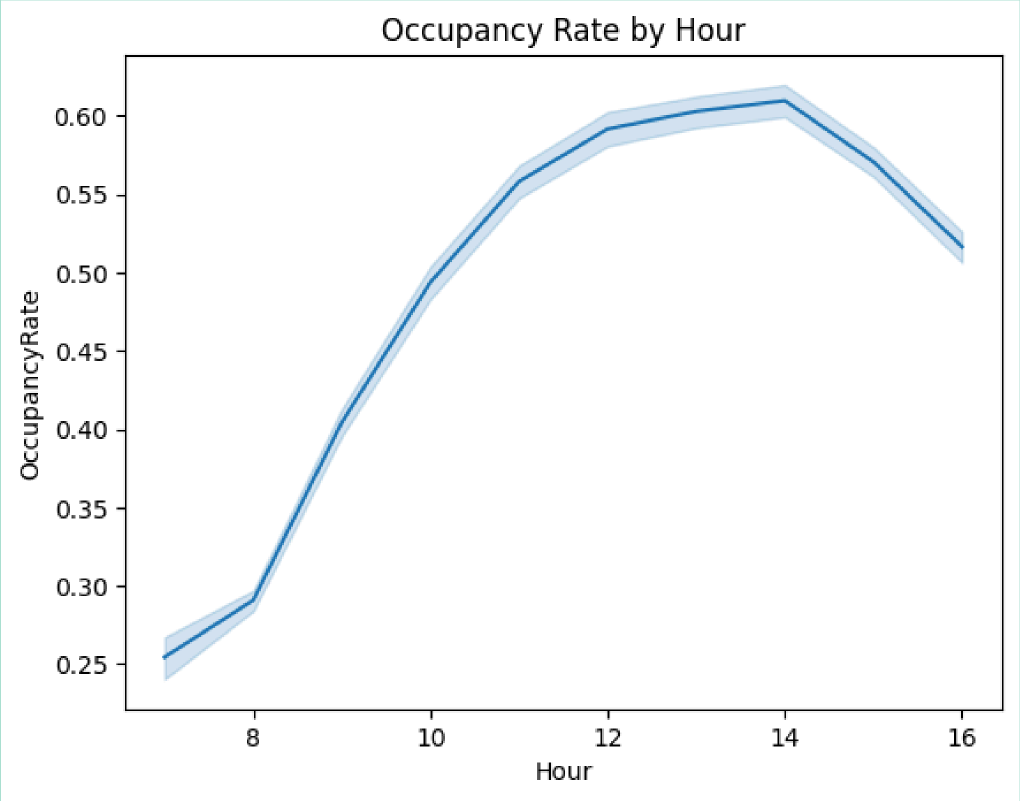
- Flags a demand spike when both: There are more than 2 vehicles waiting (QueueLength > 2) or The lot is almost full (OccupancyRate > 90%)
- Combines two indicators (high queue + high occupancy) to detect urgent high demand situations.

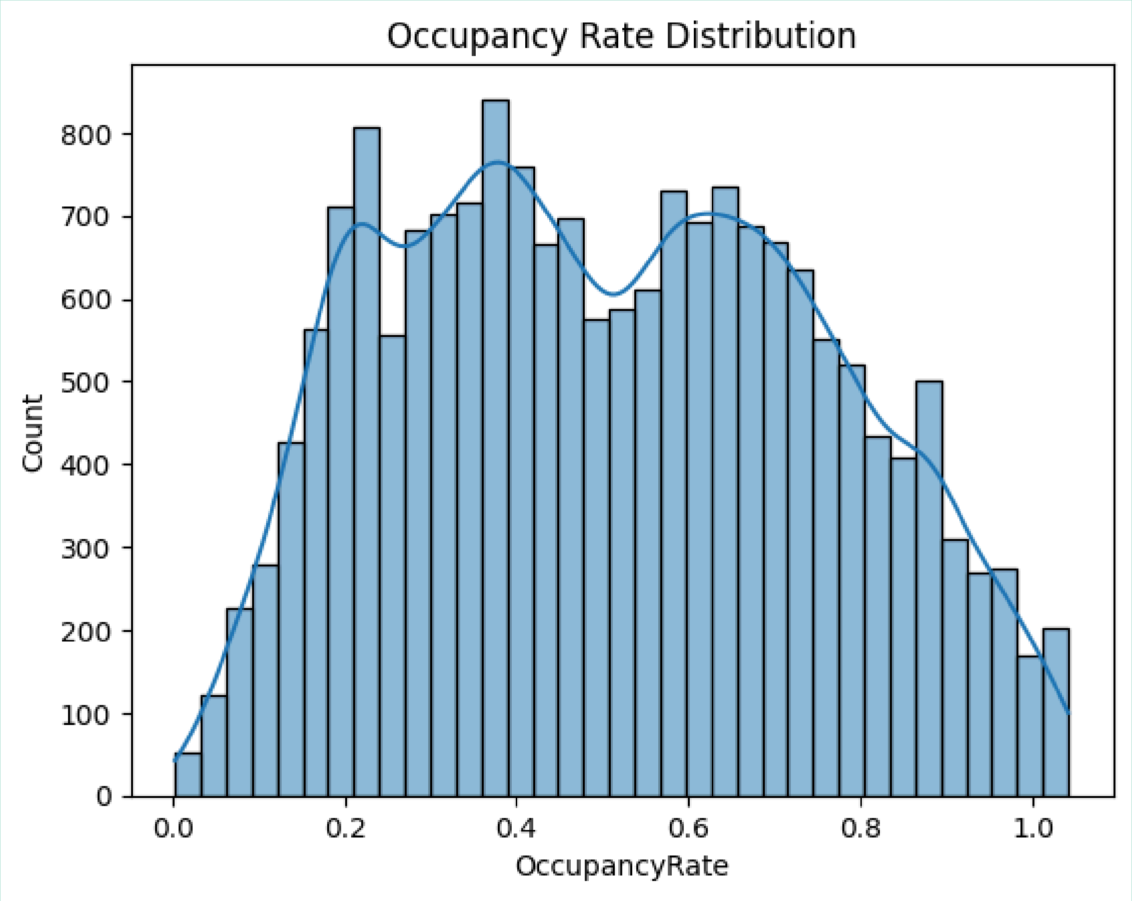
```
df['OccupancyRate'] = df['Occupancy'] / df['Capacity']
df['IsSpike'] = ((df['QueueLength'] > 2) & (df['OccupancyRate'] > 0.9)).astype(int)
```



Exploratory Data Analysis







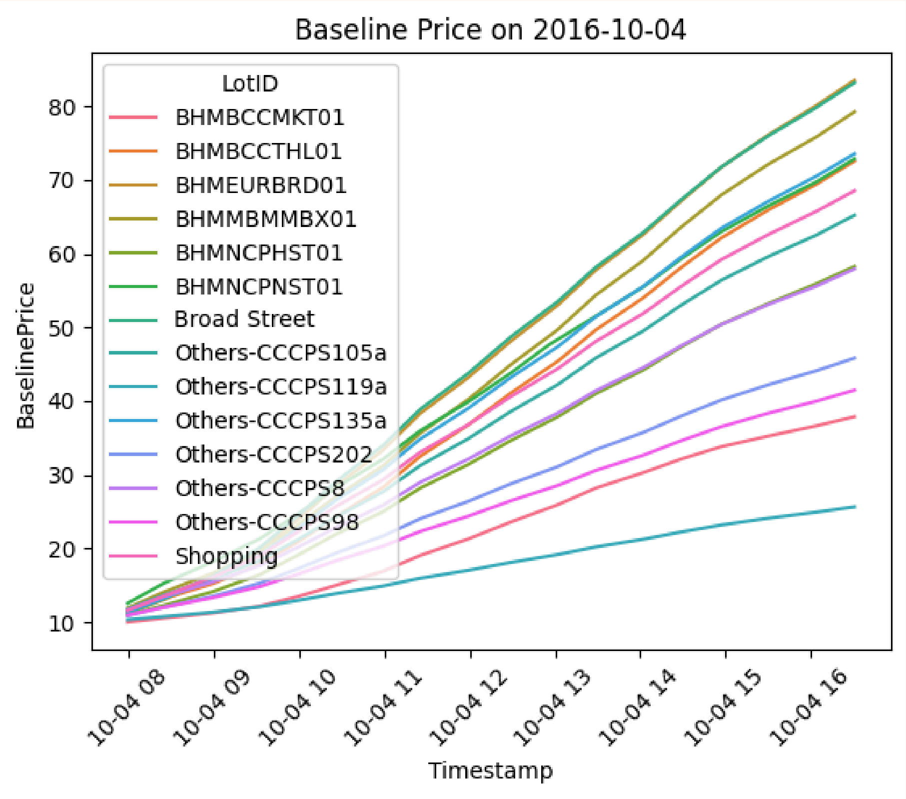
Model 1: Baseline Linear Model

A simple model where the next price is a function of the previous price and current occupancy:

- Linear price increase as occupancy increases
- Acts as a reference point

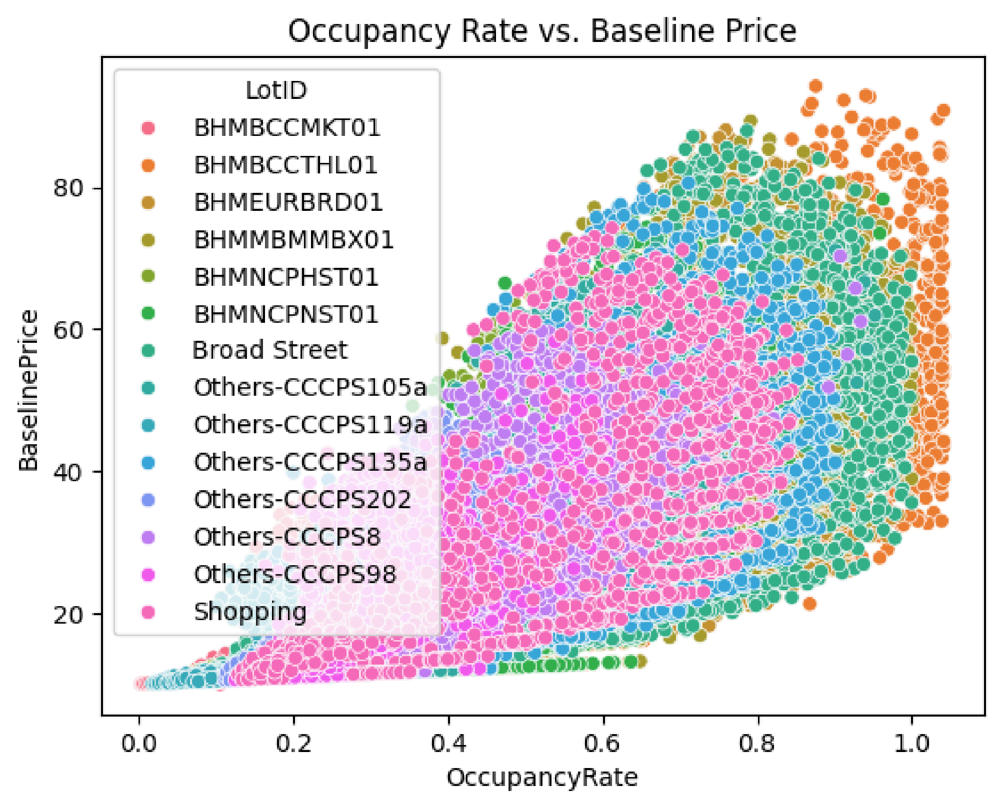
Formula Used:

$$Price(t) = Price(t-1) + \alpha \cdot OccupancyRate$$

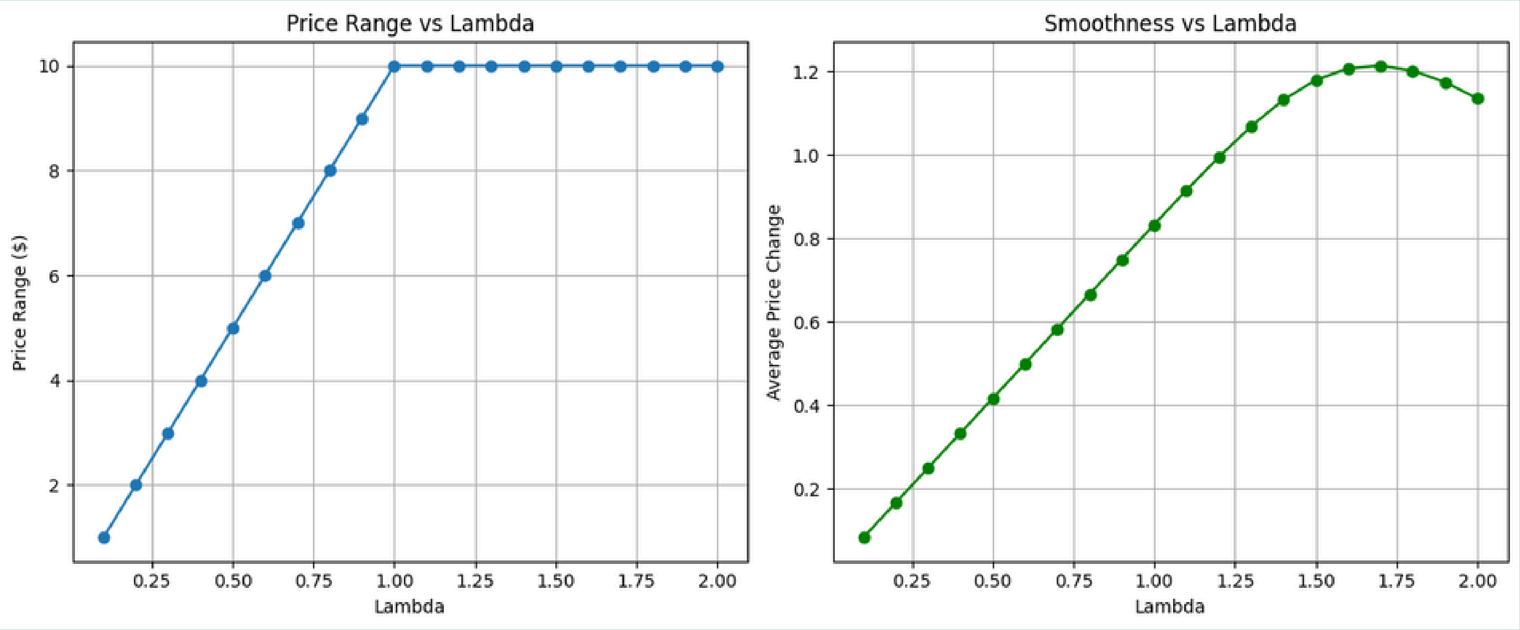


Prices start uniformly in the morning and diverge through the day. Some lots increase much faster, suggesting higher or more consistent occupancy.

Model 1 successfully captures the upward trend in pricing with occupancy, but prices compound over time, potentially creating divergence across lots.



Model 2: Demand-Based Price Function

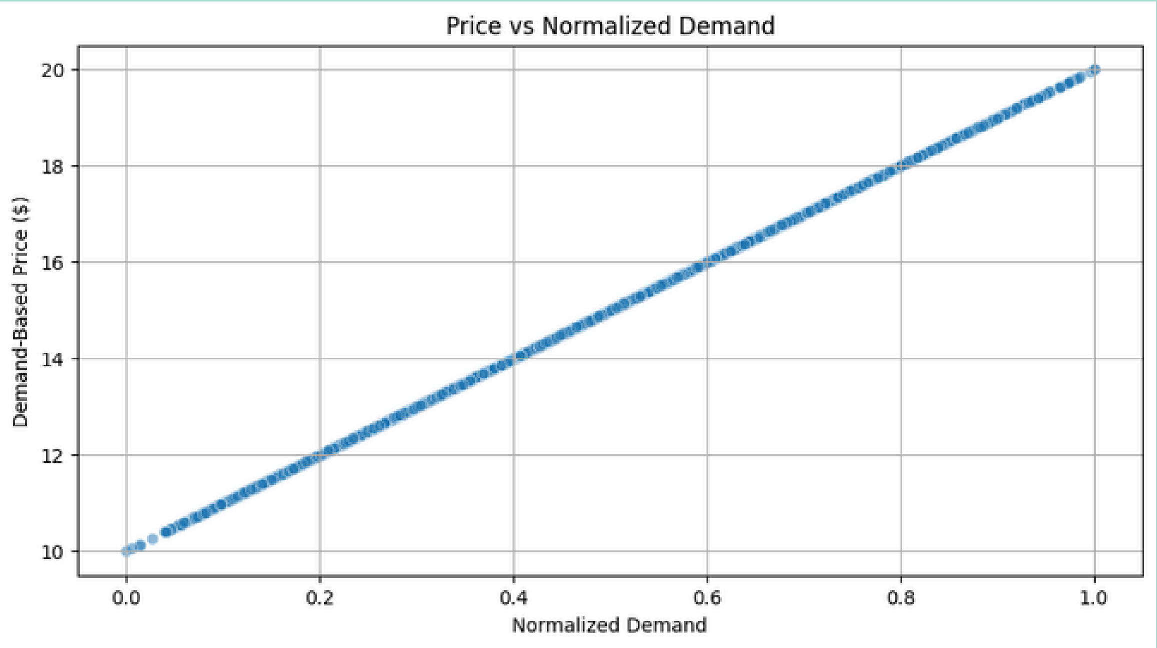


$\lambda = 1.0$ is a threshold point where:

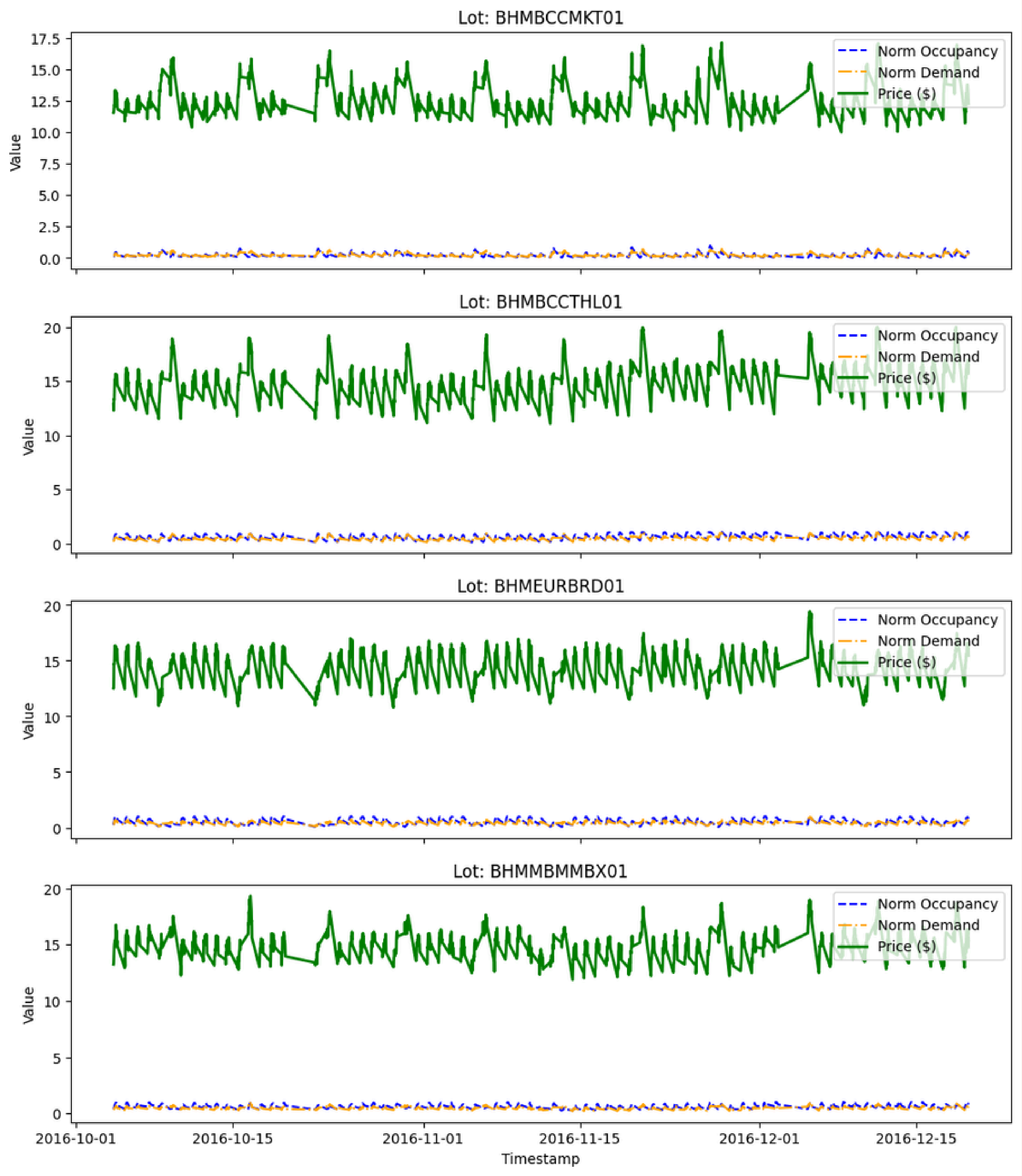
- You maximize price range (left plot).
- You still get reasonably smooth pricing (right plot).

As occupancy increases, so does price.

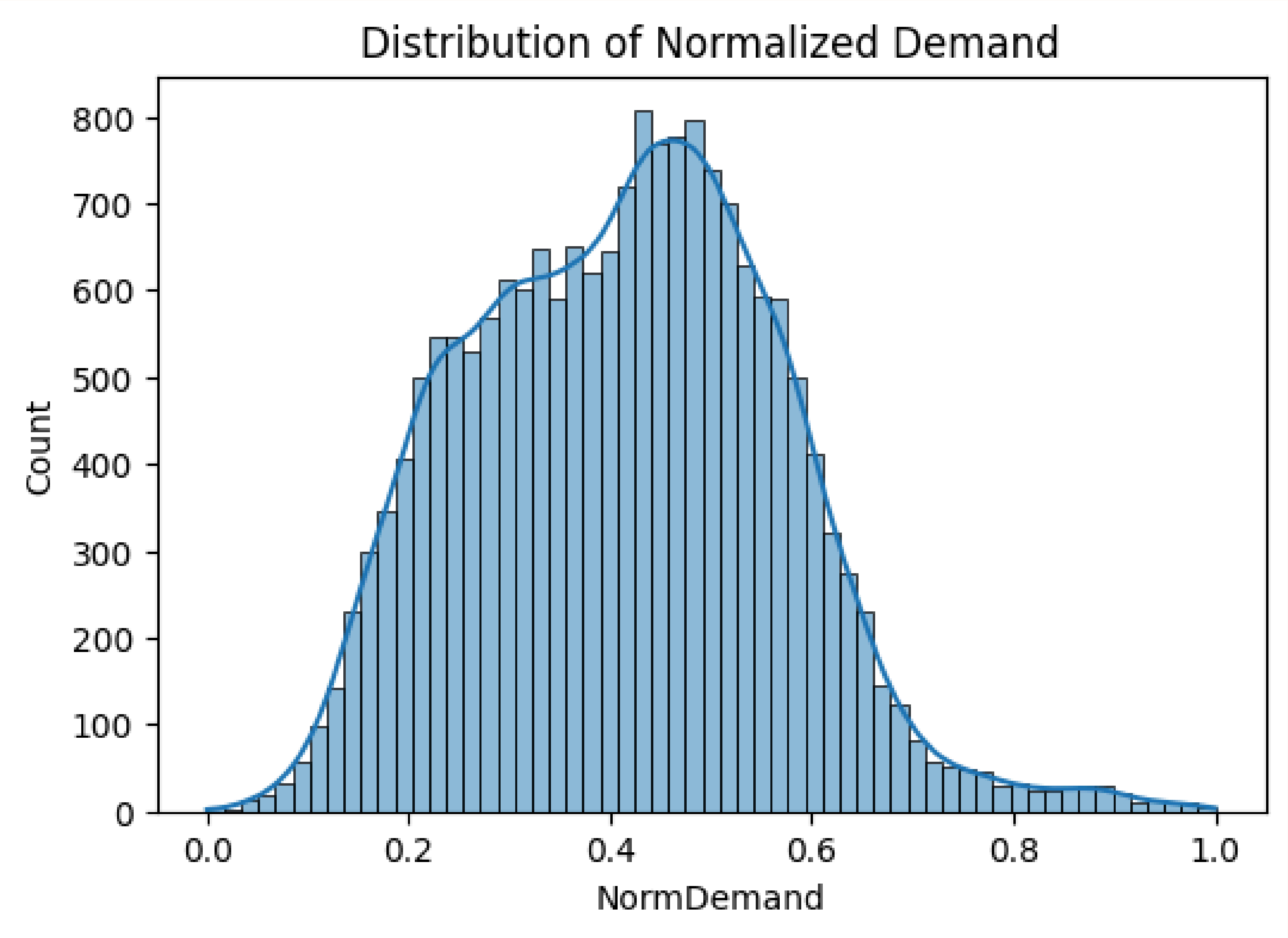
Confirms model correctly incorporates IsSpecialDay factor.



A perfect linear relationship between normalized demand and price.

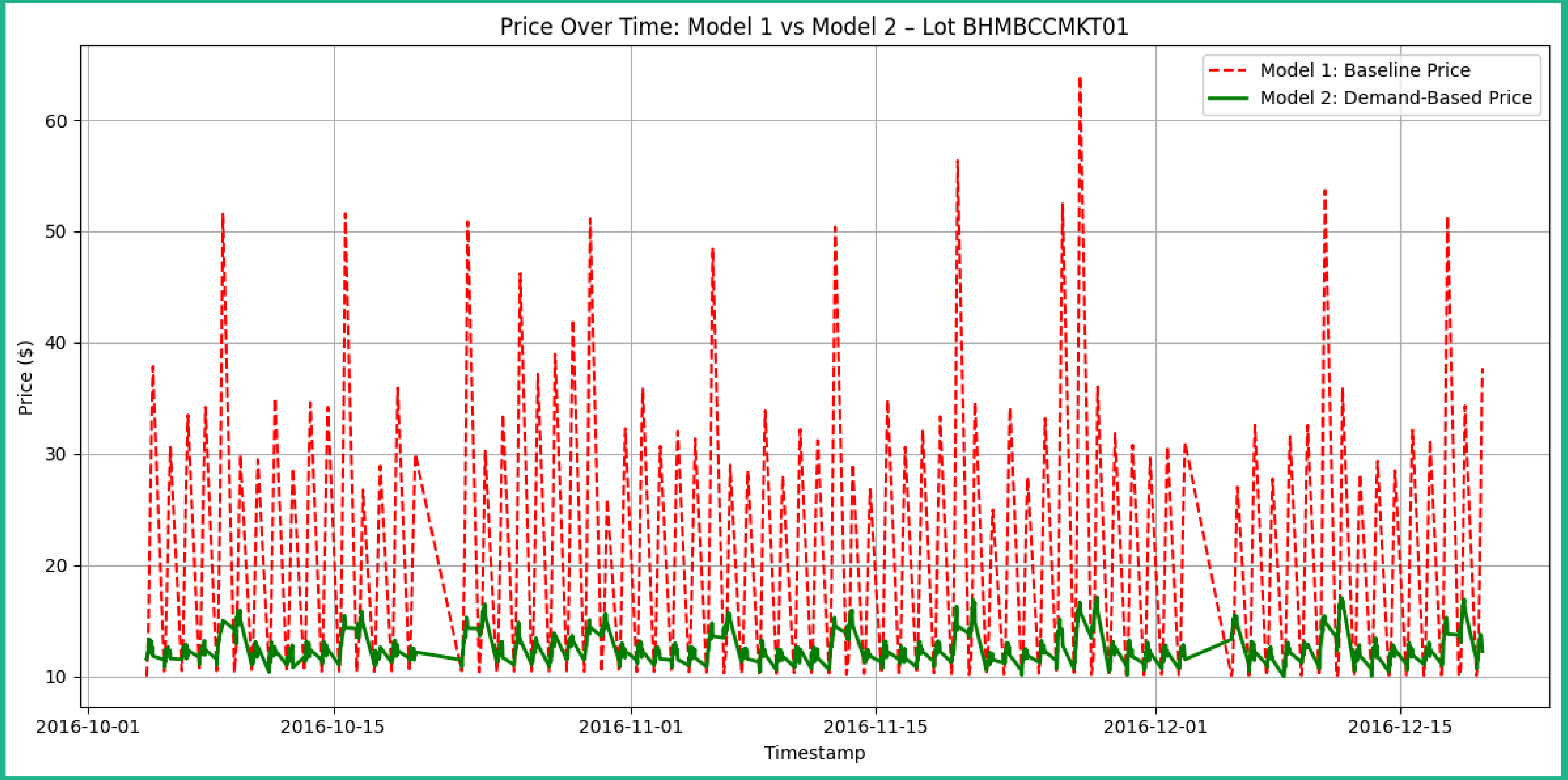


Shows price fluctuations over time for different lots.



- The normalized demand is bell-shaped with mean around 0.45–0.5.
- Confirms the demand values are well normalized (0–1), making lambda tuning more predictable.

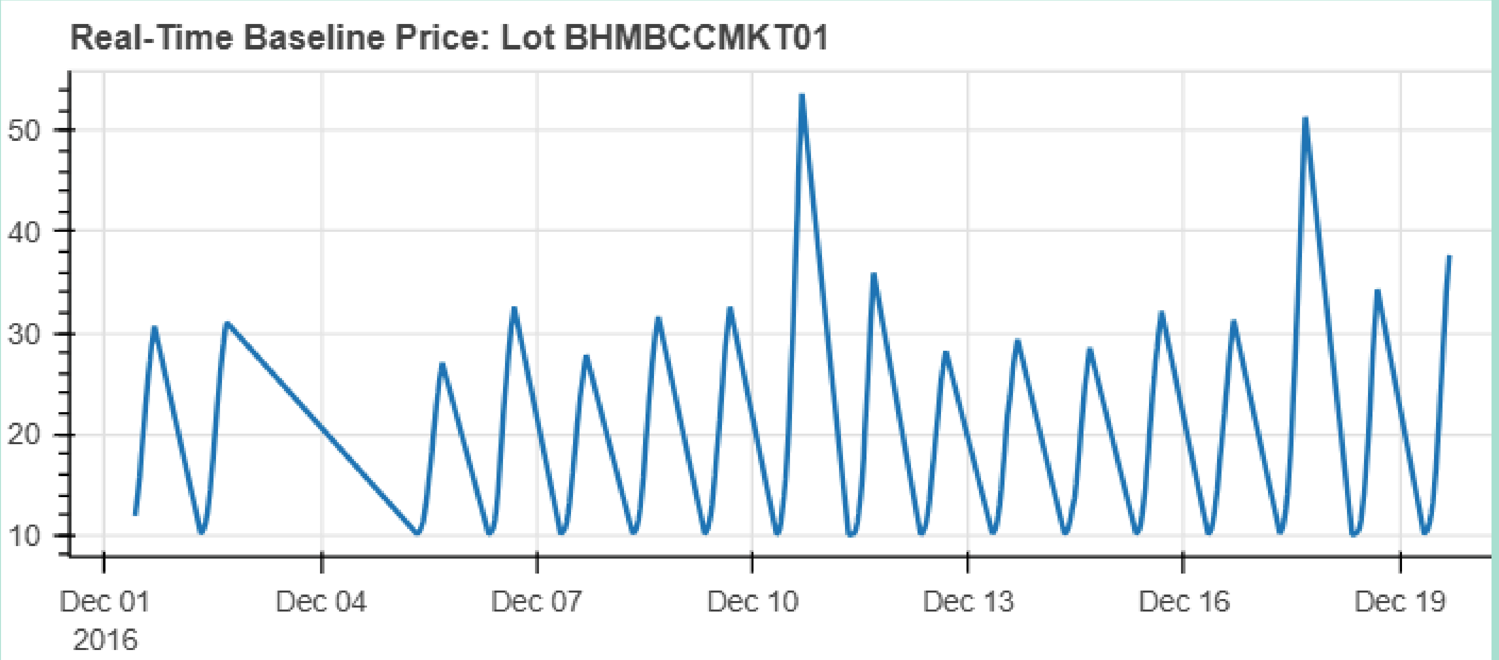
Comparison of the two models



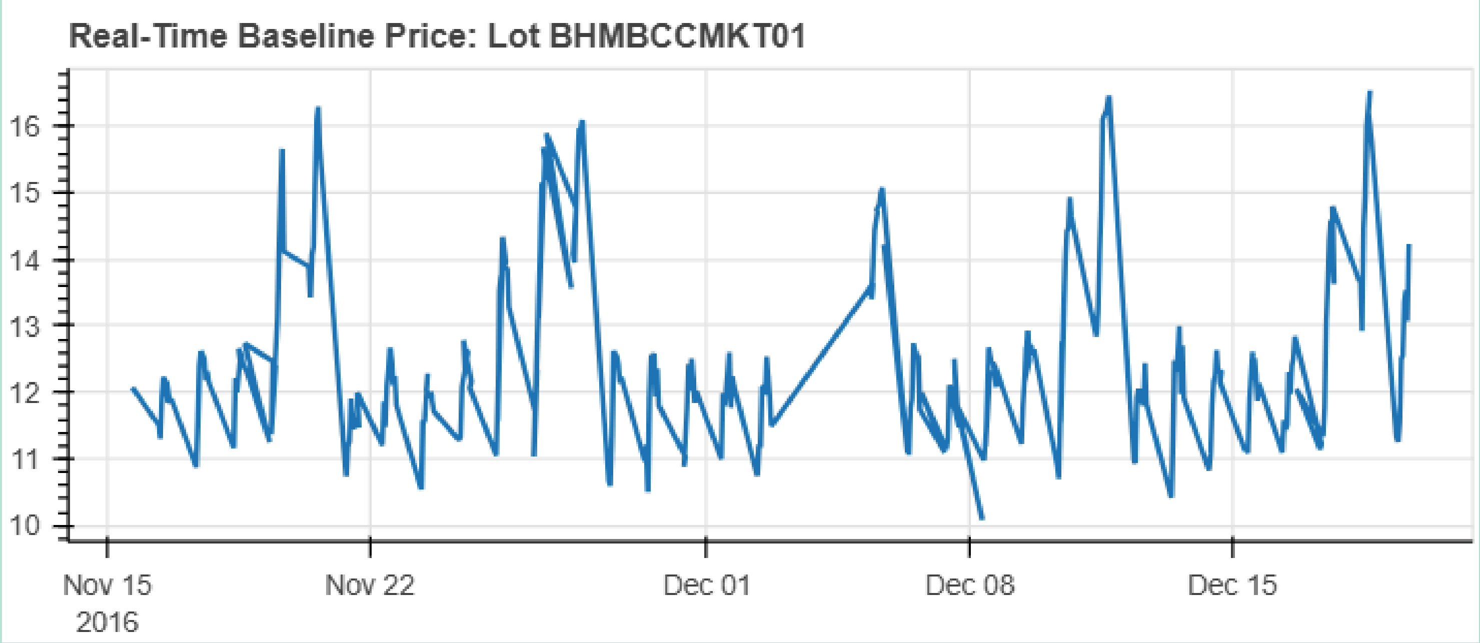

```
Volatility Comparison:
Model 1: 4.9474156197003225
Model 2: 0.5302923519451958
```

Model 1 may lead to unrealistically high prices, especially during long high-occupancy periods, while Model 2 adjusts pricing based on actual context-aware demand, not just occupancy.

Model 1 vs Model 2 , Price vs Timestamp graph for LotId: BHMBCCMKT01



Model 1 is suitable when the pricing needs to be easily interpretable and stable, for example in regular office hours or fixed event slots.



Model 2 would be better when real-time efficiency and profit optimization matter, but may need filters or smoothing to make it user-friendly and avoid frequent price jumps.

Thank you very much!