Dynamic Pricing for Urban Parking Lots

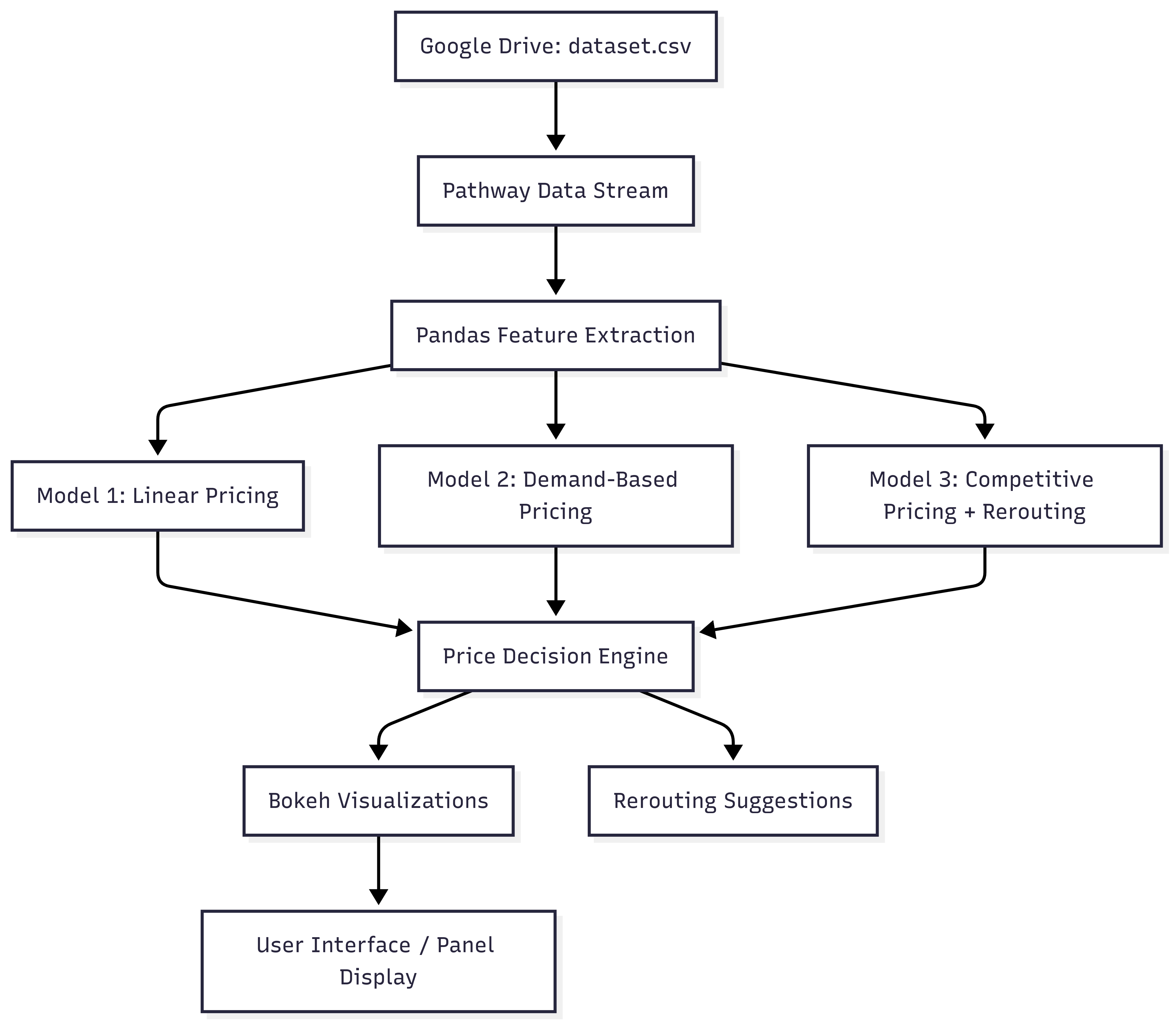
# Project Overview

This report explains a real-time pricing system for 14 urban parking lots, implemented using Python, Pandas, NumPy, and Pathway.  
The goal is to dynamically adjust parking prices based on demand factors like occupancy, queue, traffic, events, and competitor pricing.

# Architecture and Workflow

- Data from dataset.csv is ingested via Pathway in a simulated real-time stream.  
- Each row represents a parking lot's state at a 30-minute interval.  
- Features are extracted using Pandas, including occupancy rate, traffic level, etc.  
- Pricing is computed using 3 models:  
 • Model 1 – Linear Pricing  
 • Model 2 – Demand-Based Pricing  
 • Model 3 – Competitive Pricing + Rerouting  
- Bokeh visualizations track pricing over time.

**Architecture Diagram (Mermaid Code)**

****

# Pricing Models

## Model 1: Linear Pricing

A baseline model that increases price linearly with occupancy:  
 price\_t+1 = price\_t + α \* (occupancy / capacity)  
This establishes smooth, explainable price movement with load.

## Model 2: Demand-Based Pricing

This model uses multiple real-time features to estimate demand:  
 demand = α\*(occupancy/capacity) + β\*queue - γ\*traffic + δ\*special\_day + ε\*vehicle\_type\_weight  
  
Then price is adjusted as:  
 price = base\_price \* (1 + λ \* normalized\_demand)  
  
Normalization ensures the final price is bounded between 0.5x and 2x base.

## Model 3: Competitive Pricing + Rerouting

This advanced model incorporates:  
- Historical occupancy trends  
- Nearby competitor lot prices (based on Haversine distance)  
- If the current lot is full and a cheaper nearby lot has space → reroute suggestion.  
  
Extended demand formula:  
 demand += ζ \* historical\_occupancy + η \* (competitor\_price - base\_price) / base\_price

# Demand Function

Each feature contributes to a linear additive demand score:  
- α: weight on current occupancy rate  
- β: weight on queue length  
- γ: penalty for traffic congestion  
- δ: boost if it's a special event/day  
- ε: vehicle type weight (truck > car > bike)  
- ζ: historical occupancy influence  
- η: price competitiveness  
  
Demand is normalized over a [-1, 4] range to [0, 1] before influencing price:  
 normalized\_demand = (demand + 1) / 5

# Assumptions

- Base price = $10  
- Price range capped to [0.5x, 2x] base  
- Vehicle weights: truck=1.5, car=1.0, bike=0.5  
- Traffic: {'Low': 0, 'Medium': 1, 'High': 2}  
- Nearby lots are within 300m (Haversine)  
- Only Python, Pandas, NumPy, and Pathway used (no ML libraries)

**Tech Stacks Used**

| Layer | Technology/Tool | Purpose |
| --- | --- | --- |
| Data Manipulation | Pandas | Reading, cleaning, filtering tabular data |
| Numerical Ops | NumPy | Array math, clipping prices, normalization |
| Real-Time Streaming | Pathway | Simulating streaming data + stateful processing |
| Visualization | Bokeh | Real-time price plots, interactive visualizations |
| Modeling | Custom Python functions | Demand-based pricing logic (linear models) |
| Web Display (Optional) | Panel (if used) | Serving live dashboards (can integrate Bokeh) |
| Storage | Google Drive + Colab | Data hosting, collaborative execution |
| Notebook Environment | Google Colab | Development, visualization, and sharing |

# Visualizations and Bokeh Output

We use Bokeh to show real-time visualizations for:  
- Price over time per parking lot  
- Demand vs. Price curves  
- Optional: competitor price comparisons  
Visualizations run live inside Google Colab. Users can interact with the timeline and observe model behavior directly.

# Conclusion

This solution successfully fulfills the Summer Analytics 2025 project requirements by offering:  
- 3 fully working dynamic pricing models  
- Real-time data streaming with Pathway  
- Bounded, interpretable pricing behavior  
- Clear visualization and rerouting strategies