We first determine the objectives of the dynamic pricing model and which of the factors from the data we collected in the survey can influence pricing.

Objectives of the dynamic pricing model

* Maximize revenue
* Increase ticket sales during non-peak
* Offer discounts to specific customer segments

Data collected from survey that impacts pricing

* Time of visit (e.g., weekends, peak seasons)
* Customer demographics (e.g., tourists vs. locals)
* Customer satisfaction scores (e.g., willingness to pay based on satisfaction)
* Previous spending behaviors (e.g., purchasing souvenirs, food)

Next, we identify variables that might correlate with willingness to pay or visitation patterns. We then segment the data based on categories.

Variables that might correlate with willingness to pay or visitation patterns

* "Likelihood of Return,"
* "USS experience rating [Crowdedness]”,
* “'USS experience rating [Variety of Rides and Attractions]”,
* 'USS experience rating [Special Events and Performances (E.g. Halloween Horror Night)]'

Segment the Data based on category

Create customer segments based on key characteristics

* Tourists vs. locals
* High vs. low satisfaction scores
* Age groups

Identify Patterns: Analyze the data to understand how different segments vary in their willingness to pay, timing of visits, and satisfaction.

EDA

Univariate analysis

1. Distribution of satisfaction score
2. Distribution of Tourist vs. Local Visitors
3. Distribution of Age Groups
4. Average satisfaction score grouped by Tourist/Local
5. Average satisfaction scores grouped by age group

Bivariate analysis

1. correlation matrix between variables that might correlate with willingness to pay and the ranges visitors are willing to pay for peak and non peak hours
2. Heatmap for Satisfaction Score vs. Likelihood of Return (by age group)
3. Heatmap for Key Factors Affecting Satisfaction

Trends: Are there notable trends in satisfaction scores across different visitor types or visit days?

Relationships: Which factors correlate most with satisfaction? Are there clear drivers of satisfaction that could inform a dynamic pricing model?

Distributions: Are there any patterns in spending behavior or likelihood of return based on demographics or satisfaction levels?

step 3:

A dynamic pricing model adjusts prices based on factors like demand, time, and customer segment. For your prototype, you could use a rule-based or a simple machine learning approach.

(OUR FIRST OPTION) Rule-Based Pricing Model

Define a set of rules based on observed trends:

Peak vs. Off-Peak Pricing: Charge higher prices during high-demand times (e.g., weekends, holidays) and lower prices during weekdays or low-demand periods.

Segment-Based Discounts: Offer discounted prices to customer segments that might be more sensitive to price (e.g., locals or frequent visitors).

Satisfaction-Based Adjustments: If satisfaction is linked with spending, offer perks or discounts to guests with lower satisfaction scores to increase likelihood of return.

(OUR SECOND OPTION). Machine Learning Model - PREFERABLE OPTION

Regression Model: Use linear regression or a tree-based model (e.g., Decision Tree, Random Forest) to predict optimal ticket prices based on the input variables in your dataset.

Input Variables: Include features such as visit timing, visitor type, satisfaction metrics, and prior spending data.

Target Variable: Define a target variable for the model, such as “willingness to pay” if it’s present, or approximate it based on spending data or responses to pricing-related questions in the survey.

RULES BASED

import pandas as pd

# Load the dataset

data = pd.read\_excel('/path\_to\_your\_file/Survey\_cleaned\_balanced.xlsx')

# Define basic dynamic pricing rules

def dynamic\_pricing(row):

base\_price = 100 # Starting price in SGD

# Adjust price based on visit timing

if row['Visit Day'] in ['Saturday', 'Sunday']:

price = base\_price \* 1.2 # 20% increase on weekends

else:

price = base\_price # Regular price on weekdays

# Segment-based discounts

if row['Tourist/Local'] == 'Local':

price \*= 0.9 # 10% discount for locals

# Satisfaction-based adjustments

if row['Satisfaction Score'] < 3:

price \*= 0.8 # 20% discount for low satisfaction scores

return price

# Apply dynamic pricing function to each row in your data

data['Dynamic Price'] = data.apply(dynamic\_pricing, axis=1)

# Check results

print(data[['Tourist/Local', 'Visit Day', 'Satisfaction Score', 'Dynamic Price']])

MACHINE LEARNING

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# Select features and target

features = data[['Visit Day', 'Tourist/Local', 'Satisfaction Score']] # Replace with relevant columns

target = data['Willingness to Pay'] # Assume this or a similar target exists

# Convert categorical variables to numerical

features = pd.get\_dummies(features)

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.2, random\_state=42)

# Train model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Make predictions and evaluate

predictions = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, predictions)

print(f"Mean Squared Error: {mse}")

# Add predictions as dynamic prices

data['Dynamic Price ML'] = model.predict(pd.get\_dummies(data[['Visit Day', 'Tourist/Local', 'Satisfaction Score']]))

Evaluate Performance: Use metrics like Mean Squared Error (MSE) or Mean Absolute Error (MAE) to evaluate the accuracy of the machine learning model.

Refine Rules or Model: Based on initial results, adjust rules in the rule-based model or tune hyperparameters in the machine learning model.

A/B Testing (Optional): If possible, test the dynamic pricing model in a live environment with a subset of users to validate its impact on revenue or customer satisfaction.

Price Distribution: Plot a histogram or box plot to show the distribution of dynamic prices.

Revenue Forecast: Estimate potential revenue based on projected attendance and dynamic prices.

Customer Segment Analysis: Use a bar plot or scatter plot to show pricing differences across segments (e.g., tourists vs. locals).

import numpy as np

import pandas as pd

# Sample dataset of visitors with different characteristics

simulated\_data = pd.DataFrame({

'Visit Day': np.random.choice(['Weekday', 'Weekend'], size=1000),

'Tourist/Local': np.random.choice(['Tourist', 'Local'], size=1000),

'Satisfaction Score': np.random.randint(1, 6, size=1000) # Satisfaction from 1 to 5

})

# Define a function to apply pricing to simulated data

def run\_simulation(data, base\_price=100):

data['Dynamic Price'] = data.apply(dynamic\_pricing, axis=1)

total\_revenue = data['Dynamic Price'].sum()

avg\_price = data['Dynamic Price'].mean()

print(f"Total Revenue: ${total\_revenue:.2f}")

print(f"Average Price per Visitor: ${avg\_price:.2f}")

return data, total\_revenue, avg\_price

# Run the simulation for a standard scenario

simulated\_data, total\_revenue, avg\_price = run\_simulation(simulated\_data)

from sklearn.linear\_model import LinearRegression

# Generate simulated data with specific characteristics

X\_simulated = pd.DataFrame({

'Visit Day': np.random.choice(['Weekday', 'Weekend'], size=1000),

'Tourist/Local': np.random.choice(['Tourist', 'Local'], size=1000),

'Satisfaction Score': np.random.randint(1, 6, size=1000)

})

# Convert categorical data to numerical

X\_simulated = pd.get\_dummies(X\_simulated)

# Predict prices using the trained model

predicted\_prices = model.predict(X\_simulated)

X\_simulated['Predicted Price'] = predicted\_prices

# Calculate total and average revenue for this simulation

total\_revenue\_ml = X\_simulated['Predicted Price'].sum()

avg\_price\_ml = X\_simulated['Predicted Price'].mean()

print(f"Total Revenue (ML): ${total\_revenue\_ml:.2f}")

print(f"Average Price per Visitor (ML): ${avg\_price\_ml:.2f}")

CHECK IF THE SIMULATIONS ARE USABLE

You can automate the simulation by defining different scenarios in a loop or creating different parameter combinations. For example, you might loop through different Visit Day values, varying proportions of tourists and locals, or different satisfaction levels:

# Define scenarios for peak/off-peak and tourist/local mix

scenarios = [

{'Visit Day': 'Weekend', 'Tourist/Local': 'Tourist', 'Satisfaction Score': 5},

{'Visit Day': 'Weekday', 'Tourist/Local': 'Local', 'Satisfaction Score': 3},

# Add more scenarios as needed

]

# Run each scenario

for scenario in scenarios:

scenario\_data = pd.DataFrame([scenario] \* 100) # 100 visitors in each scenario

\_, revenue, avg\_price = run\_simulation(scenario\_data)

print(f"Scenario {scenario} - Revenue: ${revenue:.2f}, Average Price: ${avg\_price:.2f}")

visualisation of simulation results:

import matplotlib.pyplot as plt

# Example: Plot total revenue across different scenarios

scenarios = ["Weekend - Tourists", "Weekday - Locals", ...]

revenues = [2000, 1500, ...] # Replace with revenue results from each scenario

plt.figure(figsize=(10, 6))

plt.bar(scenarios, revenues)

plt.title("Revenue across Different Scenarios")

plt.xlabel("Scenarios")

plt.ylabel("Total Revenue ($)")

plt.show()