**Steps**

**1. Data Collection:** Collect data from the company

**2. Data Preprocessing**

- **Cleaning:** Handle missing values, incorrect data entries, and outliers.

- **Feature Engineering:** Create new features that might influence demand, such as weather conditions, special events, or holidays.

- **Normalization:** Standardize data for better comparison and modeling.

**~~3. Cost Modelling~~**

~~-~~ **~~Fixed Costs:~~** ~~Include costs such as vehicle maintenance, insurance, and administrative expenses.~~

~~-~~ **~~Variable Costs:~~** ~~Include fuel, electricity, and other usage-based expenses.~~

~~-~~ **~~Tools:~~** ~~Use Pandas to calculate and aggregate costs.~~

**~~4. Competitor Analysis~~**

~~-~~ **~~Data Collection:~~** ~~Scrape or collect competitor pricing data.~~

~~-~~ **~~Benchmarking:~~** ~~Compare Co Wheels' pricing with competitors using Pandas and visualization libraries like Matplotlib or Seaborn.~~

**5. Dynamic Pricing Tool Development**

- **Input Parameters:** Allow inputs for demand factors data.

- **Modelling:** Use machine learning algorithms to predict demand based on pricing. *Libraries like Scikit-learn or TensorFlow can be useful here.*

- **Optimisation:** Implement optimisation techniques to find the best pricing strategy. Consider *libraries like SciPy for optimisation tasks*.

**6. Scenario Testing and Validation**

- **Simulation:** Create different pricing scenarios and simulate their outcomes using historical data.

- **Analysis:** Assess the impact on profitability and vehicle utilisation using Pandas and visualization tools.

- **Validation:** Compare simulated results with actual historical data to validate the model's accuracy.

**7. Implementation Strategy**

- **Integration:** Develop an API or web interface for the pricing tool.

- **Monitoring:** Set up real-time monitoring of key metrics to adjust pricing dynamically.

**8. Evaluation and Reporting**

- **Performance Metrics:** Track metrics like revenue, profit margins, utilization rates, and customer satisfaction.

- **Visualisation:** Use Matplotlib, Seaborn, or Plotly for creating visual reports.

- **Reporting:** Generate detailed reports using libraries like ReportLab or exporting to formats like PDF or Excel.

**Sample Python Workflow**

Here's a brief example of how you might start implementing this in Python:

```python

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# Load data

trip\_data = pd.read\_csv('trip\_data.csv')

cost\_data = pd.read\_csv('cost\_data.csv')

competitor\_data = pd.read\_csv('competitor\_data.csv')

# Data preprocessing

trip\_data['timestamp'] = pd.to\_datetime(trip\_data['timestamp'])

trip\_data = trip\_data.dropna()

# Feature engineering

trip\_data['hour'] = trip\_data['timestamp'].dt.hour

trip\_data['day\_of\_week'] = trip\_data['timestamp'].dt.dayofweek

# Merge cost data

merged\_data = pd.merge(trip\_data, cost\_data, on='vehicle\_id')

# Train-test split

X = merged\_data[['hour', 'day\_of\_week', 'distance', 'duration']]

y = merged\_data['price']

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

# Model training

model = LinearRegression()

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

# Prediction and validation

y\_pred = model.predict(X\_test)

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

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

# Scenario testing

def simulate\_pricing\_scenario(data, new\_prices):

data['simulated\_price'] = data.apply(lambda row: new\_prices.get(row['hour'], row['price']), axis=1)

return data

new\_prices = {hour: price for hour, price in zip(range(24), np.linspace(5, 15, 24))}

simulated\_data = simulate\_pricing\_scenario(X\_test.copy(), new\_prices)

profit\_impact = simulated\_data['simulated\_price'].sum() - y\_test.sum()

print(f'Profit Impact: {profit\_impact}')

# Visualization

plt.plot(y\_test.values, label='Actual')

plt.plot(y\_pred, label='Predicted')

plt.legend()

plt.show()

```