**PUBLIC TRANSPORTATION EFFICIENCY**

**DEVELOPMENT PART 2**

**Data Collection:** To analyze public transportation efficiency, you need to gather data from various sources. This data can include schedules, real-time GPS tracking, fare collection, and ridership statistics.

1. **Data Cleaning:** Raw data may contain errors, inconsistencies, and outliers. Data cleaning involves removing or correcting such issues to ensure data quality.
2. **Data Integration:** Different sources of data need to be integrated into a unified dataset for comprehensive analysis.
3. **Data Transformation:** You may need to create derived variables or features, such as travel time, on-time performance, and passenger load, to gain more insights.
4. **Descriptive Analytics:** This involves summarizing and visualizing data to identify trends, patterns, and anomalies. Tools like data visualization libraries (e.g., Matplotlib, Seaborn, or Tableau) can be very useful.
5. **Predictive Analytics:** Use predictive modeling techniques, such as regression, time series analysis, or machine learning algorithms, to forecast future trends like ridership or on-time performance. Predictive analytics can help in proactive decision-making.
6. **Prescriptive Analytics:** Suggest actions based on predictive insights. For instance, optimizing bus routes, adjusting schedules, or increasing staff during peak hours.

**Key Performance Indicators (KPIs) for Public Transportation Efficiency:**

1. **On-Time Performance:** Measure how well public transportation adheres to its schedules. This is typically expressed as a percentage of on-time arrivals and departures.
2. **Ridership Metrics:** Analyze the number of passengers, average passenger load, and peak hours. These metrics can help in optimizing services.
3. **Revenue and Cost Analysis:** Assess the revenue generated by public transportation compared to its operational and maintenance costs. Determine the cost per passenger or per mile.
4. **Customer Satisfaction:** Use surveys or social media sentiment analysis to gauge passenger satisfaction. Higher satisfaction can indicate better efficiency.
5. **Service Reliability:** Evaluate the consistency of service quality, including factors like waiting times, frequency, and route reliability.
6. **Accessibility:** Analyze the accessibility of public transportation to various demographics, including elderly and disabled individuals.
7. **Environmental Impact:** Consider the environmental footprint of public transportation, including emissions and fuel efficiency.
8. **Accident and Safety Metrics:** Examine accident rates and safety records to ensure the well-being of passengers.

**Tools for Public Transportation Efficiency Analytics:**

1. **Data Analysis Tools:** Software like Python with libraries like pandas, NumPy, and scikit-learn, or R, can be used for data analysis, cleaning, and modeling.
2. **Geospatial Analysis Tools:** Geographic Information System (GIS) software such as ArcGIS or QGIS is valuable for spatial analysis, route optimization, and geospatial visualization.
3. **Real-time Data Processing:** Apache Kafka and Apache Spark can be used for handling and processing real-time data, such as GPS coordinates and sensor data.
4. **Dashboard and Visualization Tools:** Tools like Tableau, Power BI, or custom web applications can be employed for creating interactive dashboards to monitor KPIs and trends.
5. **Machine Learning and AI:** For predictive analytics, machine learning libraries like TensorFlow or scikit-learn can be beneficial.
6. **Transportation-specific Software:** Commercial software like Trapeze, Clever Devices, or Syncromatics provides transportation-specific analytics and fleet management solutions.
7. **Database Management Systems:** Efficient data storage and retrieval are crucial. Databases like PostgreSQL, MySQL, or NoSQL databases can be used.

**Step 1: Import necessary libraries**

pythonCopy code

import pandas as pd import matplotlib.pyplot as plt import seaborn as sns

**Step 2: Load your transportation data**

pythonCopy code

# Assuming you have a CSV file with your transportation data data = pd.read\_csv('transportation\_data.csv')

**Step 3: Data Preprocessing** This step involves cleaning and preparing the data for analysis. Common tasks include:

* Handling missing values.
* Removing duplicates.
* Data type conversion.
* Renaming columns for clarity.
* Filtering or selecting relevant columns.

Example:

# Remove rows with licates() # Convert date columns to datetime objects data['date'] = pd.to\_datetime(data['date']) # Rename columns for clarity data = data.rename(columns={'bus\_speed': 'Bus Speed', 'passenger\_count': 'Passenger Count'})

**Step 4: Exploratory Data Analysis (EDA)** EDA helps you understand your data better. You can perform tasks like summary statistics, data visualization, and correlations.

Example:

pythonCopy code

# Summary statistics summary = data.describe() # Data visualization sns.histplot(data['Bus Speed'], kde=True) plt.xlabel('Bus Speed') plt.ylabel('Frequency') plt.title('Distribution of Bus Speed') plt.show() sns.scatterplot(data=data, x='Bus Speed', y='Passenger Count') plt.xlabel('Bus Speed') plt.ylabel('Passenger Count') plt.title('Bus Speed vs Passenger Count') plt.show()

**Step 5: Analyze Transportation Efficiency** To analyze transportation efficiency, you can calculate various metrics or perform regression analysis. For instance, you can calculate average bus speed, passenger counts, or perform regression to understand the relationship between speed and passenger count.

Example:

pythonCopy code

# Calculate average bus speed average\_speed = data['Bus Speed'].mean() # Calculate average passenger count average\_passengers = data['Passenger Count'].mean() # Perform a regression analysis to understand the relationship between speed and passenger count import statsmodels.api as sm X = data['Bus Speed'] X = sm.add\_constant(X) y = data['Passenger Count'] model = sm.OLS(y, X).fit() print(model.summary())

**Step 6: Visualize Results** Visualizing your results can help in better understanding and communication of your findings.

Example:

# Visualize the regression line plt.scatter(data['Bus Speed'], data['Passenger Count']) plt.plot(data['Bus Speed'], model.predict(X), color='red', linewidth=2) plt.xlabel('Bus Speed') plt.ylabel('Passenger Count') plt.title('Regression Analysis') plt.show()

**Step 1: Import necessary libraries** Make sure to include the libraries we previously imported in part 1, and add any additional libraries as needed for advanced analysis and visualization.

import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression

**Step 2: Continue Data Preprocessing**

Depending on your data and specific goals, you may need to perform more advanced data preprocessing tasks, such as feature engineering, handling outliers, and scaling data.

**Step 3: Advanced Analysis**

In this step, you can perform more advanced statistical and machine learning analyses to uncover patterns and trends. Some examples include:

* Time series analysis: If your data has a time component, analyze it using time series techniques to identify trends and seasonality.
* Clustering: Group stations or routes based on similarities.
* Classification: Predict the likelihood of a delay or overcrowding.

**Step 4: Advanced Visualization**

To visualize complex relationships and patterns, you can use advanced plotting libraries and techniques.

Example of time series analysis:

# Time series plot of bus speed plt.figure(figsize=(12, 6)) sns.lineplot(data=data, x='Date', y='Bus Speed') plt.xlabel('Date') plt.ylabel('Bus Speed') plt.title('Bus Speed Over Time') plt.grid(True) plt.show()

**Step 5: Predictive Modeling**

If you want to predict future transportation efficiency or passenger counts, you can build predictive models. Linear regression is a simple example.

pythonCopy code

# Split the data into training and testing sets X = data[['Bus Speed']] y = data['Passenger Count'] X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) # Train a linear regression # Evaluate the model's performance from import2 = r2\_score(y\_test, y\_pred) print(f'Mean Squared Error: {mse}') print(f'R-squared: {r2}')

**Step 6: Interactive Visualizations**

To make your results more interactive, consider using libraries like Plotly to create interactive plots that allow users to explore the data.

Example:

pythonCopy code

import as data, x='Bus Speed', y='Passenger Count', title='Bus Speed vs. Passenger Count'size=5)) fig.sho()

**Step 7: Interpret Results and Recommendations**

Interpret the results of your analysis and use them to make recommendations for improving public transportation efficiency. For example, you can provide insights into how changes in bus speed affect passenger counts and suggest strategies to optimize routes or schedules.

Output:

Mean Squared Error: 142.58

R-squared: 0.74

PROGRAM:

import pandas as pd

import numpy as np

# Sample data for public transportation arrivals

data = {

'bus\_id': [1, 2, 3, 4, 5],

'scheduled\_time': ['08:00', '08:15', '08:30', '08:45', '09:00'],

'actual\_time': ['08:02', '08:20', '08:40', '08:46', '09:10']

}

# Create a DataFrame from the sample data

df = pd.DataFrame(data)

# Convert time columns to datetime

df['scheduled\_time'] = pd.to\_datetime(df['scheduled\_time'])

df['actual\_time'] = pd.to\_datetime(df['actual\_time'])

# Calculate time deviation (positive for delays, negative for early arrivals)

df['time\_deviation'] = (df['actual\_time'] - df['scheduled\_time']).dt.total\_seconds() / 60

# Calculate on-time performance (percentage of on-time arrivals)

on\_time\_arrivals = df[df['time\_deviation'].between(-5, 5)] # Assume a 5-minute threshold

on\_time\_performance = len(on\_time\_arrivals) / len(df) \* 100

print("On-Time Performance: {:.2f}%".format(on\_time\_performance))

output:

On-Time Performance: 60.00%

