# Innovation:

Public transportation efficiency analysis

Certainly! When applying data analytics in public transportation efficiency analysis using Cognizant, several key aspects can be considered as data analysis. By analyzing passenger boarding summaries, routes, trips, stops, buses, and trains, valuable insights can be gained. This data can inform optimization strategies, improve route planning, and enhance overall service quality. Cognizant's analytics tools can help identify patterns, optimize schedules, and contribute to innovative solutions for public transport systems.



# Public transportation efficiency analysis

**Passenger Boarding Summaries:**

Demographic Tailoring: Cognizant's analytics can analyze passenger data, helping transportation authorities understand the demographics of their riders. This information allows for tailoring services to specific groups, such as adjusting schedules or providing targeted amenities.

# Boarding Time Analysis:

By delving into historical boarding data, Cognizant's analytics can identify peak boarding times. This insight enables optimized staffing during busy periods, ensuring a smoother boarding process.

# Routes Optimization:

Machine Learning Route Analysis: Leveraging machine learning algorithms, Cognizant can analyze historical traffic data to optimize bus and train routes. This includes identifying congestion patterns, traffic hotspots, and suggesting alternate routes for more efficient travel.

# Dynamic Route Adjustments:

Real-time analytics can trigger dynamic route adjustments based on live traffic conditions, minimizing delays and improving overall route efficiency**.**

# Trips and Schedules:

Predictive Analytics for Delays: Predictive analytics can forecast potential delays by analyzing historical data and external factors. This allows for proactive adjustments to schedules, reducing disruptions to passenger journeys**.**

# Optimization Considering External Factors:

Cognizant's analytics considers external factors like weather, events, and holidays to optimize schedules, ensuring a robust and adaptable transportation system.

# Stops and Stations:

RFID Tracking for Passenger Counts: Implementing RFID or similar tracking technologies at stops provides precise passenger count data. This data is invaluable for optimizing staffing, predicting demand, and ensuring the efficient deployment of resources.

# Optimized Stop Placement:

Analytics can analyze passenger flow data to optimize the placement of stops. This minimizes travel time, enhances efficiency, and improves the overall transit experience.

# Bus and Train Utilization:

Predictive Maintenance Models: Cognizant's analytics can develop predictive maintenance models, enabling proactive scheduling of maintenance activities. This reduces downtime, improves reliability, and ensures vehicles are in optimal condition.

# Optimized Fleet Allocation:

By analyzing demand patterns and vehicle performance, analytics helps optimize the allocation of buses and trains, ensuring resources are deployed where they are needed most.

# Data Integration:

Centralized Data Platform: Cognizant's approach involves creating a centralized data platform that integrates data from various sources. This unified view provides a comprehensive understanding of the entire transportation system.

# Data Governance Practices:

Implementing robust data governance practices ensures data quality and consistency, critical for making informed decisions based on accurate information.

# Predictive Analytics:

Ridership Prediction Models: Develop predictive models that forecast future ridership based on historical data and external factors. This enables transportation authorities to proactively plan for changing demand.

# Proactive Decision-Making:

Utilize predictive analytics for proactive decision-making in resource allocation, ensuring that the system can adapt to evolving transportation needs.

# Innovation and Technology:

IoT for Real-Time Tracking: Explore the use of IoT devices for real-time tracking of vehicles and infrastructure. This technology provides up-to- the-minute data crucial for real-time decision-making.

# AI-Driven Chatbots:

Integrating AI-driven chatbots enhances passenger experience by providing real-time information, addressing queries, and resolving issues promptly.

# Customer Experience:

Sentiment Analysis: Cognizant's analytics can include sentiment analysis of social media and customer feedback. This ongoing analysis ensures continuous improvement by identifying areas that impact passenger satisfaction.

# Mobile Ticketing and Apps:

Implementing features like mobile ticketing and journey planning apps enhances the overall passenger experience, providing convenience and accessibility.

# Regulatory Compliance:

Compliance Dashboards: Develop dashboards that monitor and report regulatory adherence, ensuring that the public transportation system complies with all relevant regulations.

# Data Security and Privacy Compliance:

Cognizant ensures that data security and privacy compliance are prioritized, especially when dealing with sensitive passenger data, safeguarding against unauthorized access and ensuring public trust.

By incorporating these detailed strategies, Cognizant's data analytics can contribute to a comprehensive and effective framework for enhancing public transportation efficiency, providing actionable insights for continuous improvement and innovation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TripID** | **RouteID** | **StopID** | **StopName** | **WeekBeginning** | **Number of**  **boardings** |
| 23631 | 100 | 14156 | 181 Cross Rd | 30-06-2013  00:00 | 1 |
| 23631 | 100 | 14144 | 177 Cross Rd | 30-06-2013  00:00 | 1 |
| 23632 | 100 | 14132 | 175 Cross Rd | 30-06-2013  00:00 | 1 |
| 23633 | 100 | 12266 | Zone A  Arndale Interchange | 30-06-2013  00:00 | 2 |
| 23633 | 100 | 14147 | 178 Cross Rd | 30-06-2013  00:00 | 1 |

# Sample output:

**TripID RouteID StopID NumberOfBoardings Cluster**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **23631** | **100** | | **14156** | **1** | | **1** | |
| **1** | **23631** | **100** | | **14144** | **1** | | **1** | |
| **2** | **23632** | **100** | | **14132** | **1** | | **1** | |
| **3** | **23633** | **100** | | **12266** | **2** | | **1** | |
| **4** | **23633** | **100** | | **14147** | **1** | | **1** | |
| **...** | **...** | **...** | **...** | **...** | | **...** | | |
| **4931** | **44681** | | **100** | **12352** | | **1** | | **0** |
| **4932** | **44681** | | **100** | **14000** | | **7** | | **0** |
| **4933** | **44681** | | **100** | **14110** | | **5** | | **0** |
| **4934** | **44681** | | **100** | **14167** | | **5** | | **0** |
| **4935** | **44681** | | **100** | **12663** | | **2** | | **0** |

# [4936 rows x 5 columns]

Program:

# Import necessary libraries import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler from sklearn.impute import SimpleImputer

from sklearn.pipeline import Pipeline

from sklearn.compose import ColumnTransformer from sklearn.preprocessing import OneHotEncoder

# Load your dataset

# Replace 'your\_dataset.csv' with the actual filename data = pd.read\_csv('your\_dataset.csv')

# Assuming you have features for clustering, excluding the 'ServiceDisruption' column

X = data[['TripID', 'RouteID', 'StopID', 'NumberOfBoardings']]

# Define preprocessing steps for numeric and categorical features numeric\_features = ['TripID', 'RouteID', 'StopID', 'Number OfBoardings'] numeric\_transformer = Pipeline(steps=[

('imputer', SimpleImputer(strategy='mean')), # Handle missing values with mean imputation

('scaler', StandardScaler()) # Standardize numeric features

])

categorical\_features = [] # Add categorical features if any categorical\_transformer = Pipeline(steps=[

('imputer', SimpleImputer(strategy='most\_frequent')), # Handle missing values with most frequent imputation

('onehot', OneHotEncoder(handle\_unknown='ignore')) # One-hot encode categorical features

])

# Combine preprocessing steps preprocessor = ColumnTransformer(

transformers=[

('num', numeric\_transformer, numeric\_features), ('cat', categorical\_transformer, categorical\_features)

])

# Create a pipeline with preprocessing and the K-means clustering algorithm

pipeline = Pipeline(steps=[('preprocessor', preprocessor),

('kmeans', KMeans(n\_clusters=2, random\_state=42))]) # You can adjust the number of clusters

# Fit the model and get cluster assignments data['Cluster'] = pipeline.fit\_predict(X)

# View the resulting clusters

print(data[['TripID', 'RouteID', 'StopID', 'Number OfBoardings', 'Cluster']])

Github:

[https://github.com/StanlySK/DAC\_Phase1/blob/43d1549fef65045f34c9b44730a5b4b55c66b00c/DAC\_Phase2 coding](https://github.com/StanlySK/DAC_Phase1/blob/43d1549fef65045f34c9b44730a5b4b55c66b00c/DAC_Phase2%20coding)