**PUBLIC TRANSPORT EFFICIENCY ANALYSIS**

**TEAM MEMBER: ANUPRIYA D**

**registration no: 412621243009**

****

**PROJECT DEFINITION:**

The project involves analyzing public transportation data to assess service efficiency, on time performance, and passenger feedback. The objective is to provide insights that support transportation improvement initiatives and enhance the overall public transportation experience. This project includes defining analysis objectives, collecting transportation data, designing relevant visualizations in IBM Cognos, and using code for data analysis

**DESIGN THINKING:**

**Analysis objectives:**

* **On-Time Performance Assessment**

- Objective: Evaluate punctuality, reduce delays.

- Metrics: On-time trips, average delay time.

- Actions: Identify issues, improve schedules, manage traffic.

* **Passenger Satisfaction Analysis**

- Objective: Enhance passenger experience.

- Metrics: Surveys, complaints, load factor.

- Actions: Analyze feedback, improve comfort, safety.

* **Service Efficiency Evaluation**

- Objective: Optimize operations and costs.

- Metrics: Load per trip, operational costs.

- Actions: Route optimization, efficient maintenance, sustainability.

* **Route Optimization and Expansion**

- Objective: Optimize routes, expand as needed.

Metrics: Ridership, demographics, cost-benefit.

- Actions: Restructure routes, serve underserved areas.

* **Safety and Security Enhancement**

- Objective: Ensure passenger/staff safety.

- Metrics: Incident reports, surveillance data.

- Actions: Improve security systems, conduct safety drills.

* **Fare Pricing and Revenue Analysis**

- Objective: Maximize revenue, affordability.

- Metrics: Fare revenue, fare evasion rates.

- Actions: Analyze fare data, prevent fare evasion, assess fare changes.

This summary provides a concise overview of the specific objectives for analyzing public transportation data related to on-time performance, passenger satisfaction, service efficiency, route optimization, safety, and fare pricing.

**Data Collection for Transportation Data Analysis**

Transportation data can be collected from various sources and using different methods to analyze schedules, real-time updates, and passenger feedback. Below are common sources and methods:

**1. Schedules Data:**

- Source: Transportation authorities, agencies, or operators typically provide schedule data. This data can often be accessed through their official websites or public data portals.

- Method: Downloading schedules from official websites or using APIs (Application Programming Interfaces) provided by transportation authorities.

**2. Real-Time Updates:**

- Source: Real-time data includes GPS signals from vehicles, sensors at transit stops, and information from traffic management systems.

- Method:

- GPS Data: Collect data from GPS devices installed on vehicles. This data can be accessed through APIs or onboard data recorders.

- Sensors: Access real-time data from sensors at transit stops or along transportation routes.

- Traffic Management Systems: Data from traffic management centers can be obtained through APIs or data feeds.

**3. Passenger Feedback:**

- Source: Passenger feedback can be collected through surveys, social media, mobile apps, or dedicated feedback channels provided by transportation agencies.

- Surveys: Conduct online or offline surveys to gather feedback on various aspects of public transportation.

- Social Media Monitoring: Use social media listening tools to track mentions and sentiments related to public transportation.

- Mobile Apps: If available, extract feedback data from official transportation mobile apps.

- Dedicated Feedback Channels: Access data from official websites or feedback portals.

**VISUALIZATION STRATEGIES**

**1. Data Preparation:**

- Import and clean the provided Kaggle dataset in IBM Cognos, ensuring data quality.

**2. Define Key Metrics and KPIs:**

- Identify relevant metrics and KPIs for your analysis objectives.

**3. Dashboard and Report Layout:**

- Design user-friendly layouts for dashboards and reports.

**4. Visualization Types:**

- Choose suitable chart types for your data (e.g., line charts, bar charts).

**5. Interactive Elements:**

- Add filters and slicers for user interactivity.

**6. Geospatial Visualizations:**

- Use maps to display transportation routes and locations.

**7. Data Storytelling:**

- Organize dashboards to tell a coherent data-driven story.

**8. Scheduled Updates:**

- Set up automatic data refreshes.

**9. Collaboration and Sharing:**

- Enable collaboration features and define sharing channels.

**10. Testing and Validation:**

- Test dashboards, gather feedback, and make adjustments.

**11. Documentation:**

- Document data sources and methodologies.

**12. Training and Support:**

- Provide training for end-users.

**13. Deployment:**

- Deploy dashboards and reports in a production environment.

**14. Continuous Improvement:**

- Regularly update dashboards based on changing data and needs.

This strategy outlines the steps to effectively use IBM Cognos to create informative and actionable visualizations from your transportation data. It ensures that your visualizations are accurate, user-friendly, and support data-driven decision-making.

**CODE INTEGRATION:**

Certainly, here's a brief summary of how code integration can enhance different aspects of the transportation data analysis using the provided Kaggle dataset:

**1. Data Cleaning:**

- Use code to clean and preprocess data, handle missing values, and format data consistently.

**2. Data Transformation:**

- Code assists in transforming data into suitable formats, aggregating, deriving variables, and normalizing.

**3. Exploratory Data Analysis (EDA):**

- Leverage code for EDA, generating descriptive statistics, visualizations, and conducting hypothesis testing.

**4. Advanced Analytics:**

- Implement code for statistical analysis, predictive modeling, and machine learning to gain deeper insights.

**5. Automation:**

- Automate data processing tasks with code to ensure consistency and efficiency.

**6. Integration with Real-Time Data:**

- Code can fetch and integrate real-time data into analysis for up-to-date insights.

**7. Dashboard and Report Generation:**

- Code can dynamically create dashboards and reports with updated data and visualizations.

**8. Data Export and Sharing:**

- Use code to export analysis results in various formats for sharing with stakeholders.

**9. Version Control:**

- Employ version control systems to manage code and ensure collaboration and reproducibility.

**10. Documentation:**

- Embed code comments and documentation within scripts to explain analysis steps.

**11. Scalability:**

- Code-based solutions can scale to handle larger datasets or complex analyses.

**12. Error Handling and Logging:**

- Implement code for error handling and logging to ensure smooth execution and troubleshooting.

**PROJECT CODE**

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import matplotlib.pyplot as plt

import datetime

import os

from math import sqrt

import warnings

data = pd.read\_csv('../input/unisys/ptsboardingsummary/20140711.CSV')

data.shape

data.head(10)

ut\_geo = pd.read\_csv('../input/outgeo/output\_geo.csv')

out\_geo.shape

out\_geo.head()

from math import sin, cos, sqrt, atan2, radians

def calc\_dist(lat1,lon1):

R = 6373.0

dlon = radians(138.604801) - radians(lon1)

dlat = radians(-34.921247) - radians(lat1)

a = sin(dlat / 2)\*\*2 + cos(radians(lat1)) \* cos(radians(-34.921247)) \* sin(dlon / 2)\*\*2

c = 2 \* atan2(sqrt(a), sqrt(1 - a))

return R \* c

out\_geo['dist\_from\_centre'] = out\_geo[['latitude','longitude']].apply(lambda x: calc\_dist(\*x), axis=1)

out\_geo.head()

out\_geo['type'].fillna('street\_address',inplace=True)

out\_geo['type'] = out\_geo['type'].apply(lambda x: str(x).split(',')[-1])

out\_geo['type'].unique()

data['WeekBeginning'] = pd.to\_datetime(data['WeekBeginning']).dt.date

data['WeekBeginning'][1]

data= pd.merge(data,out\_geo,how='left',left\_on = 'StopName',right\_on = 'input\_string')

data.head(5)

data.shape

plt.show()

R



