PUBLIC TRANSPORT OPTIMIZING USING IOT

**PHASE 3** Submission document

**Projrct Title** : Public Transport Optimization

**Phase 3** : Development part 1

**Topic :** Start building the public transport optimization model by loadind and pre-processing the data set.

**PUBLIC TRANSPORT OPTIMIZATION**

Introduction :

* Public transportation is a fundamental component of urban infrastructure, catering to the mobility needs of millions of people worldwide. However, it often faces challenges like inefficiency, congestion, and lack of real-time information.
* To address these issues and revolutionize public transportation, the integration of the Internet of Things (IoT) technology holds immense promise. This project focuses on harnessing the capabilities of IoT to enhance public transport systems by enabling real-time monitoring, passenger information access, route optimization, energy efficiency, and safety and security enhancements.
* **Through this initiative, we aim to create a more efficient, sustainable, and user-centric public transport experience, benefiting both passengers and providers.**

**Necessary step to follow :**

1. **Import Libraries :**

Start by importing necessary library

**Program:**

# Import necessary libraries

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

1. **Load The Public Transport Data Set :**

**Program :**

# Load the public transport dataset (replace 'your\_dataset.csv' with the actual dataset file)

data = pd.read\_csv('your\_dataset.csv')

1. **Data Cleaning :**

Data cleaning involves identifying and correcting errors or inconsistencies in a dataset to ensure its accuracy and reliability.

**Program:**

# Data Cleaning

# Remove missing values

data.dropna(inplace=True)

1. **Data Transformation :**

Data transfer refers to the movement of data from one location to another.

**Program :**

# Data Transformation

# Convert stop addresses to latitude and longitude

# You can use a geocoding library or web service for this

1. **Feature Engineering:**

Depending on your dataset, you may need to create new features ortransform existing ones. This can involve one-hot encoding categoricalvariables, handling date/time data, or scaling numerical features.

**Program :**

# Feature Engineering

# Calculate travel time between stops

data['travel\_time'] = data['arrival\_time'] - data['departure\_time']

# Normalize numerical features (if needed)

scaler = StandardScaler()

data['normalized\_travel\_time'] = scaler.fit\_transform(data['travel\_time'].values.reshape(-1, 1))

1. **Data Splitting :**

Split your dataset into training and testing sets. This helps you evaluate,your model's performance later.

# Split the data into training, validation, and test sets

# You may want to do this if you plan to build a machine learning model

1. **Export Preprocessed Data :**

data.to\_csv('preprocessed\_data.csv', index=False)

**Importance of loading and processing dataset :**

Loading and preprocessing the dataset is an important first step in building any machine learning model. However, it is especially important for ,public transport optimization**.**

By loading and preprocessing the dataset, we can ensure that the machine learning algorithm is able to learn from the data effectively and accurately.

**Python script on the IoT sensors to send real-time location and ridership data to the transit information platform.**

Python script for IoT sensors to send real-time location and ridership data to a transit information platform, you can use various technologies and protocols such as MQTT for communication and GPS for location tracking. I'll provide you with a simplified example using Python, the paho-mqtt library for MQTT communication, and a simulated GPS module. Keep in mind that you'll need to adapt this script to your specific hardware and platform.

Here's a step-by-step guide:

1. Install the necessary libraries if you haven't already:

**Bash**

pip install paho-mqtt

**2. Create a Python script for the IoT sensor:**

import paho.mqtt.client as mqtt

import random

import time

from geopy.geocoders import Nominatim

**# Simulated ridership and location data**

def generate\_random\_data():

ridership = random.randint(0, 50)

latitude = 37.7749 + random.uniform(-0.01, 0.01)

longitude = -122.4194 + random.uniform(-0.01, 0.01)

return ridership, latitude, longitude

**# MQTT configuration**

mqtt\_broker = "your\_mqtt\_broker\_address"

mqtt\_port = 1883

mqtt\_topic = "transit/sensors"

**# Initialize the MQTT client**

client = mqtt.Client()

client.connect(mqtt\_broker, mqtt\_port, 60)

**# Main loop to send data**

**while True:**

ridership, latitude, longitude = generate\_random\_data()

**# Reverse geocoding to get a location name from coordinates**

geolocator = Nominatim(user\_agent="transit\_sensor")

location = geolocator.reverse((latitude, longitude))

location\_name = location.address

**# Prepare the message**

message = f"Ridership: {ridership}, Location: {location\_name}, Lat: {latitude}, Lon: {longitude}"

**# Publish data to the MQTT topic**

client.publish(mqtt\_topic, message)

print(f"Published: {message}")

**# Adjust the sleep interval based on your desired update frequency**

time.sleep(30) # Update every 30 seconds

**In this script**:

* We generate random ridership and location data for simulation purposes. In a real scenario, you would replace this with actual sensor data.
* We use the paho-mqtt library to establish an MQTT connection and publish data to a specific topic on the MQTT broker.
* We simulate sending data every 30 seconds. You can adjust the sleep interval based on your requirements.

1. Replace `"your\_mqtt\_broker\_address"` with the address of your MQTT broker.

2.Run this script on your IoT sensor device, and it will continuously send ridership and location data to the specified MQTT topic.

3.Make sure that your transit information platform subscribes to the same MQTT topic to receive and process the data. You'll also need to handle real GPS data and connectivity to the transit platform, which may involve additional hardware and configurations.