PHASE-3 PROJECT SUBMISSION DOCUMENT

PROJECT TITLE: PUBLIC TRANSPORTATION OPTIMIZATION.

PHASE 3: DEVELOPMENT PART-1

TOPIC: DEVELOP A PYTHON SCRIPT ON THE IOT SENSORS TO SEND REAL- TIME LOCATION AND RIDESHIP DATA TO THE TRANSIT INFORMATION PLATFORM.

INTRODUCTION:

Public transportation optimization is a crucial process that aims to enhance the efficiency and effectiveness of public transportation systems in order to meet the ever-increasing demands and challenges faced by urban areas. This optimization involves implementing various strategies and technologies to improve the overall performance, sustainability, accessibility, and convenience of public transportation networks.

The importance of optimizing public transportation cannot be overstated. As cities continue to grow, traffic congestion and environmental concerns become more prevalent. Public transportation optimization seeks to provide viable alternatives to private car usage, reducing traffic congestion, carbon emissions, and air pollution. It also aims to enhance the accessibility of transportation for all members of the community, including those with mobility challenges or limited access to private vehicles.

There are several key elements involved in public transportation optimization. These include route planning and design, scheduling and frequency adjustments, fare integration and pricing strategies, vehicle fleet management, and the integration of technology and data analytics. By analyzing data, traffic patterns, and passenger behavior, decision-makers can make informed choices to optimize routes, reduce travel times, and ensure efficient use of resources.

In addition to improving the operational aspects of public transportation, optimization efforts can also focus on enhancing the overall passenger experience. This can be achieved through the implementation of technology solutions, such as real-time passenger information systems, mobile ticketing, and smart payment options. These innovations can make public transportation more user-friendly, convenient, and attractive to commuters.

Overall, public transportation optimization plays a crucial role in creating sustainable and efficient transportation systems in urban areas. By utilizing data-driven strategies, advanced technologies, and intelligent planning, cities can improve the reliability, accessibility, and affordability of public transportation, ultimately benefiting both the environment and the communities they serve.

INTRODUCTION OF IOT PUBLIC TRANSPORTATION OPTIMIZATION WITH PYTHON

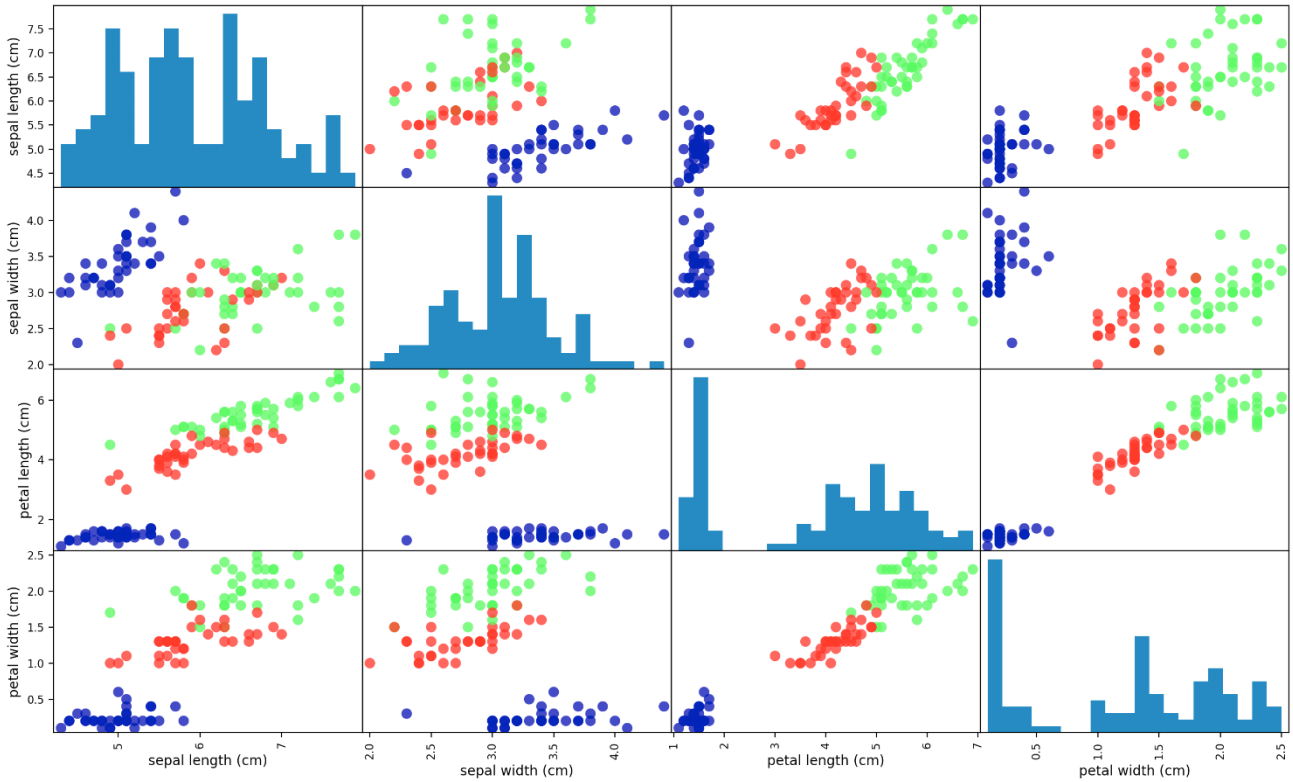
The Internet of Things (IoT) has revolutionized various industries, including transportation. One area where IoT has made significant advancements is in public transportation optimization. By implementing IoT technologies and using Python programming, public transportation systems can become more efficient, reliable, and user-friendly.

Public transportation optimization using IoT and Python involves the integration of various sensors, data collection devices, and intelligent algorithms. These technologies work together to gather real-time data about transit vehicles, passengers, and traffic conditions. This data is then analyzed and used to optimize routes, improve scheduling, and offer personalized services to passengers.

Python, being a versatile programming language, is widely used in developing IoT applications for public transportation optimization. It offers a range of libraries and frameworks that facilitate the collection, analysis, and visualization of data. Python's simplicity and readability make it easy to develop and maintain IoT solutions, even for those with limited programming experience.

DATA PREPARTION:

Public transportation plays a crucial role in urban transportation systems, providing an efficient and sustainable mode of commuting for a large number of people. To ensure an optimized and effective public transportation system, it is essential to analyze and utilize data to make informed decisions and improvements. This data presentation aims to explore the various aspects of public transportation optimization using data-driven approaches.



PAIR PLOT OF DIFFERENT VARIABLES.

LOADING AND PREPROCESSING DATASETS IN PUBLIC TRANSPORTATION OPTIMIZATION.

1. IMPORT THE REQUIRED LIBRARIES:

```python
import pandas as pd
import numpy as np
```

2. LOAD THE DATASET INTO A PANDAS DATAFRAME:

```python
df = pd.read_csv('dataset.csv')
```

Replace 'dataset.csv' with the actual filename and path of your dataset.

3. Perform initial data exploration:

```python
# Display the first few rows of the dataset
print(df.head())

# Check the dimensions of the dataset
print(df.shape)

# Check the data types of the columns
print(df.dtypes)

# Check for missing values
print(df.isnull().sum())
```

4. Preprocess the dataset:

- Handle missing values: Decide how to handle missing values based on the specific dataset and problem domain. Options include dropping the rows or columns with missing values, imputing missing values, or using other advanced techniques.
- Convert data types: If necessary, convert any columns with incorrect data types to the appropriate format (e.g., converting strings to datetime objects).
- Remove unnecessary columns: Identify and remove any columns that are not relevant to the optimization problem.
- Handle categorical variables: If the dataset contains categorical variables, encode them into numerical representations using techniques like one-hot encoding or label encoding.

Here are some examples preprocessing steps:

```python
# Drop rows with missing values
df = df.dropna()

# Convert date column to datetime object
df['date'] = pd.to_datetime(df['date'])

# Remove unnecessary columns
df = df.drop(['column1', 'column2'], axis=1)

# Perform one-hot encoding for categorical variables
```

```
df = pd.get_dummies(df, columns=['category'])
```

5. Normalize or scale the data: If required, apply normalization or scaling techniques to ensure the features are on a similar scale.

This can be important for certain optimization algorithms.

Here is an example of applying min-max scaling:

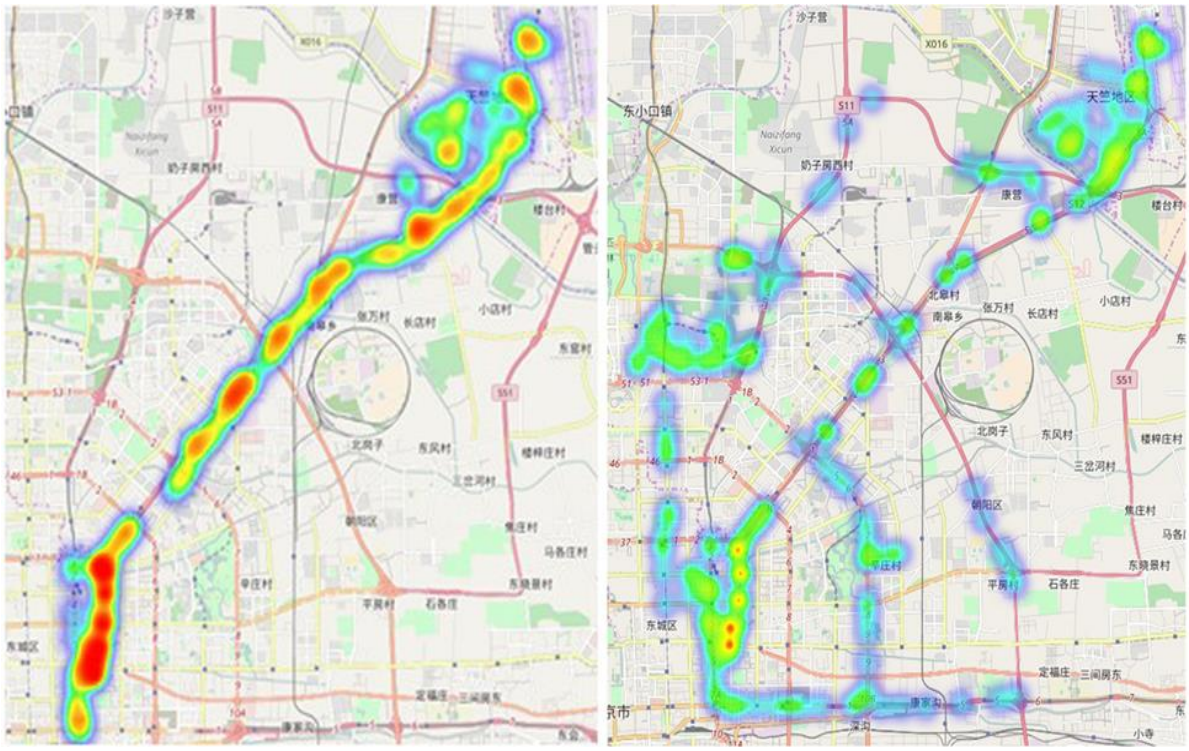
```python
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
```

DATA ACQUISITION

Data acquisition refers to the process of collecting and capturing data from various sources, such as sensors, instruments, and other data-generating devices. This can include physical measurements, signals, images, videos, and other forms of data.

The process of data acquisition typically involves connecting the data source to a data acquisition system or device, which can be a computer, data logger, or specialized hardware. The data acquisition system then captures and converts the data into a digital format for further processing and analysis.

It can be stored, processed, analyzed, and visualized to extract meaningful insights and make informed decisions. Data acquisition is a fundamental step in many scientific, industrial, and research applications, including environmental monitoring, industrial automation, quality control, scientific experiments, and more.

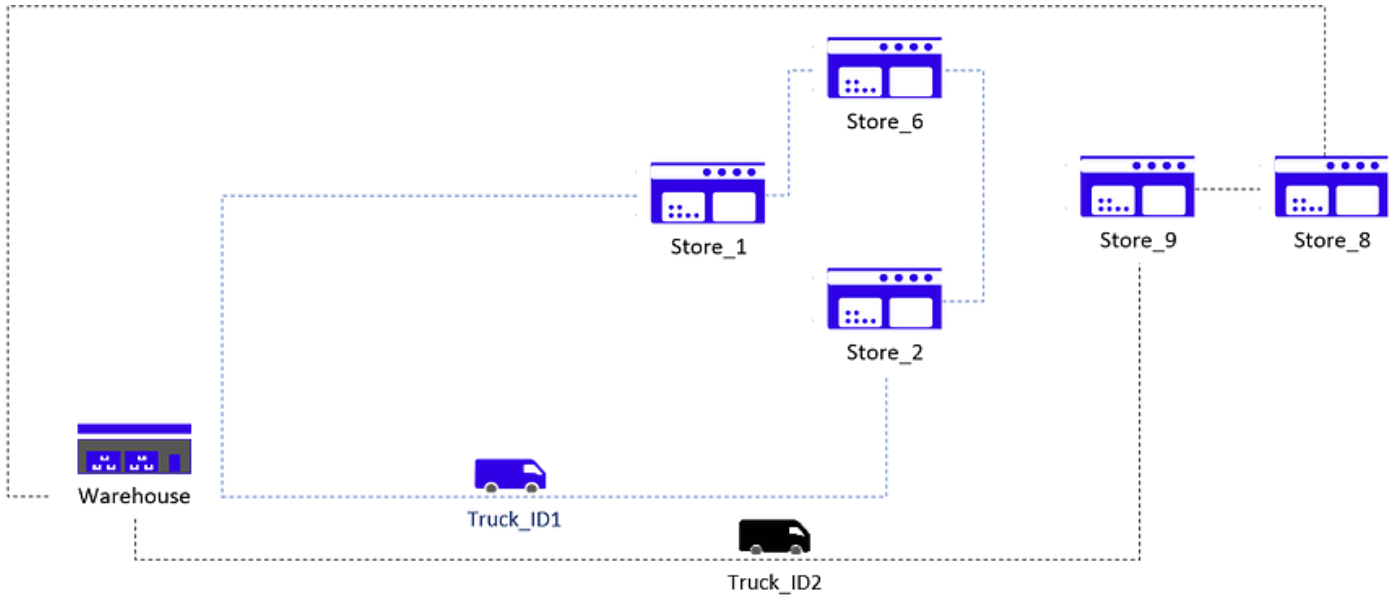


I. How do you make a transport plan

1. Problem Statement

Retail Stores Distribution with Full Truck Load (FTL)

- 1 Warehouse delivering stores by using three types of Trucks
- 49 Stores delivered
- 12 Months of Historical Data with 10,000 Deliveries
- 7 days a week of Operations
- 23 Cities
- 84 Trucks in your fleet



Date	Store	Truck_ID	Vol (T)
XX/XX/XXXX	Store_2	Truck_ID1	1.2
XX/XX/XXXX	Store_6	Truck_ID1	0.3
XX/XX/XXXX	Store_1	Truck_ID1	1.9
XX/XX/XXXX	Store_9	Truck_ID2	1.2
XX/XX/XXXX	Store_8	Truck_ID2	0.3

Transportation Routes for 2 trucks covering 5 stores in 2 routes — (Image by Author)

2. Objective: Reduce the Cost per Ton

Method: Shipment Consolidation

In this scenario, you are using 3rd party carriers that charge full trucks per destination:

City_En	3.5T (Rmb)	5T (Rmb)	8T (Rmb)	3.5T (Rmb/Ton)	5T (Rmb/Ton)	8T (Rmb/Ton)
City_1	485	650	800	139	130	100
City_2	640	700	820	183	140	103
City_3	690	780	890	197	156	111
City_4	810	1,000	1,150	231	200	144
City_5	1,300	1,568	1,723	371	314	215
City_6	1,498	1,900	2,100	428	380	263
City_7	980	1,250	1,450	280	250	181
City_8	1,350	1,450	1,500	386	290	188
City_9	1,350	1,450	1,500	386	290	188
City_10	850	1,000	1,200	243	200	150

The table above shows rates applied by carriers for each city delivered for each type of truck. Observing costs per ton are lower for larger trucks, one lever of improvement is maximizing shipments consolidation when building routes.Thus, the Route Transportation Planning Optimization's main target will be to cover a maximum number of stores per route.

II. Data Processing: Understand the Current Situation

1. Import Datasets

Before starting to think about the optimization model, your priority is to understand the current situation.

Starting with unstructured data coming from several sources, we'll need to build a set of data frames to model our network and provide visibility on the loading rate and list of stores delivered for each route.

Records of Deliveries per Store

Date	Truck_ID	Store_ID	FTL	Order	BOX	SKU	Loading (Tons)
9/1/2016	Truck_ID1	Store_ID1	3.5	16	311	83	2.404
9/1/2016	Truck_ID1	Store_ID2	3.5	18	178	83	1.668
9/1/2016	Truck_ID2	Store_ID3	3.5	10	74	54	0.81
9/1/2016	Truck_ID2	Store_ID4	3.5	19	216	88	2.413
9/1/2016	Truck_ID3	Store_ID5	3.5	10	117	54	1.119
9/1/2016	Truck_ID3	Store_ID6	3.5	15	294	92	2.962
9/1/2016	Truck_ID4	Store_ID7	3.5	5	42	19	0.421
9/1/2016	Truck_ID4	Store_ID8	3.5	12	125	88	1.138
9/1/2016	Truck_ID5	Store_ID9	5	18	201	95	2.19

Store Address

Code	city	Long	Lat	address
Store_ID1	City_Store1	31.952792	118.8192708	Address_1
Store_ID2	City_Store2	31.952792	118.8192718	Address_2
Store_ID3	City_Store3	31.675948	120.7468221	Address_3
Store_ID4	City_Store4	31.664448	120.7700006	Address_4
Store_ID5	City_Store5	31.750971	119.9478857	Address_5
Store_ID6	City_Store6	31.791351	119.9232302	Address_6
Store_ID7	City_Store7	31.79233	119.9768294	Address_7
Store_ID8	City_Store8	31.982972	119.5832084	Address_8
Store_ID9	City_Store9	31.996161	119.6341775	Address_9
Store_ID10	City_Store10	31.885547	121.1886473	Address_10
Store_ID11	City_Store11	30.310079	120.1515734	Address_11
Store_ID12	City_Store12	31.383616	121.2569408	Address_12
Store_ID13	City_Store13	31.387863	121.2797154	Address_13

Transportation Costs

City_En	3.5T (Rmb)	5T (Rmb)	8T (Rmb)	3.5T (Rmb/Ton)	5T (Rmb/Ton)	8T (Rmb/Ton)
City_1	485	650	800	139	130	100
City_2	640	700	820	183	140	103
City_3	690	780	890	197	156	111
City_4	810	1,000	1,150	231	200	144
City_5	1,300	1,568	1,723	371	314	215
City_6	1,498	1,900	2,100	428	380	263
City_7	980	1,250	1,450	280	250	181
City_8	1,350	1,450	1,500	386	290	188
City_9	1,350	1,450	1,500	386	290	188
City_10	850	1,000	1,200	243	200	150

2. Listing of stores delivered by each route

Let us process the initial data frame to list all stores delivered for each route.

1 Route = 1 Truck ID + 1 Date

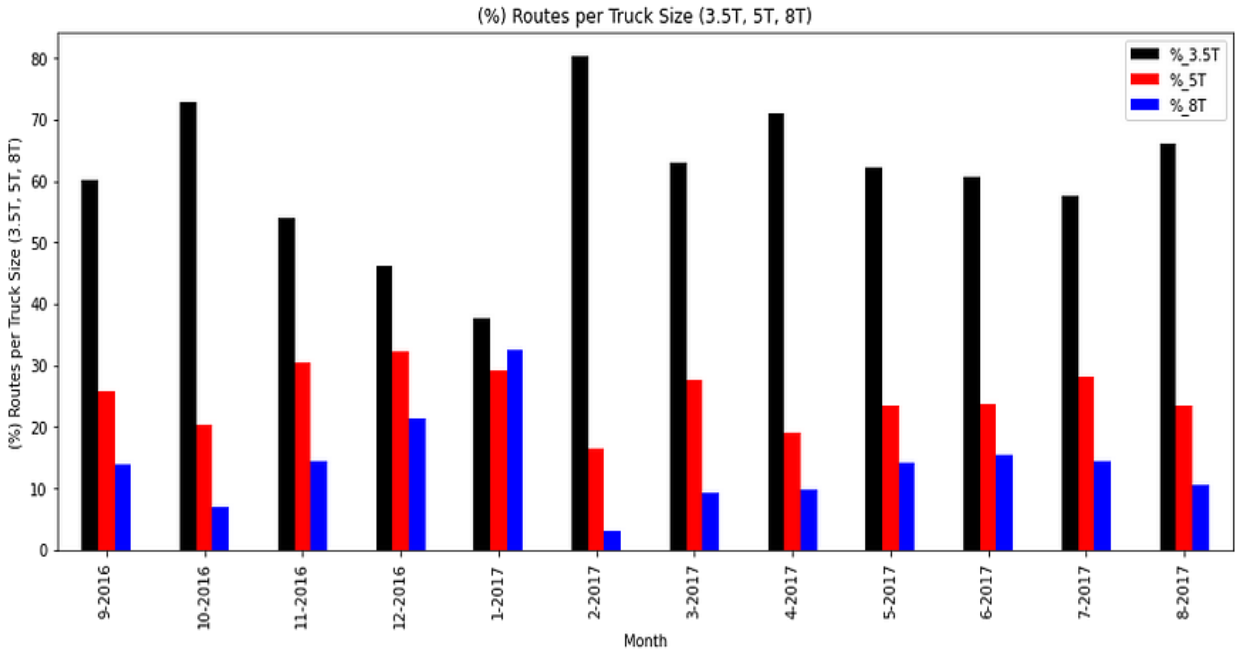
```
# Create Transport Plan
def transport_plan(data, dict_trucks, capacity_dict):
    # List of Stores per Truck for each DAY
    df_plan = pd.DataFrame(data.groupby(['Date', 'TruckID'])['Code'].apply(list))
    df_plan.columns = ['List_Code']
    # List of Box Quantity
    df_plan['List_BOX'] = data.groupby(['Date', 'TruckID'])['BOX'].apply(list)
    # Mean of FTL
    df_plan['FTL']  = data.groupby(['Date', 'TruckID'])['FTL'].mean()
    df_plan['Capacity(T)'] = df_plan['FTL'].map(capacity_dict)
    df_plan['List_Loading'] = data.groupby(['Date', 'TruckID'])['Loading(T)'].apply(list)
    df_plan['Count'] = df_plan['List_Loading'].apply(lambda t: len(t))
    df_plan['Total_tons(T)'] = data.groupby(['Date', 'TruckID'])['Loading(T)'].sum()

    # Distribute: one shipment per col
    # Stores
    d = df_plan['List_Code'].apply(pd.Series)
    for col in d:
       df_plan["Store%d" % (col+1)] = d[col]
```

```
# Boxes number
d = df_plan['List_BOX'].apply(pd.Series)
for col in d:
    df_plan["Box%d" % (col+1)] = d[col]
# Shipments Tonnage
d = df_plan['List_Loading'].apply(pd.Series)
for col in d:
    df_plan["Tons%d" % (col+1)] = d[col]

# Fill NaN + Drop useless columns
df_plan.fillna(0, inplace = True)
if 1 == 0:
        df_plan.drop(['List_Code'], axis = 1, inplace = True)
        df_plan.drop(['List_BOX'], axis = 1, inplace = True)
        df_plan.drop(['List_Loading'], axis = 1, inplace = True)

return df_plan
```

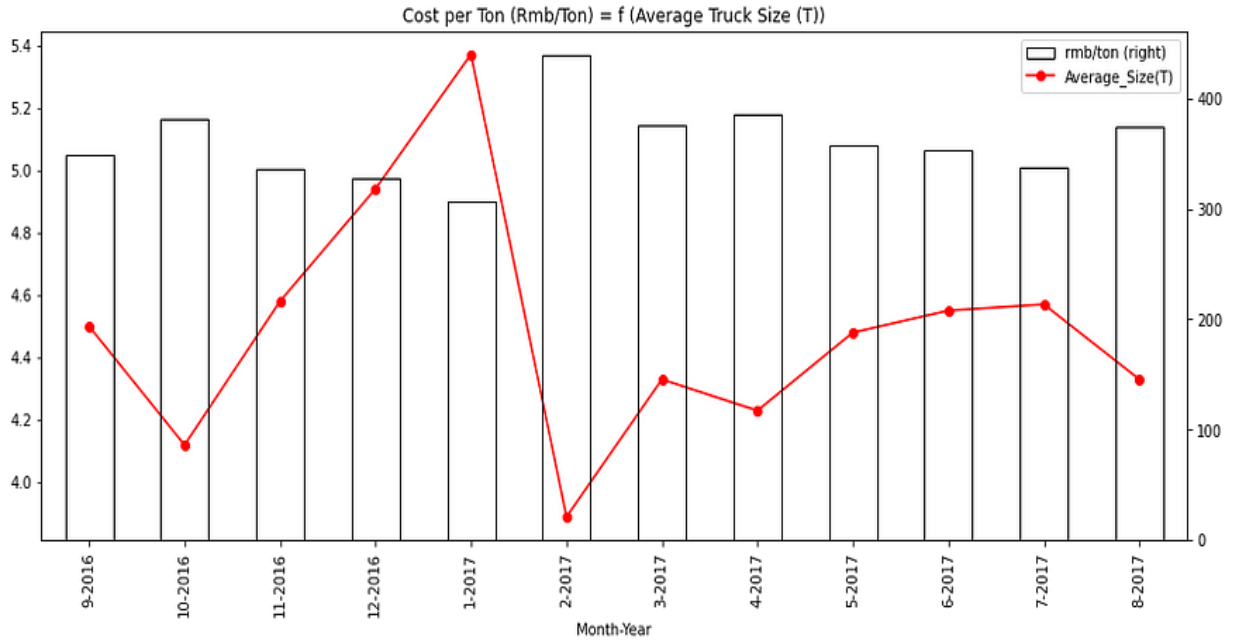
OUTPUT:

Date	TruckID	List_Code	Capacity(T)	List_Loading	Count	Total_tons(T)	Store1	Store2	Store3	Store4	Box1	Box2	Box3	Box4	Tons1	Tons2	Tons3	Tons4	Occupation(%	Available(T)
9/1/2016	Truck_ID1	['Store_ID6']	3.5	[2.91]	1	2.91	ID6	0	0	0	243	0	0	0	2.91	0	0	0	83.14	0.59
9/1/2016	Truck_ID2	['Store_ID34', '3.5		[0.3, 1.37, 0.47]	3	2.14	ID34	ID22	ID9	0	31	116	44	0	0.3	1.37	0.47	0	61.14	1.36
9/1/2016	Truck_ID3	['Store_ID18']	3.5	[1.5]	1	1.5	ID18	0	0	0	174	0	0	0	1.5	0	0	0	42.86	2
9/1/2016	Truck_ID4	['Store_ID37']	3.5	[2.3]	1	2.3	ID37	0	0	0	179	0	0	0	2.3	0	0	0	65.71	1.2
9/1/2016	Truck_ID5	['Store_ID34', '3.5		[2.14, 0.51]	2	2.65	ID34	ID48	0	0	168	46	0	0	2.14	0.51	0	0	75.71	0.85

Visualization: % Deliveries per Truck Size



(%) of Route per Truck Size (3.5T, 5T, 8T) — (Image by Author)



Impact of Average Truck Size (Ton) on Overall Cost per Ton (Rmb/Ton) — (Image by Author)

Insights

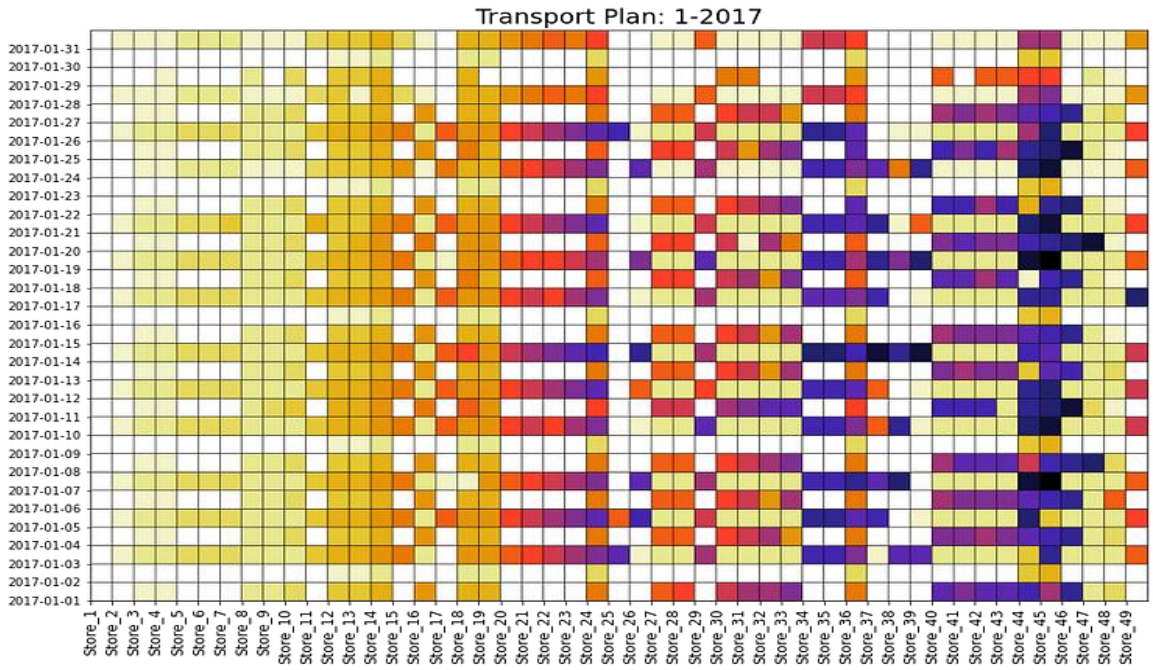
- **Average Truck Size:** a large majority of small trucks
- **Cost per ton:** the inverse proportion of cost per ton and average truck size

Find more inspiration for smart visualizations in this tutorial,

III. Understand Current Situation: Visualisation

1. Transportation Plan Visualisation

Objective: Get a simple visualisation of all deliveries per day with a focus on the number of different routes



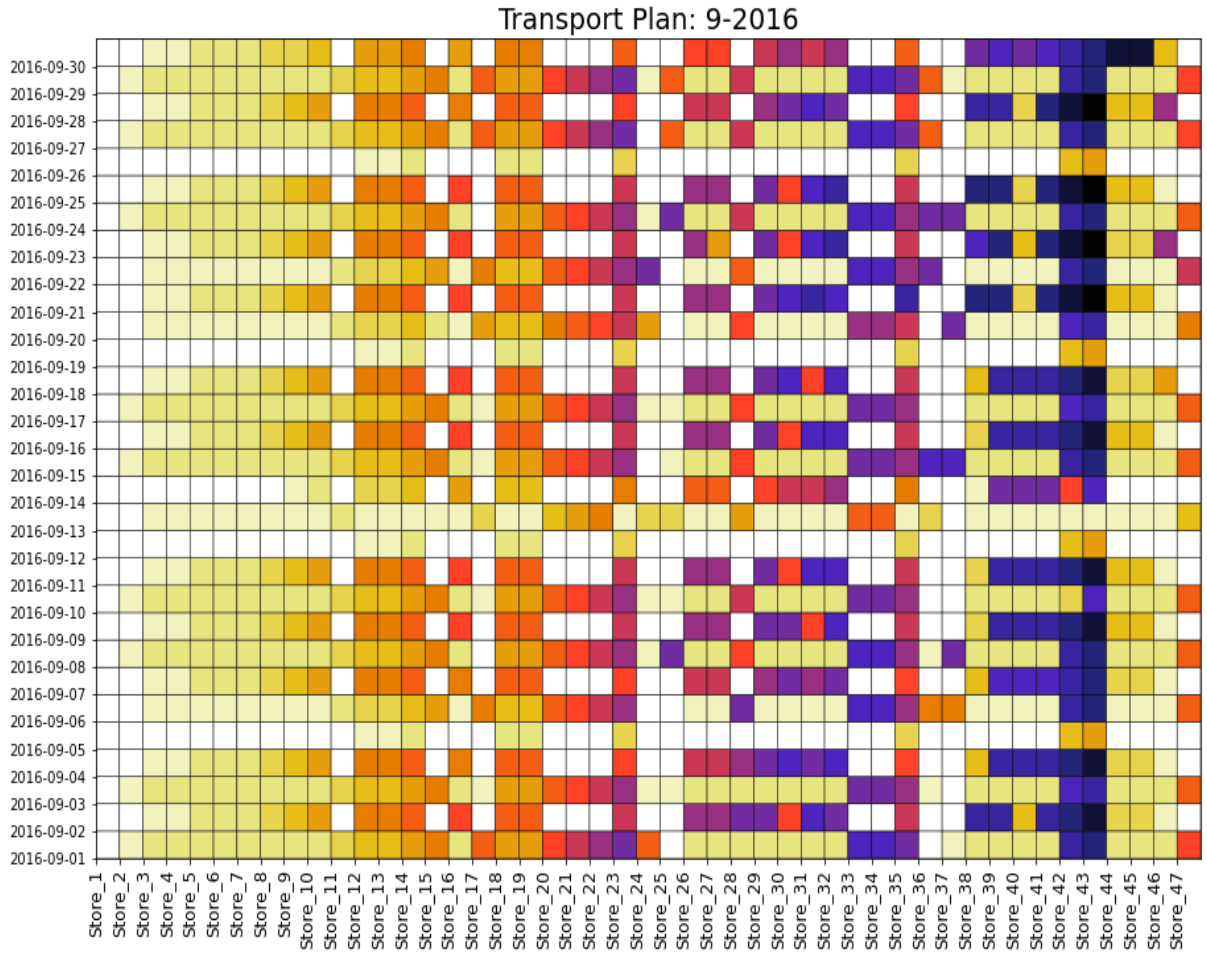
Solution: Python's Matplotlib grid function

- **Columns:** 1 Column = 1 Store
- **Rows:** 1 Row = 1 Day
- **Colour = White:** 0 delivery
- **Colours:** 1 Color = 1 Route (1 Truck)

Visual Insights

- **Delivery Frequency:** n deliveries per week
- **Number of Routes:** how many different colours do you have a day?
- **Colour = White:** no delivery
- **Routes:** do we have the same stores grouped from one day to another?

12 Months Overview

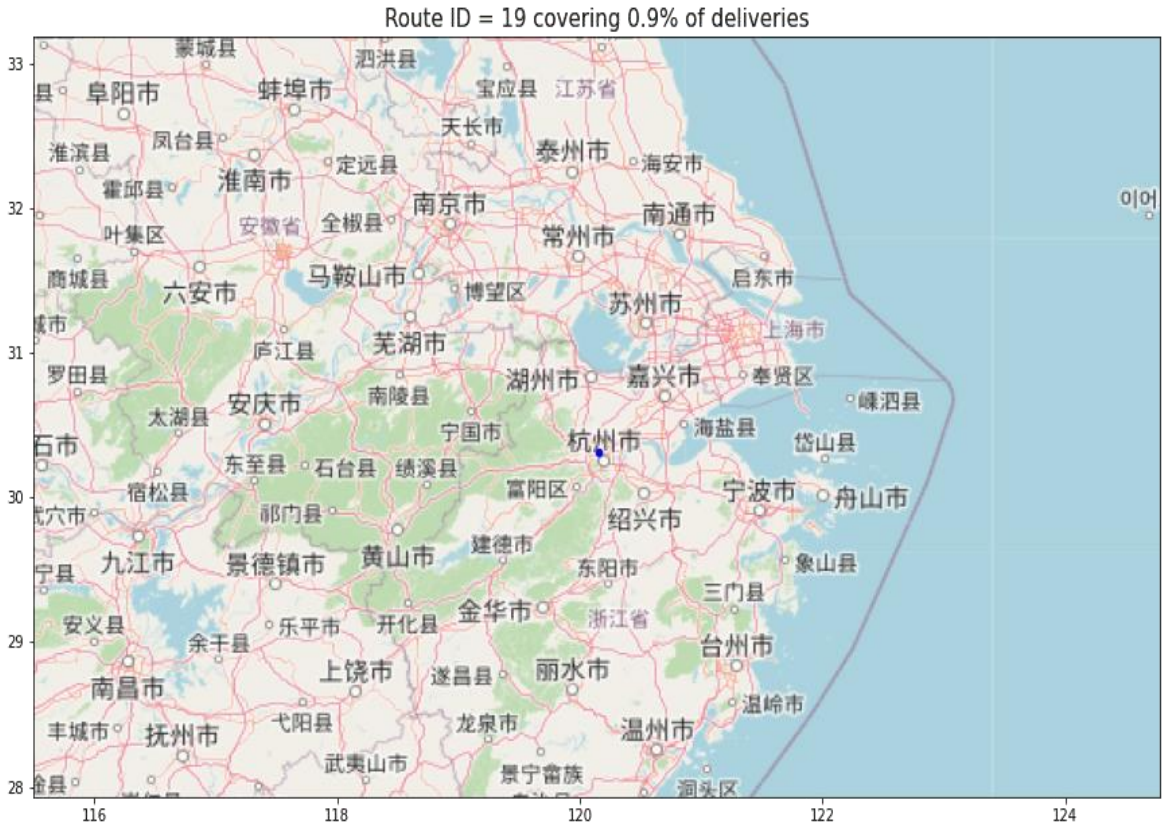


After optimization, this chart will help us to easily visualize the impact of new routing. A better routing means fewer routes per day so you'll have fewer colours per line.

2. Geographical Visualization of Store Deliveries

Objective

Visualisation of geographical locations delivered in the same route



VISUALIZATION OF ANALYTICS RESULTS:

Visualization of analytics results for public transportation optimization enables decision-makers to easily interpret the data and gain valuable insights. By transforming complex data into visual representations such as graphs, charts, maps, and interactive dashboards, the information becomes accessible and actionable.

These visualizations can help identify patterns, trends, and correlations in public transportation data, allowing stakeholders to make better-informed decisions. For example, by visualizing passenger flow in real-time, transport authorities can identify

overcrowded routes and take measures to optimize schedules or allocate additional resources.

CONCLUSION:

optimizing public transportation is crucial for improving mobility, reducing traffic congestion, and achieving sustainability goals. By implementing smart technologies, connectivity, and prioritizing efficiency, cities can create a more convenient, accessible, and eco-friendly transportation system for all.

THANK YOU