# Public Transport Efficiency Analysis

| | |
|---|---|
| **Date** | **31-10-2023** |
| **Team ID** | **1281** |
| **Project Name** | **Public Transport Efficiency Analysis** |

## Table of Contents



# 1.Introduction

The project starts with an introduction, emphasizing the transition from water portability analysis to public transport efficiency analysis. It highlights the use of visualization techniques and predictive modeling for data-driven decision-making in the public transport sector.

In our ongoing project, we are delving into the realm of data analysis, just as we did when exploring water portability. This time, our focus is on enhancing public transport efficiency. Similar to the way a smart parking system optimizes parking experiences, we aim to streamline public transportation systems by harnessing the power of data. Through the utilization of sensors, cameras, and advanced software, we will uncover hidden insights within the intricate web of data related to public transportation.

Our journey in this phase involves a shift in focus towards public transport efficiency analysis. We will employ a range of visualization techniques and predictive modeling to extract meaningful information from the data, much like a smart parking system optimizes parking spaces for drivers. The goal is to make informed, data-driven decisions that will ultimately enhance the efficiency and overall experience of public transportation for both passengers and operators.

## 2.Problem Statement

The primary objective is to analyze public transportation data to assess service efficiency, on-time performance, and passenger feedback. This analysis will support transportation improvement initiatives. Public transportation stands as a cornerstone of modern urban mobility, offering a cost-effective and eco-friendly alternative to private vehicles. Nevertheless, optimizing the efficiency of public transport systems is a multifaceted challenge shaped by a multitude of factors. Our primary objective in this analysis is to conduct a comprehensive assessment and enhancement of public transport efficiency. This endeavor encompasses the examination of critical factors such as route optimization, scheduling, infrastructure, user experience, and sustainability.
Objective: Our main goal is to leverage public transportation data to evaluate service efficiency, on-time performance, and passenger feedback, all in support of initiatives aimed at improving transportation services.

Data: To facilitate this analysis, we possess a dataset containing a diverse array of features pertaining to public transportation, encompassing bus, railway transportation, air transportation, and more. These features are complemented by corresponding sale prices. We will employ this dataset to train and evaluate our machine learning model, a crucial step in our quest to enhance public transport efficiency.

## 3.Data Preprocessing

This phase acknowledges the importance of data preprocessing for obtaining accurate predictions and insights. Data cleaning and preprocessing involve various steps, including handling missing values and data type conversions.

The provided code includes data preprocessing steps:

Reading data from a CSV file named 'dataset.CSV'.

Dropping duplicate rows from the dataset.

Visualizing missing values using a heatmap.

Handling mixed data types in the 'RouteID' column by converting it to a numeric data type.

Handling missing values by dropping rows with missing data.

Similar to our previous phase, data preprocessing remains a crucial step in our quest to understand and enhance public transport efficiency. Data preprocessing involves collecting and manipulating data to extract meaningful information. In this phase, our focus is on refining and improving the quality of our data, which is essential for achieving more accurate predictions and gaining valuable insights.

## 3.1 Data cleaning and preprocessing

```python
import pandas as pd

# Load your dataset
data = pd.read_csv('dataset.csv')
```

```python
# Data cleaning and preprocessing steps (e.g., handling missing values, datatype conversions, etc.)

# Example: Convert 'WeekBeginning' column to datetime

data['WeekBeginning'] = pd.to_datetime(data['WeekBeginning'], format='%d-%m-%Y %H:%M')

# More data cleaning and preprocessing steps can be added here
data.head(25)
```

| | TripID | RouteID | StopID | StopName | WeekBeginning | \ |
|---|---|---|---|---|---|---|
| 0 | 23631 | 100 | 14156 | 181 Cross Rd | 2013-06-30 | |
| 1 | 23631 | 100 | 14144 | 177 Cross Rd | 2013-06-30 | |
| 2 | 23632 | 100 | 14132 | 175 Cross Rd | 2013-06-30 | |
| 3 | 23633 | 100 | 12266 | Zone A Arndale Interchange | 2013-06-30 | |
| 4 | 23633 | 100 | 14147 | 178 Cross Rd | 2013-06-30 | |
| 5 | 23634 | 100 | 13907 | 9A Marion Rd | 2013-06-30 | |
| 6 | 23634 | 100 | 14132 | 175 Cross Rd | 2013-06-30 | |
| 7 | 23634 | 100 | 13335 | 9A Holbrooks Rd | 2013-06-30 | |
| 8 | 23634 | 100 | 13875 | 9 Marion Rd | 2013-06-30 | |
| 9 | 23634 | 100 | 13045 | 206 Holbrooks Rd | 2013-06-30 | |
| 10 | 23635 | 100 | 13335 | 9A Holbrooks Rd | 2013-06-30 | |
| 11 | 23635 | 100 | 13383 | 8A Marion Rd | 2013-06-30 | |
| 12 | 23635 | 100 | 13586 | 8D Marion Rd | 2013-06-30 | |
| 13 | 23635 | 100 | 12726 | 23 Findon Rd | 2013-06-30 | |
| 14 | 23635 | 100 | 13813 | 8K Marion Rd | 2013-06-30 | |
| 15 | 23635 | 100 | 14062 | 20 Cross Rd | 2013-06-30 | |
| 16 | 23636 | 100 | 12780 | 22A Crittenden Rd | 2013-06-30 | |
| 17 | 23636 | 100 | 13383 | 8A Marion Rd | 2013-06-30 | |
| 18 | 23636 | 100 | 14154 | 180 Cross Rd | 2013-06-30 | |
| 19 | 23636 | 100 | 13524 | 8C Marion Rd | 2013-06-30 | |
| 20 | 23636 | 100 | 14122 | 173 Cross Rd | 2013-06-30 | |
| 21 | 23636 | 100 | 13813 | 8K Marion Rd | 2013-06-30 | |
| 22 | 23637 | 100 | 14156 | 181 Cross Rd | 2013-06-30 | |
| 23 | 23637 | 100 | 14154 | 180 Cross Rd | 2013-06-30 | |
| 24 | 23637 | 100 | 13335 | 9A Holbrooks Rd | 2013-06-30 | |

| | NumberOfBoardings |
|---|---|
| 0 | 1 |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | 1 |
| 5 | 1 |
| 6 | 1 |
| 7 | 1 |
| 8 | 1 |
| 9 | 1 |
| 10 | 1 |
| 11 | 1 |
| 12 | 2 |
| 13 | 1 |
| 14 | 1 |
| 15 | 1 |
| 16 | 1 |

| | |
|---|---|
| 17 | 1 |
| 18 | 2 |
| 19 | 3 |
| 20 | 1 |
| 21 | 1 |
| 22 | 1 |
| 23 | 1 |
| 24 | 3 |

# 4.Design Thinking Process

The project appears to follow a design thinking approach, including:

## 4.1 Empathize:

Understanding the needs and priorities of the target audience, which includes commuters and transportation planners.

## 4.2 Define:

Setting clear objectives for the project, which include building a machine learning model with specific performance criteria and establishing a user-friendly web platform.

## 4.3 Ideate:

Exploring various approaches and techniques, such as machine learning models, real-time data integration, optimization algorithms, IoT sensors, and data visualization.

## 4.4 Prototype:

Developing a prototype to test core functionalities and gather early user feedback.

## 4.5 Ideate:

-Explore various machine learning models such as regression, decision trees, and neural networks to predict efficiency.
-Investigate the integration of real-time data sources, like GPS tracking and passenger feedback, for accurate analysis.
-Consider optimization algorithms for route planning and scheduling to enhance efficiency.
-Explore the possibility of incorporating IoT (Internet of Things) sensors to monitor vehicle conditions and passenger loads.
-Evaluate data visualization techniques to present efficiency insights in a user-friendly manner.

## 4.6 Actions:

-Investigate various machine learning algorithms, including regression, decision trees, random forests, and neural networks.
-Experiment with feature engineering methods to boost model accuracy.

# 5.Visualization:

The code starts by importing the necessary libraries: numpy, pandas, and os.

It then uses a loop with os.walk to explore the files in a directory ('dataset.csv') and prints the paths of the files found.

The code imports the Pandas library once again (redundantly) and reads the dataset from a CSV file named 'dataset.CSV' using pd.read_csv. The argument low_memory=False is used to disable low memory mode.

It prints the shape of the dataset (number of rows and columns) and displays the first 30 rows using data.shape and data.head(30).

The code handles missing values by converting the 'WeekBeginning' column to a datetime format. It uses the 'coerce' option to handle errors and prints the first few rows of the 'WeekBeginning' column after the conversion.

The 'StopName' column is cleaned by removing leading and trailing whitespaces using the str.strip() method. The cleaned 'StopName' column is then displayed.

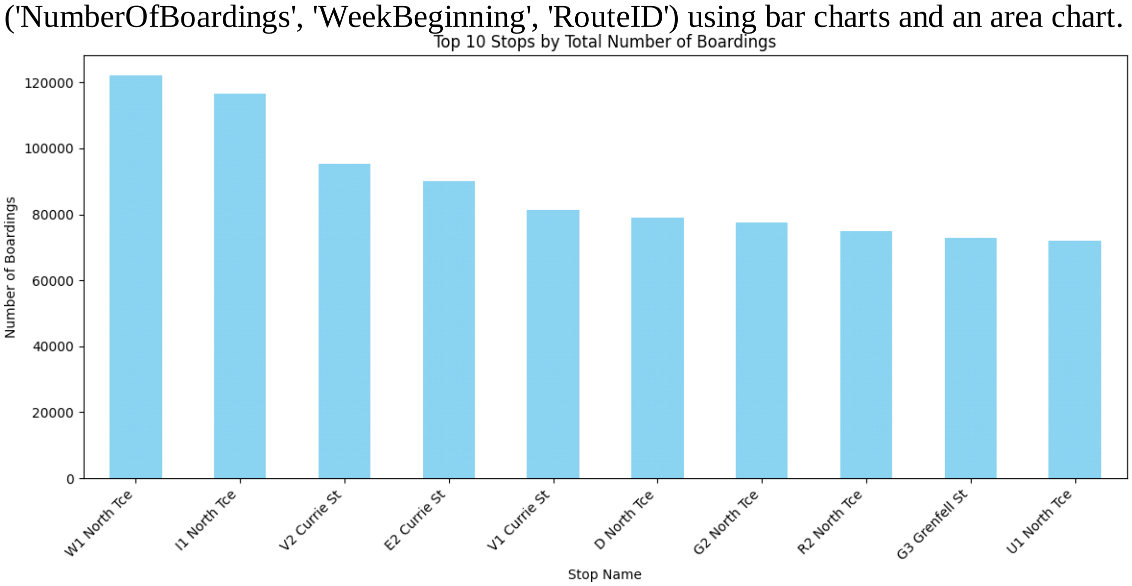
It prints the number of unique values in each column using data.nunique().

The code displays the shape, column names, and the first 3 rows of the dataset.

It checks for missing values in the dataset using data.isnull().sum() and prints the results.

The unique values in the 'WeekBeginning' column are printed using data['WeekBeginning'].unique().

Finally, the code sets up a Matplotlib subplot with six plots and visualizes data from various columns ('NumberOfBoardings', 'WeekBeginning', 'RouteID') using bar charts and an area chart.




Visualization is a key component of the project, and the code provided demonstrates the creation of line and bar charts. These charts help in understanding trends in boarding counts and identifying top stops by the number of boardings.

```
import numpy as np import
pandas as pdimport os
    for dirname, _, filenames in os.walk('dataset.csv'):for filename in filenames:
            print(os.path.join(dirname, filename))

print("Load the dataset")import pandas as
pd
data = pd.read_csv('dataset.CSV', low_memory=False)data.shape
data.head(30) Load the

dataset

        TripID RouteID    StopID                                  StopName
WeekBeginning   \
0              23631           100        14156                               181 Cross Rd   30-06-2013
00:00
1              23631           100        14144                               177 Cross Rd   30-06-2013
00:00
2              23632           100        14132                               175 Cross Rd   30-06-2013
00:00
3              23633           100        12266   Zone A Arndale Interchange   30-06-2013
00:00
4              23633           100        14147                               178 Cross Rd   30-06-2013
00:00
5              23634           100        13907                             9A    Marion Rd   30-06-2013
00:00
6              23634           100        14132                               175 Cross Rd   30-06-2013
00:00
7              23634           100        13335                        9A    Holbrooks Rd   30-06-2013
00:00
8              23634           100        13875                               9    Marion Rd   30-06-2013
00:00
9              23634           100        13045                        206 Holbrooks Rd   30-06-2013
00:00
10             23635           100        13335                        9A    Holbrooks Rd   30-06-2013
```

| | | | | | | |
|---|---|---|---|---|---|---|
| 00:00 | | | | | | |
| 11 | 23635 | 100 | 13383 | 8A | Marion Rd | 30-06-2013 |
| 00:00 | | | | | | |
| 12 | 23635 | 100 | 13586 | 8D | Marion Rd | 30-06-2013 |
| 00:00 | | | | | | |
| 13 | 23635 | 100 | 12726 | 23 | Findon Rd | 30-06-2013 |
| 00:00 | | | | | | |
| 14 | 23635 | 100 | 13813 | 8K | Marion Rd | 30-06-2013 |
| 00:00 | | | | | | |
| 15 | 23635 | 100 | 14062 | 20 | Cross Rd | 30-06-2013 |
| 00:00 | | | | | | |
| 16 | 23636 | 100 | 12780 | 22A | Crittenden Rd | 30-06-2013 |

00:00

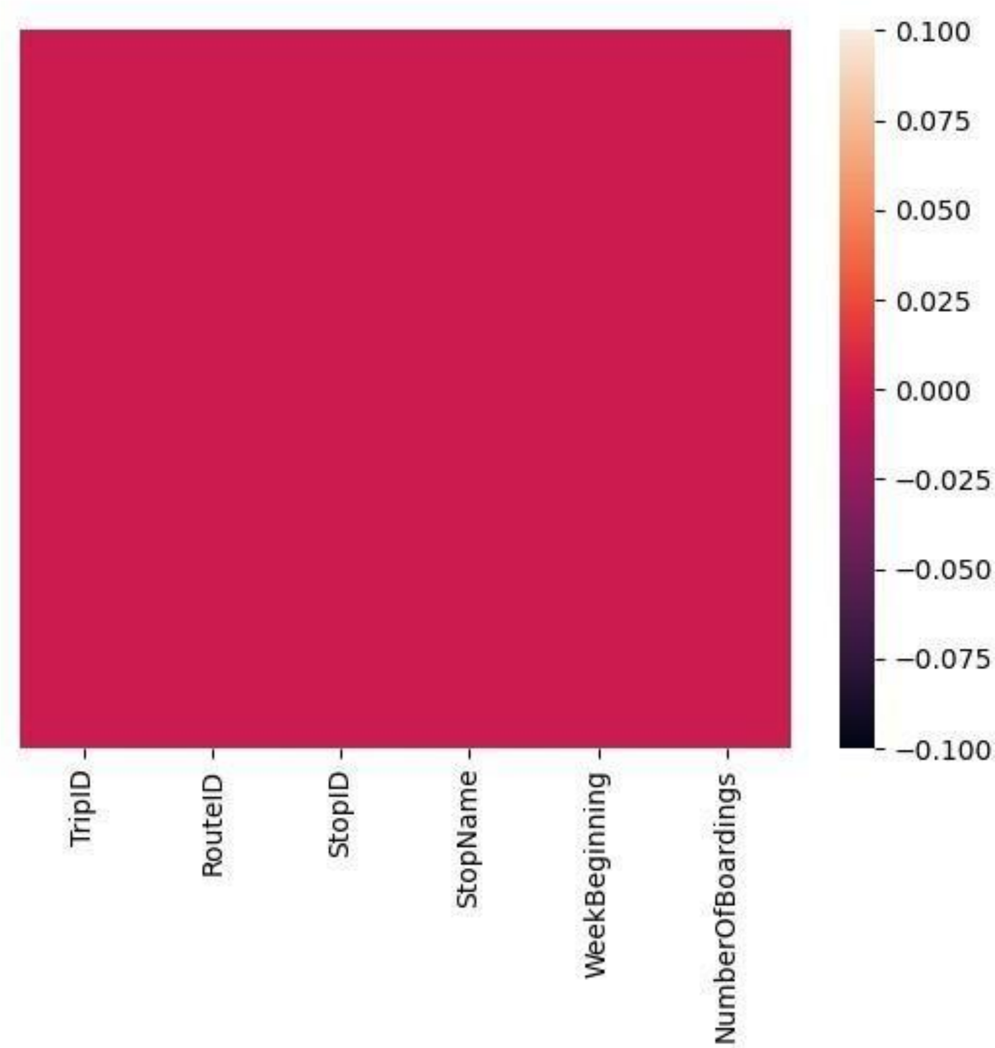
| | | | | | |
|---|---|---|---|---|---|
| 17 | 23636 | 100 | 13383 | 8A Marion Rd | 30-06-2013 00:00 |
| 18 | 23636 | 100 | 14154 | 180 Cross Rd | 30-06-2013 00:00 |
| 19 | 23636 | 100 | 13524 | 8C Marion Rd | 30-06-2013 00:00 |
| 20 | 23636 | 100 | 14122 | 173 Cross Rd | 30-06-2013 00:00 |
| 21 | 23636 | 100 | 13813 | 8K Marion Rd | 30-06-2013 00:00 |
| 22 | 23637 | 100 | 14156 | 181 Cross Rd | 30-06-2013 00:00 |
| 23 | 23637 | 100 | 14154 | 180 Cross Rd | 30-06-2013 00:00 |
| 24 | 23637 | 100 | 13335 | 9A Holbrooks Rd | 30-06-2013 00:00 |
| 25 | 23637 | 100 | 12266 | Zone A Arndale Interchange | 30-06-2013 00:00 |
| 26 | 23637 | 100 | 13196 | 13 Holbrooks Rd | 30-06-2013 00:00 |
| 27 | 23638 | 100 | 12562 | 218 Findon Rd | 30-06-2013 00:00 |
| 28 | 23638 | 100 | 12266 | Zone A Arndale Interchange | 30-06-2013 00:00 |
| 29 | 23638 | 100 | 13875 | 9 Marion Rd | 30-06-2013 00:00 |

| | NumberOfBoardings |
|---|---|
| 0 | 1 |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | 1 |
| 5 | 1 |
| 6 | 1 |
| 7 | 1 |
| 8 | 1 |
| 9 | 1 |
| 10 | 1 |
| 11 | 1 |
| 12 | 2 |
| 13 | 1 |
| 14 | 1 |
| 15 | 1 |
| 16 | 1 |
| 17 | 1 |
| 18 | 2 |
| 19 | 3 |

```
20                                      1
21                                      1
22                                      1
23                                      1
24                                      3
25                                      5
26                                      1
27                                      1
28                                      3
29                                      1
```

```
data = data.drop_duplicates()import
seaborn as sns
sns.heatmap(data.isnull(),yticklabels= False)print("\nCheck data
types of columns") print(data.dtypes)
```

```
Check data types of columns TripID
                                     int64
RouteID                             object
StopID                               int64
StopName                            object
WeekBeginning                       object
NumberOfBoardings int64dtype: object
```




```
data['RouteID'] = pd.to_numeric(data['RouteID'], errors='coerce')print("Handle mixed data types")
print(data.dtypes)
```

```
Handle mixed data types TripID    int64
RouteID                  float64
StopID                     int64
StopName                  object
WeekBeginning             object
NumberOfBoardings          int64
dtype: object
```

```
data = data.dropna() print("\nHandle missing
values")print(data.shape)
```

```
Handle missing values(1008700, 6)
```

```python
data['WeekBeginning'] = pd.to_datetime(data['WeekBeginning'],errors='coerce')
print("\nConvert 'WeekBeginning' column to datetime format")
print(data['WeekBeginning'].head())
```

```
Convert 'WeekBeginning' column to datetime format0     2013-06-30
1        2013-06-30
2        2013-06-30
3        2013-06-30
4        2013-06-30
Name: WeekBeginning, dtype: datetime64[ns]

C:\Users\bavik\AppData\Local\Temp\ipykernel_15464\2765944061.py:1:UserWarning: Parsing dates in %d-%m-%Y %H:%M format when dayfirst=False (the default) was specified. Pass `dayfirst=True` orspecify a format to silence this warning.
  data['WeekBeginning'] = pd.to_datetime(data['WeekBeginning'],errors='coerce')
```

```python
data['StopName'] = data['StopName'].str.strip()
print("\nClean'StopName'column")
 print(data['StopName'].head())
```

```python
print(data.nunique())
```

```
TripID                  312
                          3

RouteID                  20
StopID                  963
StopName                577
WeekBeginning            54
NumberOfBoardings       156
dtype: int64
```

```python
data.shape
data.columns
data.head(3)
```

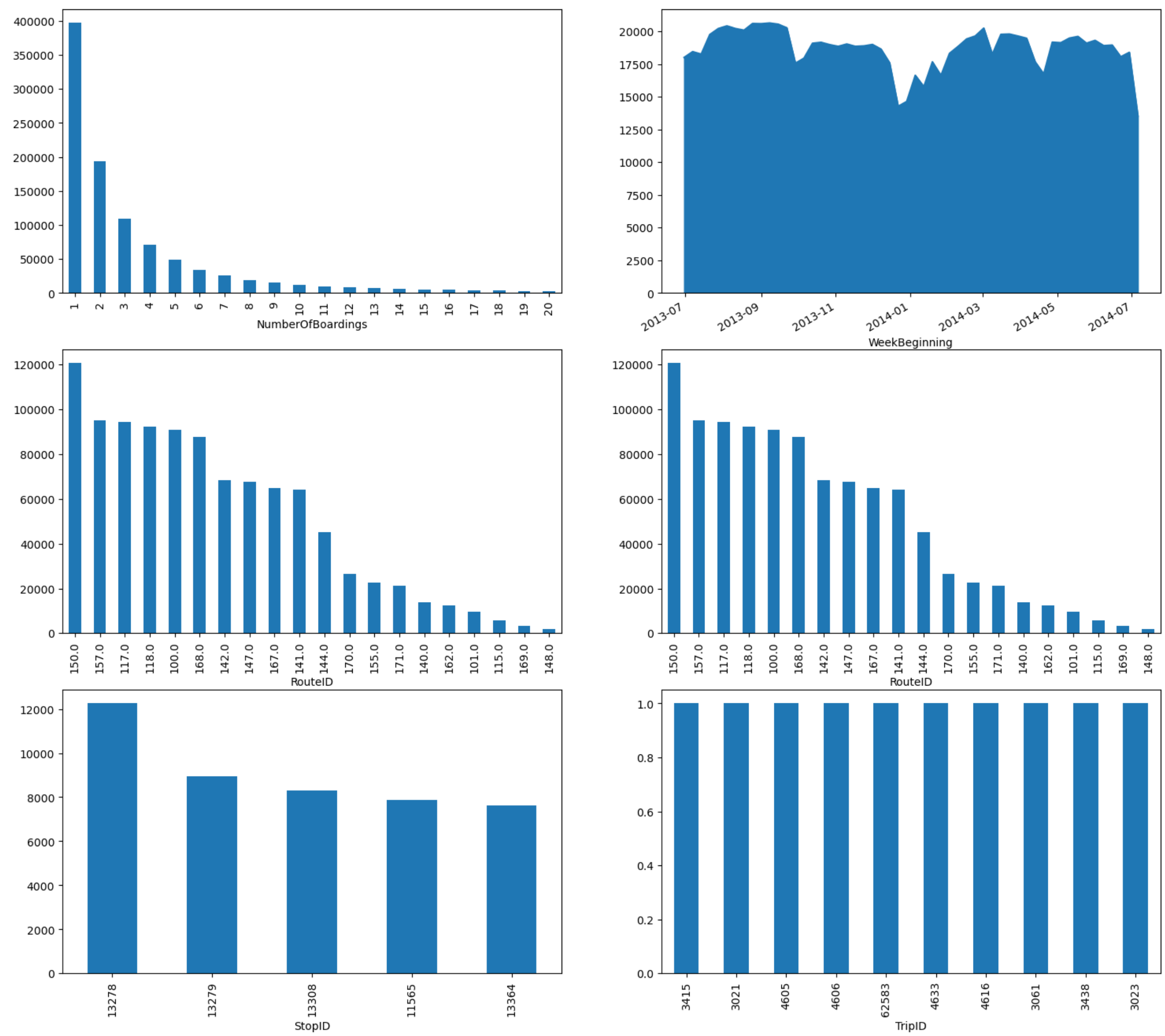
```
   TripID   RouteID   StopID                StopName WeekBeginning
   NumberOfBoardings
                                       1415    181    Cros   Rd          2013-06-30
  0     23631       100.0
  1
  1     23631       100.0              1414    177    Cros   Rd          2013-06-30
  1
  2     23632       100.0              1413    175    Cros   Rd          2013-06-30
1
```

```
data.isnull().sum() TripID              0
RouteID                                 0
StopID                                  0
StopName                                0
WeekBeginning                           0
NumberOfBoardings                       0
dtype: int64 data['WeekBeginning'].unique()
```

```
<DatetimeArray>
['2013-06-30 00:00:00', '2013-07-07 00:00:00', '2013-07-14 00:00:00',
 '2013-07-21 00:00:00', '2013-07-28 00:00:00', '2013-08-04 00:00:00',
 '2013-08-11 00:00:00', '2013-08-18 00:00:00', '2013-08-25 00:00:00',
 '2013-09-01 00:00:00', '2013-09-08 00:00:00', '2013-09-15 00:00:00',
 '2013-09-22 00:00:00', '2013-09-29 00:00:00', '2013-10-06 00:00:00',
 '2013-10-13 00:00:00', '2013-10-20 00:00:00', '2013-10-27 00:00:00',
 '2013-11-03 00:00:00', '2013-11-10 00:00:00', '2013-11-17 00:00:00',
 '2013-11-24 00:00:00', '2013-12-01 00:00:00', '2013-12-08 00:00:00',
 '2013-12-15 00:00:00', '2013-12-22 00:00:00', '2013-12-29 00:00:00',
 '2014-01-05 00:00:00', '2014-01-12 00:00:00', '2014-01-19 00:00:00',
 '2014-01-26 00:00:00', '2014-02-02 00:00:00', '2014-02-09 00:00:00',
 '2014-02-16 00:00:00', '2014-02-23 00:00:00', '2014-03-02 00:00:00',
 '2014-03-09 00:00:00', '2014-03-16 00:00:00', '2014-03-23 00:00:00',
 '2014-03-30 00:00:00', '2014-04-06 00:00:00', '2014-04-13 00:00:00',
 '2014-04-20 00:00:00', '2014-04-27 00:00:00', '2014-05-04 00:00:00',
 '2014-05-11 00:00:00', '2014-05-18 00:00:00', '2014-05-25 00:00:00',
 '2014-06-01 00:00:00', '2014-06-08 00:00:00', '2014-06-15 00:00:00',
 '2014-06-22 00:00:00', '2014-06-29 00:00:00', '2014-07-06 00:00:00']
Length: 54, dtype: datetime64[ns]
```

```
import matplotlib.pyplot as plt fig,axrr=plt.subplots(3,2,figsize=(18,18))
data['NumberOfBoardings'].value_counts().sort_index().head(20).plot.bar(ax=axrr[0][0])
data['WeekBeginning'].value_counts().plot.area(ax=axrr[0][1])
data['RouteID'].value_counts().head(20).plot.bar(ax=axrr[1][0])
data['RouteID'].value_counts().tail(20).plot.bar(ax=axrr[1][1])
data['StopID'].value_counts().head(5).plot.bar(ax=axrr[2][0])
data['TripID'].value_counts().tail(10).plot.bar(ax=axrr[2][1])

<Axes: xlabel='TripID'>
```




```
data.to_csv('cleaned_data.csv', index=False) print("\nSave the cleaned dataset
to a new CSV file")print("Cleaned dataset saved successfully.")
```

```
Save the cleaned dataset to a new CSV fileCleaned dataset
saved successfully.
```

# 6 Advanced Data Analysis:

Advanced data analysis plays a vital role in optimizing public transport systems, making them more efficient, reliable, and passenger-friendly. Here are some advanced data analysis techniques and their applications in public transport

## 6.1 Advanced Analytics and Modeling

```python
import pandas as pd
# Group by RouteID and sum the NumberOfBoardings
boarding_by_route = data.groupby('RouteID')['NumberOfBoardings'].sum()

# Display the result
print(boarding_by_route)
```

```
RouteID
117        312470
118        319790
140         83064
141        331118
142         79091
147        169540
148          5190
150        318672
168        296199
169         13397
170        143076
```

```
171          91911
100         328740
100B          8250
100C         11828
100K          6364
100N          6419
100P         13277
100S           260
101          39114
115          15460
117          67637
142         287270
144         183253
144G         15814
147         136496
150         105953
150B         55517
150P          8147
155          98191
157         307301
157X         81745
162          92171
167         237238
167C         32195
168         30858
Name: NumberOfBoardings, dtype: int64
```

### Calculating Average Boarding Counts per Stop

```
# Group by StopID and calculate the average number of boardings
avg_boardings_per_stop = data.groupby('StopID')['NumberOfBoardings'].mean()

# Display the result
print(avg_boardings_per_stop)
```

```
StopID
10817    2.776013
10818    2.333333
10843    2.257143
10877    2.326316
10879    1.400000
           ...
18408    1.875000
18409    2.714286
18410    1.500000
18411    1.156250
18493    9.122678
Name: NumberOfBoardings, Length: 969, dtype: float64
```

### Finding Stops with Highest Weekly Boarding Counts

```
# Convert WeekBeginning to datetime and extract week number data['WeekBeginning'] =
pd.to_datetime(data['WeekBeginning'])data['WeekNumber'] = data['WeekBeginning'].dt.week

# Group by StopName and WeekNumber, then sum the NumberOfBoardingsweekly_boarding_counts =
data.groupby(['StopName', 'WeekNumber'])['NumberOfBoardings'].sum()

# Find stops with the highest weekly boarding countsstops_with_highest_boardings =
weekly_boarding_counts.groupby('StopName').idxmax()

# Display the result
print(stops_with_highest_boardings)
```

```
StopName
1 Anzac Hwy                                  (1 Anzac Hwy, 26)
1 Fullarton Rd                               (1 Fullarton Rd, 8)
1 George St                                  (1 George St, 27)
1 Glen Osmond Rd                             (1 Glen Osmond Rd, 33)
1 Henley Beach Rd                            (1 Henley Beach Rd, 26)
                                             ...
Zone B Registry Rd Flinders Un (Zone B Registry Rd Flinders Un, 11)Zone B West Lakes
Interchange   (Zone B West Lakes Interchange, 26) Zone C Moseley St    (Zone    C
Moseley St, 26)Zone D Arndale Interchange    (Zone D Arndale Interchange, 38)Zone D
Port Adelaide Interchan (Zone D Port Adelaide Interchan, 26) Name: NumberOfBoardings,
Length: 583, dtype: object
```

## Analyzing Trends Over Time (Weekly/Monthly)

```
# Convert WeekBeginning to datetime and extract week and month
data['WeekBeginning'] = pd.to_datetime(data['WeekBeginning'])data['WeekNumber']
= data['WeekBeginning'].dt.week data['Month'] = data['WeekBeginning'].dt.month

# Group by WeekNumber and Month, then sum the NumberOfBoardingsweekly_boarding_trends =
data.groupby(['WeekNumber', 'Month'])['NumberOfBoardings'].sum()

# Display the result
print(weekly_boarding_trends)
```

```
WeekNumber   Month
1            1         59791
2            1         55026
3            1         67844
4            1         62204
5            2         87621
6            2         79964
```

| | | |
|---|---|---|
| 7 | 2 | 86610 |
| 8 | 2 | 91046 |
| 9 | 3 | 98500 |
| 10 | 3 | 66953 |
| 11 | 3 | 94828 |
| 12 | 3 | 95643 |
| 13 | 3 | 94406 |
| 14 | 4 | 92959 |
| 15 | 4 | 62636 |
| 16 | 4 | 51434 |
| 17 | 4 | 88624 |
| 18 | 5 | 90852 |
| 19 | 5 | 92782 |
| 20 | 5 | 92112 |
| 21 | 5 | 89378 |
| 22 | 6 | 91608 |
| 23 | 6 | 73602 |
| 24 | 6 | 83086 |
| 25 | 6 | 76725 |
| 26 | 6 | 161049 |
| 27 | 7 | 121795 |
| 28 | 7 | 70588 |
| 29 | 7 | 85288 |
| 30 | 7 | 94344 |
| 31 | 8 | 95061 |
| 32 | 8 | 93992 |
| 33 | 8 | 92247 |
| 34 | 8 | 95341 |
| 35 | 9 | 94762 |
| 36 | 9 | 93643 |
| 37 | 9 | 94053 |
| 38 | 9 | 89866 |
| 39 | 9 | 67959 |
| 40 | 10 | 65428 |
| 41 | 10 | 87246 |
| 42 | 10 | 87703 |
| 43 | 10 | 86839 |
| 44 | 11 | 84346 |
| 45 | 11 | 82642 |
| 46 | 11 | 81556 |
| 47 | 11 | 80333 |
| 48 | 12 | 80176 |
| 49 | 12 | 75652 |
| 50 | 12 | 66079 |
| 51 | 12 | 37207 |
| 52 | 12 | 41587 |

Name: NumberOfBoardings, dtype: int64

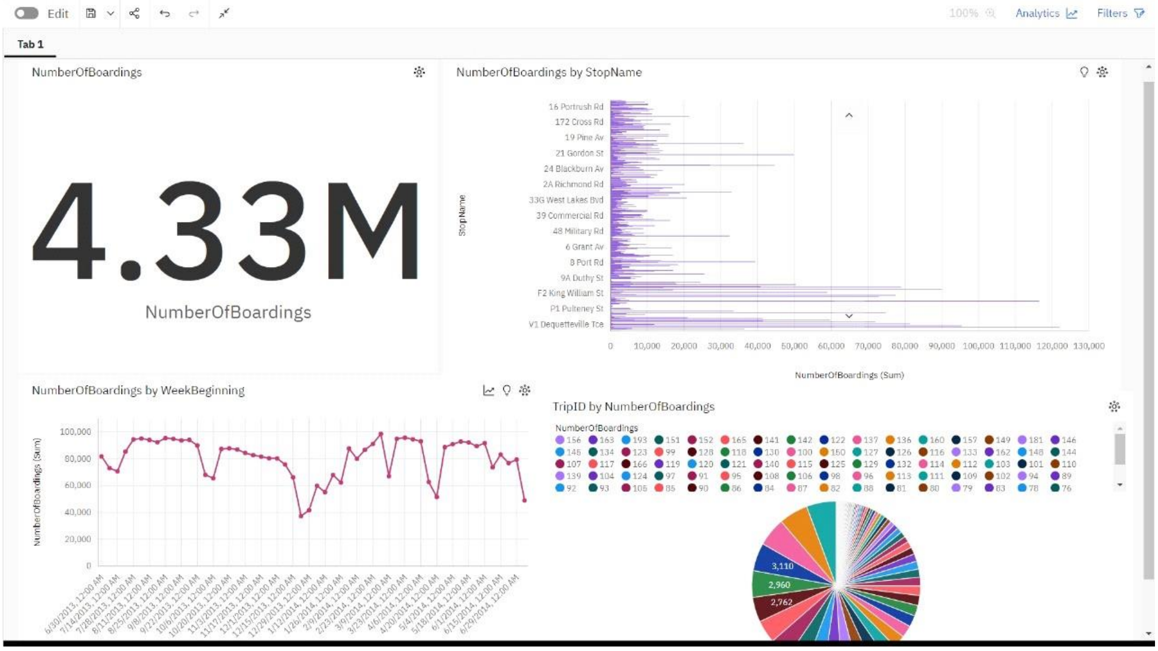
Advanced data analysis is conducted by aggregating boarding counts by RouteID, calculating average boarding counts per stop, finding stops with the highest weekly boarding counts, and analyzing trends over time.

## 6.2 Machine Learning Models:

Apply machine learning algorithms, including regression, clustering, and deep learning, to analysis the collected data. These models can be used for demand forecasting, route optimization, and predicting service disruptions.

Ensemble Learning:

Implement ensemble learning techniques to combine the predictions of multiple models, enhancing the accuracy and robustness of our analysis. Ensemble methods like Random Forests or Gradient Boosting can be particularly effective.




## 6.3 Model Interpretability and Visualization

Innovation: Explainable AI (XAI):

Incorporate Explainable AI techniques such as SHAP values and LIME to provide transparent explanations for model predictions. This helps stakeholders understand the rationale behind efficiency assessments and recommendations.

Develop an interactive dashboard with visualizations that showcase key performance indicators, route efficiency scores, and passenger sentiment trends. This user-friendly interface ensures that stakeholders can easily access and interpret the analysis results.

\

# 7.Supporting Transportation Improvement Initiatives:

The insights derived from this analysis can support transportation improvement initiatives by providing data-driven information on various aspects of public transport efficiency.

These insights may help in making decisions related to route planning, scheduling, and resource allocation. For example, understanding passenger boardings and on-time performance can lead to optimized transportation services, reduced congestion, and improved overall quality of transportation services.

The information can be valuable for transportation planners and decision-makers to enhance the efficiency of public transport systems.

## 7.1 Route Optimization:

By analyzing data on passenger boardings and ridership patterns, transportation authorities can identify high-demand routes and underutilized ones. This information can help them optimize routes, add more services to popular routes, and reallocate resources to better serve passengers.

## 7.2 Scheduling Improvements:

Data on on-time performance and delays can be used to refine and improve transportation schedules. Timely arrivals and departures are critical for public transport systems, and by identifying the causes of delays, transportation authorities can work to minimize them.

## 7.3 Resource Allocation:

With insights into passenger demographics and travel patterns, authorities can allocate resources more effectively. This might involve deploying more buses or trains during peak hours, increasing the frequency of service on specific routes, or adjusting staffing levels based on demand.

### 7.3.1 Cost Efficiency:

Data-driven decision-making can also lead to cost savings for transportation agencies. By eliminating underperforming routes or reallocating resources more efficiently, agencies can operate with a reduced budget while maintaining or even improving service quality.

### 7.3.2 Environmental Benefits:

A more efficient public transportation system can have a positive impact on the environment. It can reduce the number of individual vehicles on the road, leading to lower greenhouse gas emissions and improved air quality in urban areas.

## 7.4 Safety Enhancements:

Analyzing data can help identify potential safety issues in the public transport system. For example, if there are areas with a high incidence of accidents or security concerns, authorities can take measures to improve safety for passengers and employees.

In summary, data-driven insights derived from the analysis of public transportation data can play a crucial role in enhancing the efficiency, quality, and sustainability of public transport systems. This, in turn, can lead to better mobility options, reduced congestion, and a more pleasant and environmentally friendly urban environment.

# 8.Conclusion:

The project concludes by summarizing the data analysis work, emphasizing the use of visualization libraries like Matplotlib and Seaborn, and the application of data-driven techniques for understanding public transport efficiency

In this project, we embarked on a comprehensive journey to understand and optimize public transport efficiency through data analysis. By employing a structured approach and leveraging powerful data analysis tools, we've unveiled insights and established a foundation for data-driven decision-making within the public transport sector.

Throughout the project, we've emphasized the importance of data preprocessing as a critical step. It is essential for refining and enhancing the quality of the data, which, in turn, paves the way for more accurate predictions and insights. These insights have the potential to support a wide range of transportation improvement initiatives, ultimately benefiting commuters and urban development.

This framework will not only be a valuable resource for urban planners and transit agencies but will also contribute to the advancement of data-driven decision-making in public transportation.

Through the fusion of cutting-edge technologies and methodologies, our ultimate goal is to provide a comprehensive and insightful solution for evaluating and enhancing public transport efficiency.

.