PHASE 4

PUBLIC TRANSPORT EFFICIENCY ANALYSIS

INTRODUCTION:

Public transport efficiency analysis is a critical process that evaluates the effectiveness and sustainability of public transportation systems within a specific region or city. This examination aims to assess various aspects, including operational performance, environmental impact, accessibility, and economic viability. By conducting a comprehensive analysis, policymakers, urban planners, and transportation authorities can make informed decisions to improve public transit services and enhance the overall quality of life for residents. In this analysis, key factors such as ridership levels, cost-effectiveness.

- ✓ Feature engineering is a machine learning technique that leverages data to create new variables that aren’t in the training set. It can produce new features for both supervised and unsupervised learning, with the goal of simplifying and speeding up data transformations while also enhancing model accuracy**.** Feature engineering is required when working with machine learning models. Regardless of the data or architecture, a terrible feature will have a direct impact on your model.

Feature Selection:

- ✓ Identify the most important features and remove irrelevant or redundant ones to simplify the model.

Feature Scaling:

- ✓ Normalize or standardize numerical features to ensure they have similar scales, which can improve the performance of models like gradient-based algorithms.

Encoding Categorical Variables:

- ✓ Convert categorical variables into numerical representations. Common methods include one-hot encoding, label encoding, and binary encoding.

Handling Missing Data:

- ✓ Decide how to deal with missing values in your dataset, either by imputing them with reasonable values or marking them as a separate category.

Creating Interaction Terms:

- ✓ Combine two or more features to capture their interaction effects, which can be crucial in some modeling scenarios.

Polynomial Features:

- ✓ Generate polynomial features by squaring, cubing, or using other transformations on numerical features, which can capture non-linear relationships.

Time-based Features:

- ✓ Extract meaningful information from date and time data, such as day of the week, month, or time elapsed since a specific event.

Feature Aggregation:

- ✓ Aggregate information across multiple rows or time periods to create summary features, like means, sums, or counts.

Text Feature Engineering:

- ✓ For natural language processing tasks, create features from text data, such as word counts, TF-IDF values, or word embeddings.

Domain-Specific Features:

- ✓ Engineer features that are specific to the problem domain, drawing on your domain knowledge.

Feature Scaling:

- ✓ Scale features so that they have similar ranges, which can improve the performance of certain algorithms.

Feature Extraction:

- ✓ Use dimensionality reduction techniques like Principal Component Analysis (PCA) or t-SNE to extract essential information from high-dimensional data.

Feature Crosses:

- ✓ Combine two or more features, particularly in the case of non-linear relationships, to create new, meaningful features.

Feature Engineering - Importance, Tools and Techniques for Machine Learning

Feature engineering is the process of selecting, manipulating, and transforming raw data into features that can be used in supervised learning. Feature engineering, in simple terms, is the act of converting raw observations into desired features using statistical or machine learning ap

PROGRAM:

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import datetime
import os
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
import lightgbm as lgb
import xgboost as xgb
from sklearn.metrics import mean_squared_error
from math import sqrt
import warnings
warnings.filterwarnings('ignore')
print(os.listdir("../input/unisys/ptsboardingsummary"))
```

```
route.head(2)
out_geo.head(2)
```

Out[7]:

	route_id	agency_id	route_short_name	route_long_name	route_desc	route_type	route_url	route_color	route_text_color	Route Group
0	100	5	100	Arndale Centre Interchange to Glen Osmond	via Woodville Road, Holbrooks Road, Marion Roa...	3	http://www.adelaidemetro.com.au/routes/100	0033CC	ffffff	100-101
1	100B	5	100B	Arndale Centre Interchange / Urrbrae Agricultu...	via Kingswood, Hawthorn, Edwardstown, North Pl...	3	http://www.adelaidemetro.com.au/routes/100B	0033CC	ffffff	100-101

Out[7]:

	accuracy	formatted_address	google_place_id	input_string	latitude	longitude	number_of_results	postcode	status	type
0	ROOFTOP	181 Cross Rd, Westbourne Park SA 5041, Australia	ChIJKT7I9rbPsGoRVHMHkIy-Oyk	181 Cross Rd	-34.966656	138.592148	1	5041	OK	street_address
1	ROOFTOP	177 Cross Rd, Westbourne Park SA 5041, Australia	ChIJ-VFZ87bPsGoRyfVgC5qbPpE	177 Cross Rd	-34.966607	138.592301	1			

```
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
```

In [9]:

```
out_geo['dist_from_centre'] = out_geo[['latitude','longitude']].apply(lambda x: c
alc_dist(*x), axis=1)
```

In [10]:

```
out_geo['type'].fillna('street_address',inplace=True)
out_geo['type'] = out_geo['type'].apply(lambda x: str(x).split(',')[-1])
```

In [11]:

```
out_geo['type'].unique()
```

Out[11]:

```
 array(['street_address', 'transit_station', 'premise', 'political',
'school', 'route', 'intersection', 'point_of_interest',
'subpremise', 'real_estate_agency', 'university', 'travel_agency',
'restaurant', 'supermarket', 'store', 'post_office'], dtype=object)
```

```
fig,axrr=plt.subplots(2,2,figsize=(15,15))

ax=axrr[0][0]
ax.set_title("No of Boardings")
data['NumberOfBoardings'].value_counts().sort_index().head(20).plot.bar(ax=axrr[
0][0])

ax=axrr[0][1]
ax.set_title("WeekBeginning")
data['WeekBeginning'].value_counts().plot.area(ax=axrr[0][1])

ax=axrr[1][0]
ax.set_title("most Busiest Route")
data['RouteID'].value_counts().head(10).plot.bar(ax=axrr[1][0])

ax=axrr[1][1]
ax.set_title("least Busiest Route")
data['RouteID'].value_counts().tail(10).plot.bar(ax=axrr[1][1])
```

```
Text(0.5,1,'No of Boardings')
```

Out[27]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff880af0940>
```

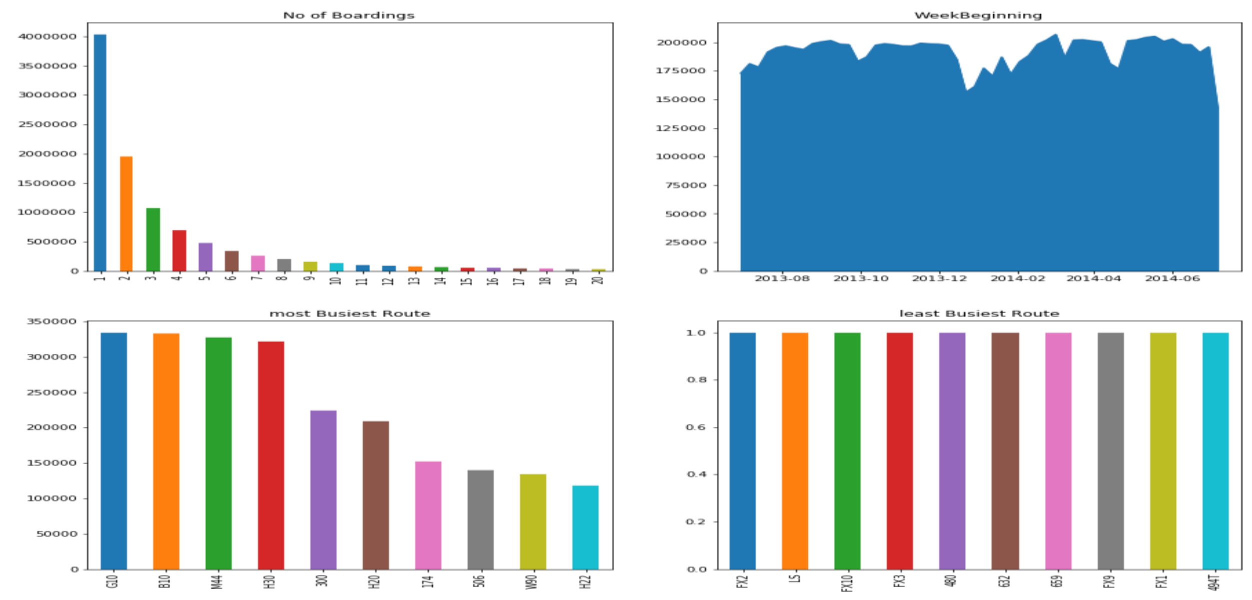
Out[27]:

```
Text(0.5,1,'WeekBeginning')
```

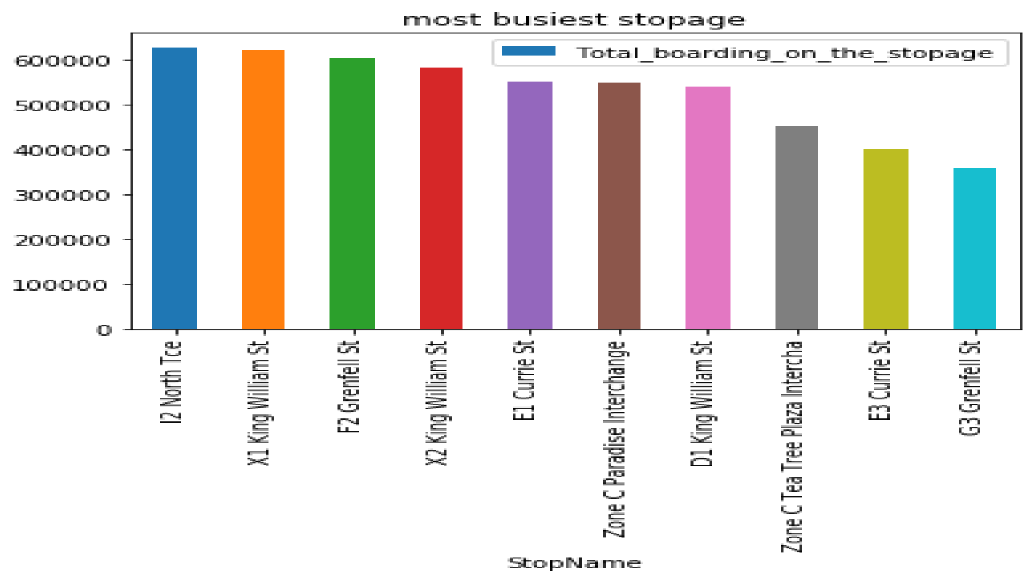
Out[27]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff709a6bb38>
```

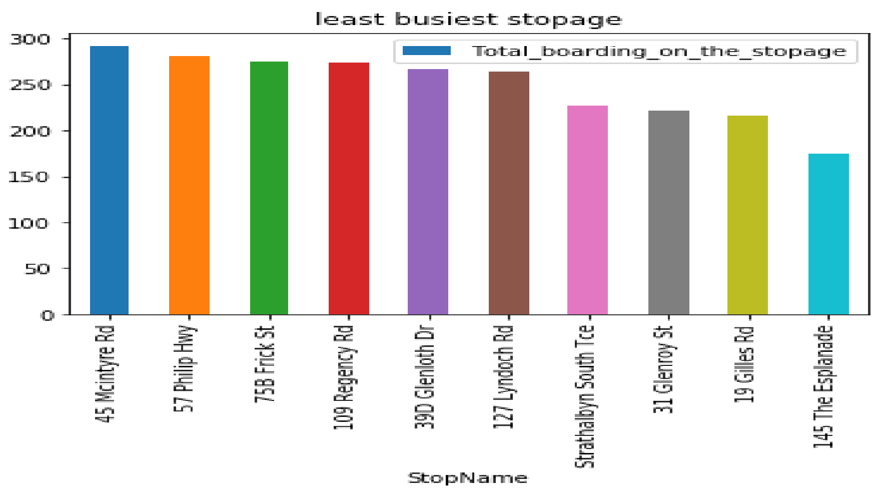
```
Out[27]:

Text(0.5,1,'most Busiest Route')

Out[27]:

<matplotlib.axes._subplots.AxesSubplot at 0x7ff709a48e10>

Out[27]:

Text(0.5,1,'least Busiest Route')

Out[27]:

<matplotlib.axes._subplots.AxesSubplot at 0x7ff736bbafd0>
```



```
ax = stopageName_with_boarding.head(10).plot.bar(x='StopName', y='Total_boarding_on_the_stopage', rot=90)
ax.set_title("most busiest stopage")
```



```
ax = stopageName_with_boarding.tail(10).plot.bar(x='StopName', y='Total_boardi
ng_on_the_stopage', rot=90)
ax.set_title("least busiest stopage")
```



```
bb_grp = data.groupby(['dist_from_centre']).agg({'NumberOfBoardings': ['sum']}).
reset_index()
bb_grp.columns = bb_grp.columns.get_level_values(0)
bb_grp.head()
bb_grp.columns
bb_grp.tail()
```

Out[33]:

	dist_from_centre	NumberOfBoardings
0	0.000018	1892435
1	0.131368	167535
2	0.309089	356518
3	0.314937	1484824
4	0.326005	120061

Out[33]:

```
Index(['dist_from_centre', 'NumberOfBoardings'], dtype='object')
```

Out[33]:

	dist_from_centre	NumberOfBoardings
2392	86.471064	18905
2393	94.826409	321

	dist_from_centre	NumberOfBoardings
2394	99.625655	1101
2395	99.665190	4373
2396	99.748995	21216

In [34]:
```
            import plotly.graph_objs as go
            from plotly.offline import iplot

trace0 = go.Scatter(
 x = bb_grp['dist_from_centre'],
 y = bb_grp['NumberOfBoardings'],mode = 'lines+markers',name = 'X2 King Willi
am St')

data1 = [trace0]
layout = dict(title = 'Distance Vs Number of boarding',
             xaxis = dict(title = 'Distance from centre'),
             yaxis = dict(title = 'Number of Boardings'))
fig = dict(data=data1, layout=layout)
iplot(fig)
```

In [35]:
```
          x = data["dist_from_centre"]
          distance_10 = []
          distance_10_50 = []
          distance_50_100 = []
          distance_100_more = []
          total = 0
          outlier = []
```

```
        outlier_ = 0
  for i in x:
        if(i<=10):
            distance_10.append(i)
             total += 1
         elif(i<=50):
            distance_10_50.append(i)
            total += 1
        elif(i<=100):
           distance_50_100.append(i)
           total += 1
```

In [36]:

```
        print(outlier_)
        0
```

In [37]:

```
         y = len(distance_10)+len(distance_10_50)+len(distance_50_100)
```

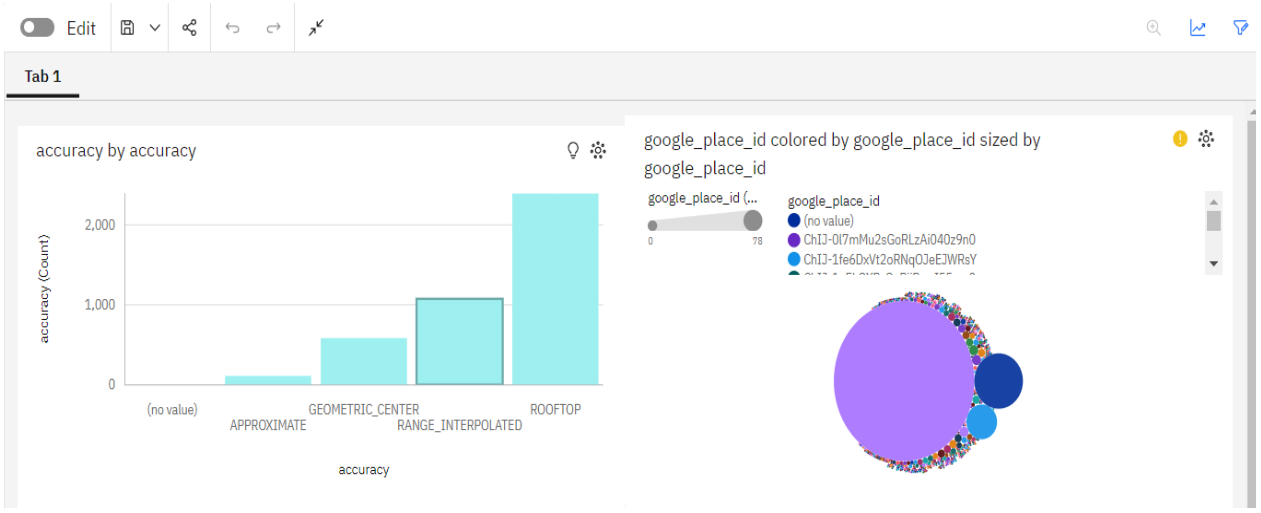
In [38]:

```
        print(total)
        print("passangers, boarding the buses in the radious of 10Km from the city c
enter = ", (len(distance_10)/total)*100)

         print("passanger, boarding the buses from the distance of 10Km to 50Km fr
om the city center = ", (len(distance_10_50)/total)*100)

         print("passanger, boarding the buses from the distance of 50Km to 100 fro
m the city center = ", (len(distance_50_100)/total)*100)
        10341468

         passangers, boarding the buses in the radious of 10Km from the city center
64.31275521038212

         passanger, boarding the buses from the distance of 10Km to 50Km from the
city  center =  33.16731241638035

       passanger, boarding the buses from the distance of 50Km to 100 from the
       city enter =  2.5199323732375327
```

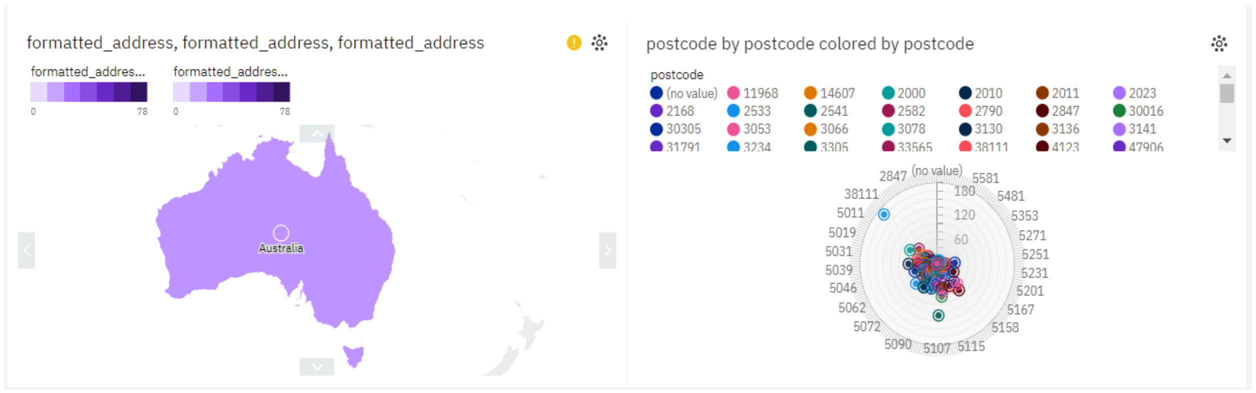
DATA VISULIZATION :

- ✓ These tools are used to represent your data through charts, graphs, and maps t hat allow you to find patterns and trends in the data.
- ✓ datapine's already mentioned BI platform also offers a wealth of powerful onl ine data visualization tools with several benefits. Some of them include: deliv ering compelling data-driven presentations to share with your entire company

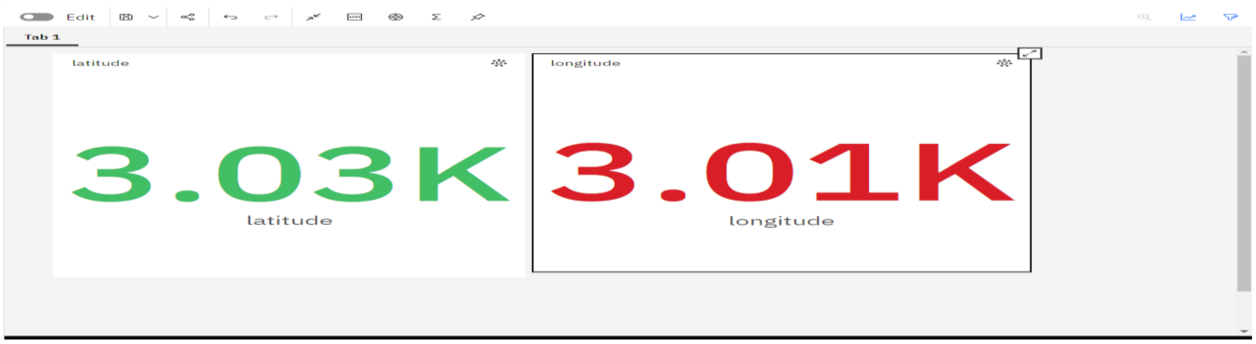
1.



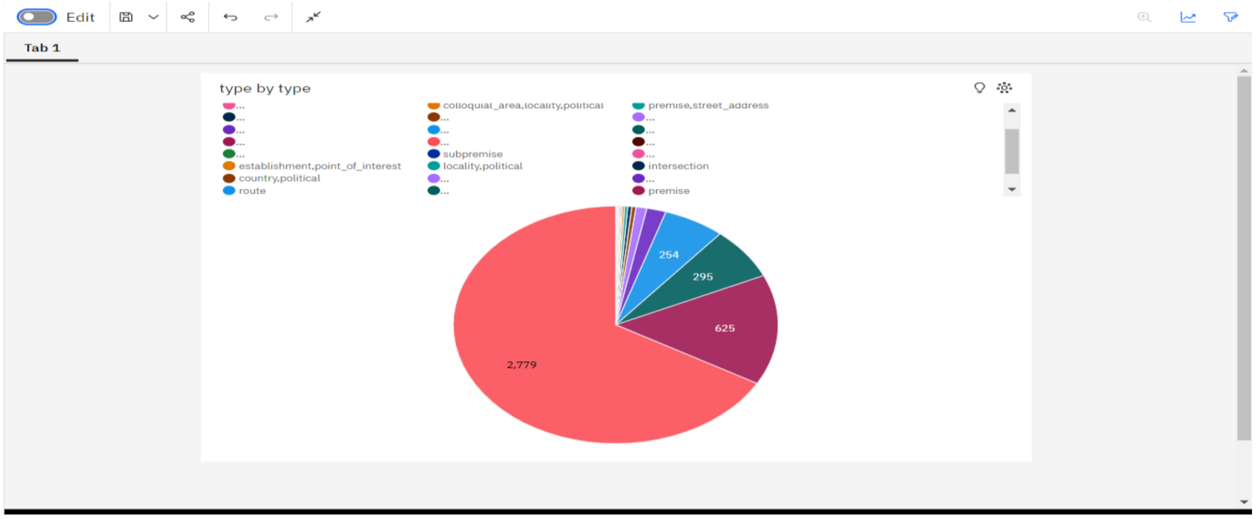
2.



3.



4.



Conclusion:

- ✓ Data collection is an essential part of the research process, whether you're conducting scientific experiments, market research, or surveys. The methods and tools used for data collection will vary depending on the research type, the sample size required, and the resources available.