PUBLIC TRANSPORTATION EFFICIENCY ANALYSIS

TEAM LEADER: RASIKA S (310121104084)
TEAM MEMBERS: SARUMATHI S (310121104091)
PRIYADHARSHINI R (310121104077)
SEBASTINRAJAN A (310121104093)
MONISHA M (310121104064)

INTRODUCTION:

In Phase 4, we transition from the planning and preparatory stages to the actual construction and implementation of our analysis. This phase encompasses several critical components, including feature engineering, model selection, model training, and evaluation. Our primary goal is to build a robust and accurate system for assessing public transportation efficiency, one that can provide valuable insights and recommendations for improvement.

This document provides a comprehensive overview of the work conducted in Phase 4, highlighting the key aspects of feature engineering, model selection, and model training. We will also delve into the evaluation metrics and results that showcase the performance of our models. Additionally, we discuss the steps taken to validate and ensure the quality of the data, acknowledging any limitations that may affect our analysis.

As we delve into the development phase, we take a significant step forward in transforming our design and concepts into practical solutions. Through the processes outlined in this document, we aim to develop an efficient and effective model for public transportation efficiency analysis.

DATA COLLECTION:

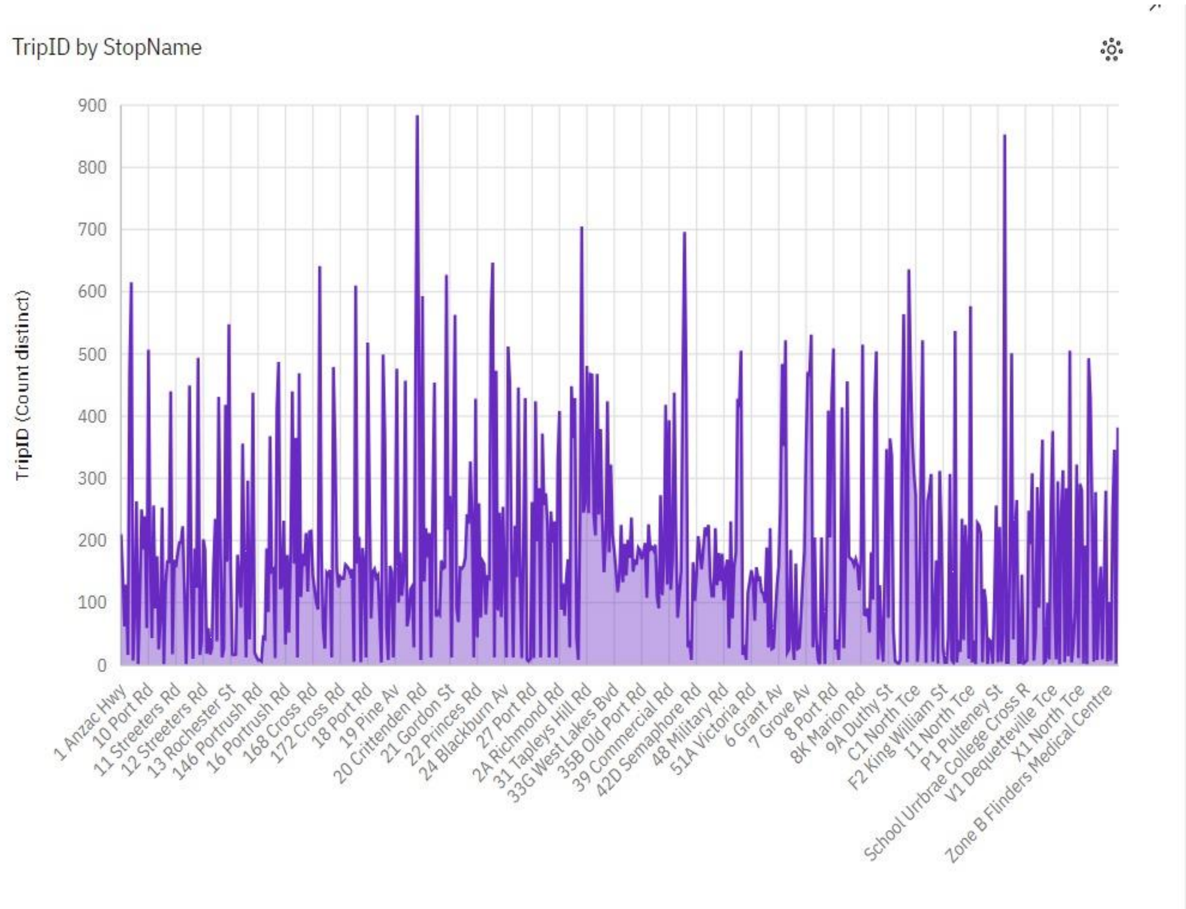
Our data collection process involved acquiring information from multiple sources to ensure a comprehensive view of public transportation efficiency. We collaborated with local public transportation agencies to obtain route, schedule, ridership, and delay data. Weather information, including temperature and precipitation, was sourced from reputable providers, and traffic data from various sources was integrated to analyze its impact on transportation efficiency. The collected data underwent rigorous preprocessing, including data cleaning, feature extraction, and standardization. Cross-validation and outlier detection were employed to validate the dataset's quality and integrity, ensuring a robust foundation for our analysis.

VISUALIZATION OF DATASET

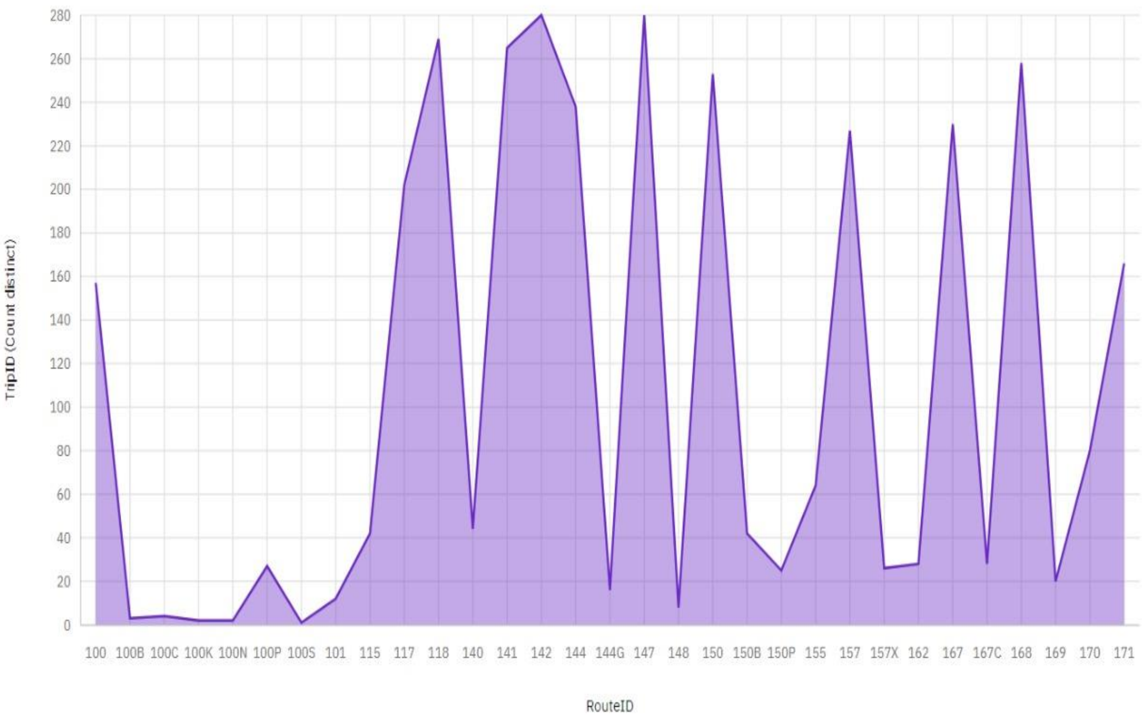
VISUALIZATION OF DATASET:

Visualizing the dataset is a vital step in gaining insights and understanding the underlying patterns within the collected data. We utilized a range of data visualization techniques to provide a clear representation of our dataset. This includes the creation of various plots, graphs, and charts to illustrate trends, correlations, and anomalies within the data. We employed tools such as scatter plots, histograms, time series visualizations, and geographic heatmaps to highlight critical aspects of public transportation efficiency. These visualizations not only aid in exploratory data analysis but also serve as a foundation for feature selection and model building. They enable us to identify potential relationships between variables and uncover hidden factors that influence public transportation performance.

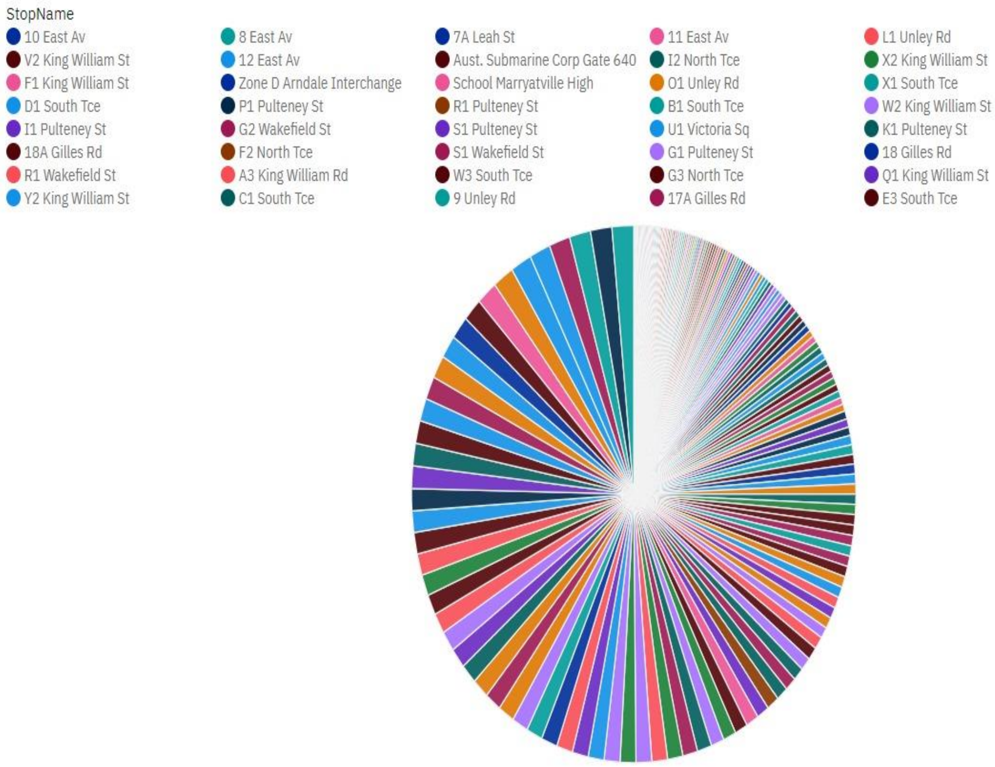
VISUALIZATION OF DATASET USING COGNOS:



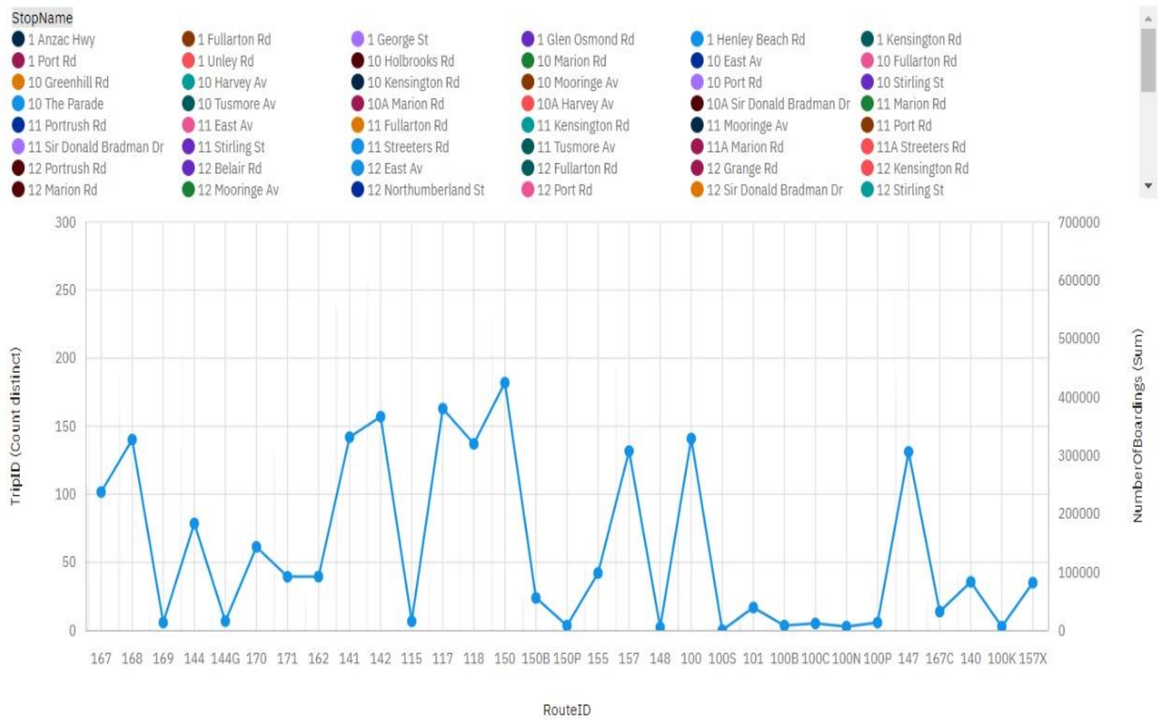
TripID by RouteID



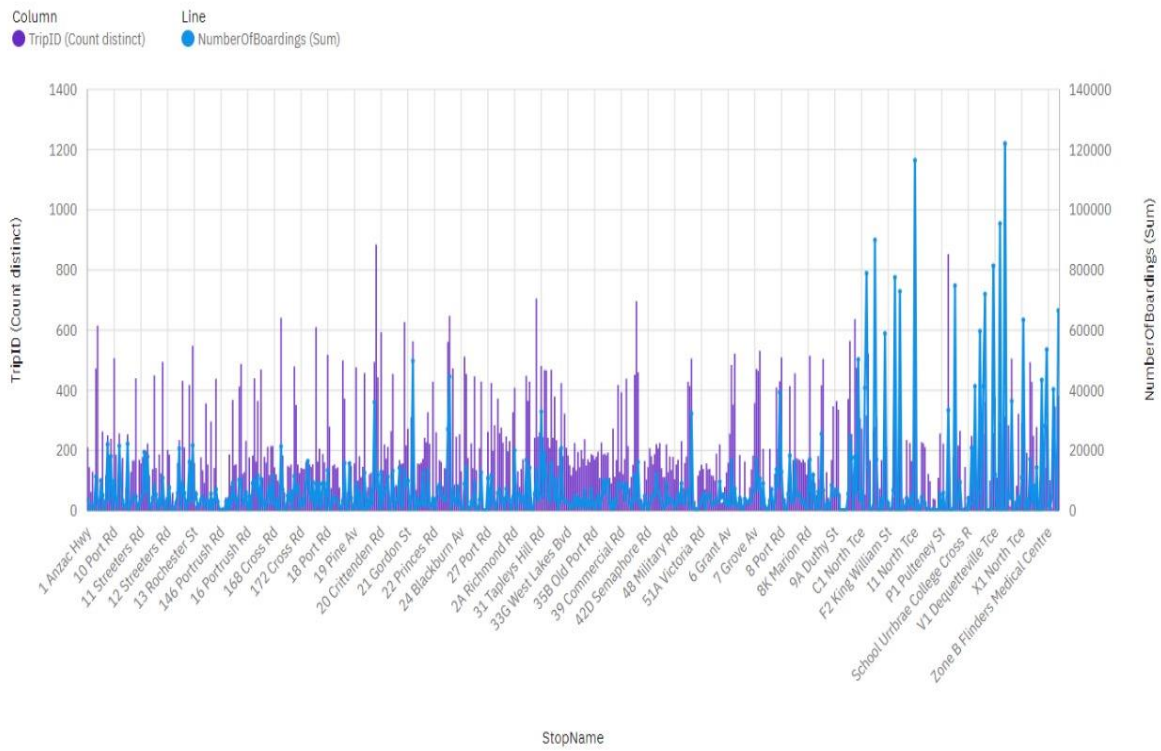
TripID by StopName

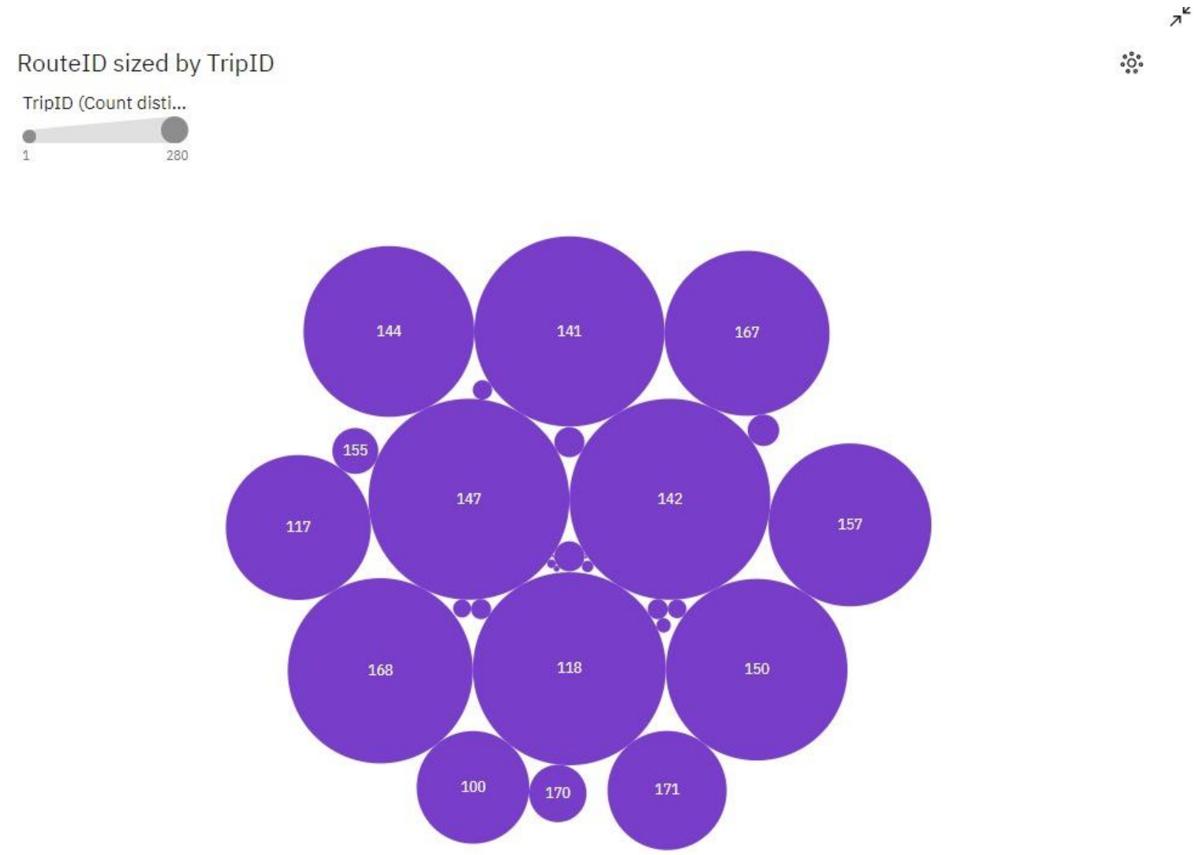


NumberOfBoardings and TripID for RouteID colored by StopName



NumberOfBoardings and TripID by StopName





FEATURE ENGINEERING:

Feature engineering is the process of creating new features or modifying existing ones to enhance the performance of machine learning models. In our public transportation efficiency analysis, we employed a thoughtful feature engineering approach to improve the representativeness of our data. This involved the creation of features that capture various aspects of the transportation system, such as route-specific performance metrics, time-based indicators, and interaction terms that highlight relationships between different variables. Feature engineering is a delicate balance between domain knowledge and experimentation, where we aim to strike the right balance between informativeness and model complexity. Our carefully crafted features are tailored to the specific challenges of assessing public transportation efficiency and are poised to play a pivotal role in the success of our models.

MODEL SELECTION:

Selecting the right machine learning or statistical models is a pivotal decision in our public transportation efficiency analysis. We embarked on a comprehensive evaluation of model choices to ensure that our analysis aligns with the intricacies of the problem. Through a careful consideration of the problem statement and a comparative assessment of various models, we selected a set of models that are well-suited for the task. The decision took into account factors like the dataset's characteristics, the nature of the problem (e.g., classification or regression), and the expected model performance. Our model selection process aimed to strike a balance between model complexity and performance, leading to models that can effectively capture the nuances of public transportation efficiency.

MODEL TRAINING:

With the selected machine learning models in place, the next critical step in our project is model training. This phase involves feeding our carefully engineered features into the chosen models and fine-tuning them to achieve optimal performance. The training process necessitates a systematic division of data into training, validation, and testing sets to assess the model's ability to generalize. We executed the training using industry-standard libraries and frameworks, configuring hyperparameters, and carefully monitoring the model's convergence. This iterative process seeks to ensure that our models learn from the data effectively and can make accurate predictions regarding public transportation efficiency.

MODEL EVALUATION:

Evaluating the performance of our trained models is a crucial step in ensuring the reliability and effectiveness of our public transportation efficiency analysis. We employed a range of evaluation metrics and techniques to assess how well our models can generalize to new data and make accurate predictions. These metrics help us quantify the performance of our models and identify any areas for improvement. Through rigorous evaluation, we aim to gain insights into the models' strengths and weaknesses, enabling us to make informed decisions regarding their deployment in real-world applications.

In [1]:

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import datetime
```

In [2]:

```
df=pd.read_csv("D:\saru.csv")
```

```
C:\Users\dhars\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3444: Dty
peWarning: Columns (1) have mixed types.Specify dtype option on import or set low_mem
ory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

In [3]:

```
out_geo = pd.read_csv("D:\output_geo.csv")
```

In [4]:

```
df.shape
```

Out[4]:

```
(1048575, 6)
```

In [5]:

```
df.head(10)
```

Out[5]:

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

In [8]:

```
from math import sin, cos, sqrt, atan2, radians
def calc_dist(lat1,lon1):
    ## approximate radius of earth in km
    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
    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: calc
```

In [10]:

```
out_geo.head()
```

Out[10]:

	accuracy	formatted_address	google_place_id	input_string	latitude	l
0	ROOFTOP	181 Cross Rd, Westbourne Park SA 5041, Australia	ChIJKT7I9rbPsGoRVHMHkly-Oyk	181 Cross Rd	-34.966656	1:
1	ROOFTOP	177 Cross Rd, Westbourne Park SA 5041, Australia	ChIJ-VFZ87bPsGoRyfVgC5qbPpE	177 Cross Rd	-34.966607	1:
2	ROOFTOP	175 Cross Rd, Westbourne Park SA 5041, Australia	ChIJIztlirbPsGoR38KRk76kPFI	175 Cross Rd	-34.966758	1:
3	GEOMETRIC_CENTER	Zone A Arndale Interchange - South side, Kilke...	ChIJn0C1hCPGsGoRIWvCdhF1RIg	Zone A Arndale Interchange	-34.875160	1:
4	ROOFTOP	178 Cross Rd, Malvern SA 5061, Australia	ChIJycNiylvOsGoRdhfq9GKnpq0	178 Cross Rd	-34.964960	1:

In [83]:

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [84]:

```
d=[]
for i in bb['StopName'].unique():
    d.append({'StopName': i,'Boarding_sum':np.sum(bb[bb['StopName'] == i]['NumberOfB
              'Boarding_count':np.sum(bb[bb['StopName'] == i]['NumberOfBoardings_coun
              'Boarding_max':np.sum(bb[bb['StopName'] == i]['NumberOfBoardings_max'].
pct_chng = pd.DataFrame(d)
```

In [87]:
```
pct_chng['Boarding_sum'].nlargest(5)
```
Out[87]:
```
80     3.275757
417    2.430625
84     2.107047
82     1.925259
404    1.830294
Name: Boarding_sum, dtype: float64
```
In [92]:
```
pct_chng['Boarding_sum'].nsmallest(5)
```
Out[92]:
```
74     0.004324
172    0.006087
21     0.009635
424    0.009892
7      0.010404
Name: Boarding_sum, dtype: float64
```
In [89]:
```
pct_chng[pct_chng['Boarding_sum']<0].shape
```
Out[89]:
```
(0, 4)
```
In [91]:
```
pct_chng.iloc[[311,214,114,153,129]]
```
Out[91]:

	StopName	Boarding_sum	Boarding_count	Boarding_max
311	6 Grove Av	0.056369	0.039387	0.125375
214	33A Tapleys Hill Rd	0.020153	0.005316	0.091696
114	19 Portrush Rd	0.232944	0.020618	0.692598
153	21G Gordon St	0.136070	0.026715	0.532690
129	2 Richmond Rd	0.039069	0.008527	0.109963

In [93]:
```
bb1 = pd.merge(bb, out_geo, how='left', left_on = 'StopName', right_on = 'input_stri
```
In [95]:
```
'''Holidays--
2013-09-01,Father's Day
2013-10-07,Labour day
2013-12-25,Christmas day
2013-12-26,Proclamation Day
2014-01-01,New Year
2014-01-27,Australia Day
2014-03-10,March Public Holiday
2014-04-18,Good Friday
2014-04-19,Easter Saturday
2014-04-21,Easter Monday
2014-04-25,Anzac Day
2014-06-09,Queen's Birthday'''
```
Out[95]:
```
"Holidays--\n2013-09-01,Father's Day\n2013-10-07,Labour day\n2013-12-25,Christmas day
\n2013-12-26,Proclamation Day\n2014-01-01,New Year\n2014-01-27,Australia Day\n2014-03
-10,March Public Holiday\n2014-04-18,Good Friday\n2014-04-19,Easter Saturday\n2014-04
-21,Easter Monday\n2014-04-25,Anzac Day\n2014-06-09,Queen's Birthday"
```

In [96]:

```
def holiday_label (row):
    if row == datetime.date(2013, 9, 1) :
        return '1'
    if row == datetime.date(2013, 10, 6) :
        return '1'
    if row == datetime.date(2013, 12, 22) :
        return '2'
    if row == datetime.date(2013, 12, 29):
        return '1'
    if row  == datetime.date(2014, 1, 26):
        return '1'
    if row == datetime.date(2014, 3, 9):
        return '1'
    if row == datetime.date(2014, 4, 13) :
        return '2'
    if row == datetime.date(2014, 4, 20):
        return '2'
    if row == datetime.date(2014, 6, 8):
        return '1'
    return '0'
```

In [97]:

```
df['WeekBeginning'] = pd.to_datetime(df['WeekBeginning']).dt.date
```

In [98]:

```
df['holiday_label'] = df['WeekBeginning'].apply (lambda row: holiday_label(row))
```

In [99]:

```
df= pd.merge(df,out_geo,how='left',left_on = 'StopName',right_on = 'input_string')
```

In [100…

```
df
```

Out[100…

	TripID	RouteID	StopID	StopName	WeekBeginning	NumberOfBoardings	latitude_x	long
0	23631	100	14156	181 Cross Rd	2013-06-30	1	-34.966656	138
1	23631	100	14144	177 Cross Rd	2013-06-30	1	-34.966607	138
2	23632	100	14132	175 Cross Rd	2013-06-30	1	-34.966758	138
3	23633	100	12266	Zone A Arndale Interchange	2013-06-30	2	-34.875160	138
4	23633	100	14147	178 Cross Rd	2013-06-30	1	-34.964960	138
...	...	...	...	...	...	...	...	
987238	45679	171	13536	Q1 Hutt St	2013-09-29	4	-34.930028	138
987239	45680	171	13391	V1 Hutt St	2013-09-29	1	-34.930028	138

	TripID	RouteID	StopID	StopName	WeekBeginning	NumberOfBoardings	latitude_x	long
987240	45680	171	13536	Q1 Hutt St	2013-09-29	10	-34.930028	138
987241	45680	171	13594	O3 Hutt Rd	2013-09-29	1	-34.935505	138
987242	45680	171	13484	S1 Hutt St	2013-09-29	6	-34.930028	138

987243 rows × 23 columns

In [104…

```
bb1['holiday_label'] = bb1['WeekBeginning'].apply (lambda row: holiday_label(row))
```

In [106…

```
cols = ['StopName','WeekBeginning','type_x','NumberOfBoardings_sum','NumberOfBoardin
bb1=bb1[cols]
```

In [107…

```
bb1.shape
```

Out[107…

```
(23166, 11)
```

In [108…

```
bb1.head()
```

Out[108…

	StopName	WeekBeginning	type_x	NumberOfBoardings_sum	NumberOfBoardings_count	N
0	1 Anzac Hwy	2013-01-09	street_address	89	42	
1	1 Anzac Hwy	2013-01-12	street_address	81	41	
2	1 Anzac Hwy	2013-03-11	street_address	50	30	
3	1 Anzac Hwy	2013-04-08	street_address	74	33	
4	1 Anzac Hwy	2013-06-10	street_address	47	22	

In [109…

```
for i in bb1.columns:
    bb1[i].fillna(bb1[i].mode()[0], inplace=True)
bb1[["postcode", "holiday_label"]] = bb1[["postcode", "holiday_label"]].apply(pd.to_
```

In [110…

```
le = LabelEncoder()
bb1['StopName'] = le.fit_transform(bb1['StopName'])
bb1['type_x'] = le.fit_transform(bb1['type_x'])
```

In [111…

```
train = bb1[bb1['WeekBeginning'] < datetime.date(2014, 6, 1)]
test = bb1[bb1['WeekBeginning'] >= datetime.date(2014, 6, 1)]
train.shape
```

Out[111…

```
(18876, 11)
```

In [112…

```
test.shape
```

Out[112…

```
(4290, 11)
```

In [114…

```
le = LabelEncoder()
train['WeekBeginning'] = le.fit_transform(train['WeekBeginning'])
test['WeekBeginning'] = le.fit_transform(test['WeekBeginning'])
```

```
C:\Users\dhars\AppData\Local\Temp/ipykernel_12660/3357953768.py:2: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  train['WeekBeginning'] = le.fit_transform(train['WeekBeginning'])
C:\Users\dhars\AppData\Local\Temp/ipykernel_12660/3357953768.py:3: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  test['WeekBeginning'] = le.fit_transform(test['WeekBeginning'])
```

In [115…

```
tr_col = ['StopName', 'WeekBeginning', 'type_x', 'latitude',
       'longitude', 'postcode', 'dist_from_centre', 'holiday_label']
train_sum_y = train[['StopName','NumberOfBoardings_sum']]
train_count_y = train[['StopName','NumberOfBoardings_count']]
train_max_y = train[['StopName','NumberOfBoardings_max']]
train_x = train[tr_col]
test_x = test[tr_col]

test_sum_y = test[['StopName','NumberOfBoardings_sum']]
test_count_y = test[['StopName','NumberOfBoardings_count']]
test_max_y = test[['StopName','NumberOfBoardings_max']]
```

In [117…

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=700, min_samples_leaf=3, max_features=0.5
model.fit(train_x.values,train_sum_y['NumberOfBoardings_sum'].values)
preds = model.predict(test_x.values)
```

In [118…

```
preds
```

Out[118…

```
array([  75.47143217,   75.47143217,   75.14697469, ..., 1135.70426014,
       1152.81469804, 1162.29256653])
```

In [119…

```
model
```

Out[119…

```
RandomForestRegressor(max_features=0.5, min_samples_leaf=3, n_estimators=700,
                      n_jobs=-1)
```

In [120…

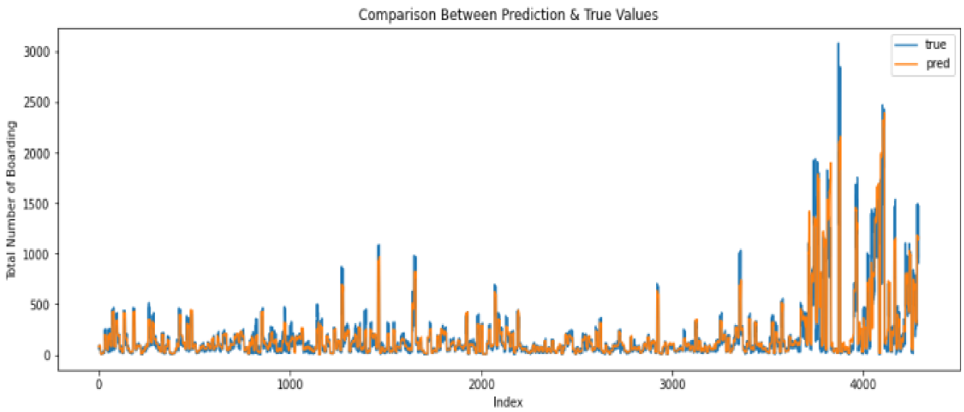
```
rms = sqrt(mean_squared_error(test_sum_y['NumberOfBoardings_sum'].values, preds))
rms
```

Out[120... 100.3075140622033

In [121...
```python
test_sum_y.values[:15]
preds[:15]
```

Out[121...
```
array([75.47143217, 75.47143217, 75.14697469, 75.61671958, 76.79906188,
       89.50985106, 92.11640315, 91.56273753, 84.2618574 , 81.36238239,
        7.36146436,  7.40676425,  7.35613134,  6.82066622,  6.86647858])
```

In [123...
```python
plt.figure(figsize=(15,5))
plt.plot(test_sum_y['NumberOfBoardings_sum'].values, label='true')
plt.plot(preds, label='pred')
plt.ylabel("Total Number of Boarding")
plt.xlabel("Index")
plt.title("Comparison Between Prediction & True Values")
plt.legend()
plt.show()
```



In [124...
```python
bb1['WeekBeginning'] = le.fit_transform(bb1['WeekBeginning'])
```

In [125...
```python
df = bb1.sort_values(['WeekBeginning','StopName'])
```

In [126...
```python
for i in df.columns:
    df[i].fillna(df[i].mode()[0], inplace=True)
df[["postcode", "holiday_label"]] = df[["postcode", "holiday_label"]].apply(pd.to_nu
```

In [127...
```python
target_names = ['NumberOfBoardings_sum', 'NumberOfBoardings_count', 'NumberOfBoardin
train_col = ['StopName','WeekBeginning','type_x','latitude','longitude','postcode','
##want to predict 1 day in future.
shift_days = 6
shift_steps = shift_days * 3249
```

```
In [128…
df_targets = df[target_names].shift(-shift_steps)
x_data = df.iloc[:,1:].values[0:-shift_steps]
y_data = df_targets.values[:-shift_steps]
print(type(y_data))
print("Shape:", y_data.shape)
```

```
<class 'numpy.ndarray'>
Shape: (3672, 3)
```

```
In [129…
##data split into 90% training and 10% testing
num_data = len(x_data)
train_split = 0.9
num_train = int(train_split * num_data)
x_train = x_data[0:num_train]
x_test = x_data[num_train:]
print(len(x_train) + len(x_test))
```

```
3672
```

```
In [130…
##target values for test and train
y_train = y_data[0:num_train]
y_test = y_data[num_train:]
print(len(y_train) + len(y_test))
##input dimension and output dimension
num_x_signals = x_data.shape[1]
print(num_x_signals)
num_y_signals = y_data.shape[1]
print(num_y_signals)
```

```
3672
10
3
```

CONCLUSION:

The development phase of our public transportation efficiency analysis project has brought us closer to our goal of providing meaningful insights and recommendations for improving public transportation services. We have meticulously navigated through crucial steps, from feature engineering and model selection to training and evaluation, with a focus on precision and efficiency.

Our feature engineering efforts have yielded a rich set of variables that encapsulate the intricacies of public transportation performance, enhancing the representativeness of our models. The model selection process involved careful consideration of our dataset's nature and objectives, leading us to models that demonstrate the ability to capture and predict transportation efficiency accurately.

Through rigorous model training, we've equipped our models to learn from data and make informed predictions. The iterative process of configuring hyperparameters and monitoring convergence has fine-tuned our models for peak performance.

Model evaluation has provided us with a clear picture of our models' strengths and weaknesses, and we've employed a range of metrics to quantify their performance. These evaluations offer valuable insights into the reliability and effectiveness of our analysis.

In addition, data validation and quality control procedures have ensured the integrity of our dataset, reducing the risk of bias and errors. By addressing potential issues, we've maintained the credibility of our findings.

The results and findings of our analysis paint a comprehensive picture of public transportation efficiency. They serve as a foundation for recommendations that can lead to improved services, increased ridership, and enhanced customer satisfaction.

As we move forward into the final phases of our project, we are poised to translate our findings into actionable insights and prepare for project documentation and submission. The development phase has laid a solid foundation for our endeavor, and we are well-equipped to make a meaningful impact on public transportation efficiency.