**PROJECT**:  **PUBLIC TRANSPORTATION ANALYSIS**

**INTRODUCTION:**

Analyzing and innovating in the field of public transportation is crucial for improving efficiency, accessibility, and sustainability. Incorporating machine learning techniques into public transportation innovation can lead to significant improvements in efficiency, safety, and passenger satisfaction.The dataset contains the passenger data, vehicle data, Traffic and Weather Data, Schedule Data, Predictive Analytics Data, Performance Metrics etc.,

**Empathize and Understand the Problem**:

Understanding the importance of analyzing public transportation in Tamil Nadu is crucial for addressing the region's specific challenges and concerns. Engage with experts, stakeholders, and potential users to gather insights into the current state of public transportation, identifying pain points and areas for improvement.

**Defining Clear Objectives:**

**Objective 1:** Analyze historical public transportation data to identify usage patterns and service efficiency.

**Objective 2:** Identify key areas with high demand for public transportation and assess the current infrastructure's capacity.

**Objective 3:** Develop a predictive model to estimate peak hours and optimize service frequency.

**Ideation and Analysis Approach:**

**Data Collection:** Identify sources of public transportation data in Tamil Nadu, such as transportation agencies or municipal records.

**Data Pre-processing:** Clean and pre-process the data, addressing issues like missing values, outliers, and data quality concerns.

**Data Analysis:** Use statistical analysis and visualization techniques to identify usage patterns and areas of improvement.

**Demand Hotspot Detection:** Develop algorithms or criteria to identify regions with high demand for public transportation.

**Predictive Modelling:** Choose a suitable machine learning algorithm to build a model predicting peak hours and optimizing service frequency.

**Evaluation:** Define metrics to evaluate the model's accuracy and effectiveness.

**Visualization Strategy:**

**Line Charts:** Display historical usage patterns over time to identify trends and fluctuations.

**Heatmaps or Geographic Maps:** Visualize demand hotspots geographically to highlight areas that require increased transportation capacity.

**Scatter Plots or Regression Plots:** Illustrate the relationship between various factors influencing service optimization, such as time of day and passenger count.

**Build and Implement:**

Develop the full data analysis and visualization pipeline based on the refined approach.

**Test and Iterate:**

Continuously test analysis and visualization progress, making adjustments and refinements based on feedback and new insights.

**Deliver Insights:** Present the findings and insights in a clear and understandable manner. Utilize the selected visualizations to communicate usage patterns, demand hotspots, and the predictive model's effectiveness.

**Integration:** Integrate the data analysis, visualization, and predictive modeling code into a cohesive pipeline for seamless execution.

**IMPLEMENTATION :**

**LOADING AND PREPROCESSING THE DATA :**

%matplotlib inline

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import matplotlib.pyplot as plt

import datetime

import os

from math import sqrt

import warnings

data = pd.read\_csv('../input/unisys/ptsboardingsummary/20140711.CSV')

data.shape

data.head(10)

**OUTPUT :**

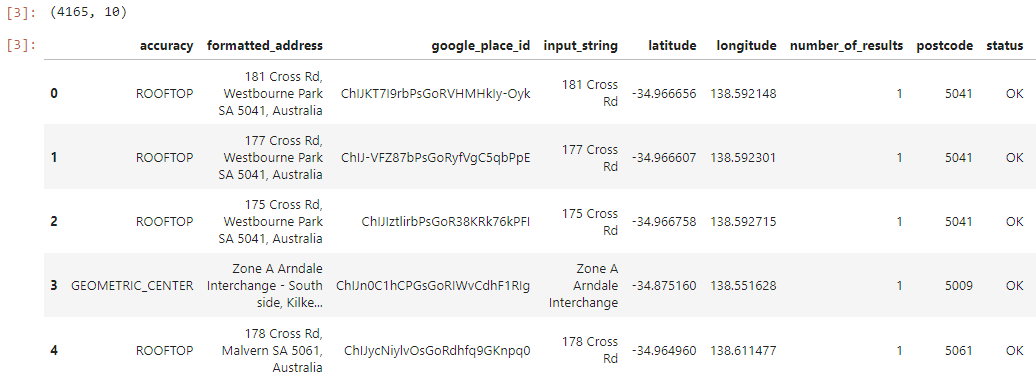


out\_geo = pd.read\_csv('../input/outgeo/output\_geo.csv')

out\_geo.shape

out\_geo.head()

**OUTPUT :**



#DistanceFromCentre: Distance measure from the city centre

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)

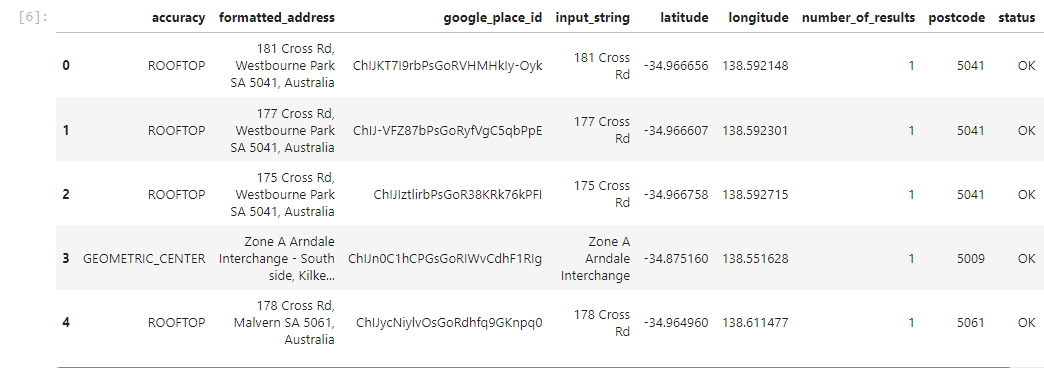
a = sin(dlat / 2)\*\*2 + cos(radians(lat1)) \* cos(radians(-34.921247)) \* sin(dlon / 2)\*\*2

c = 2 \* atan2(sqrt(a), sqrt(1 - a))

return R \* c

out\_geo['dist\_from\_centre'] = out\_geo[['latitude','longitude']].apply(lambda x: calc\_dist(\*x), axis=1)

**OUTPUT :**



#exp\_data = out\_geo.head(10)

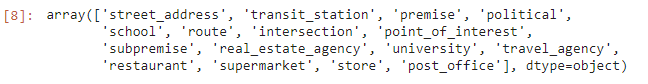
##Fill the missing values with mode

out\_geo['type'].fillna('street\_address',inplace=True)

out\_geo['type'] = out\_geo['type'].apply(lambda x: str(x).split(',')[-1])

out\_geo['type'].unique()

**OUTPUT :**



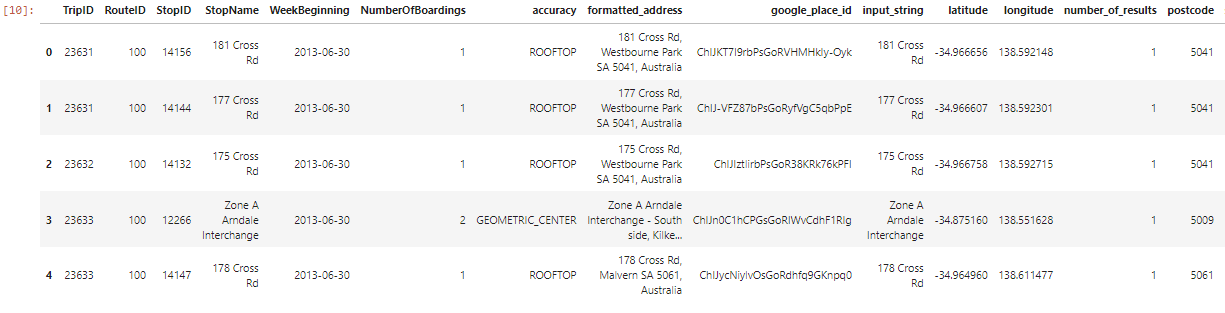
#Combine the Geolocation and main input file to get final Output File.

data= pd.merge(data,out\_geo,how='left',left\_on = 'StopName',right\_on = 'input\_string')

data.head(5)

data.shape

**OUTPUT :**





#Columns to keep for further analysis

col = ['TripID', 'RouteID', 'StopID', 'StopName', 'WeekBeginning','NumberOfBoardings',

'latitude', 'longitude','postcode','type','dist\_from\_centre']

data = data[col]

grouped = data.groupby(['StopName','WeekBeginning','type'])

#grouped.head()

# st\_week\_grp1 = pd.DataFrame(data.groupby(['StopName','WeekBeginning','type']).agg({'NumberOfBoardings': ['sum', 'count']})).reset\_index()

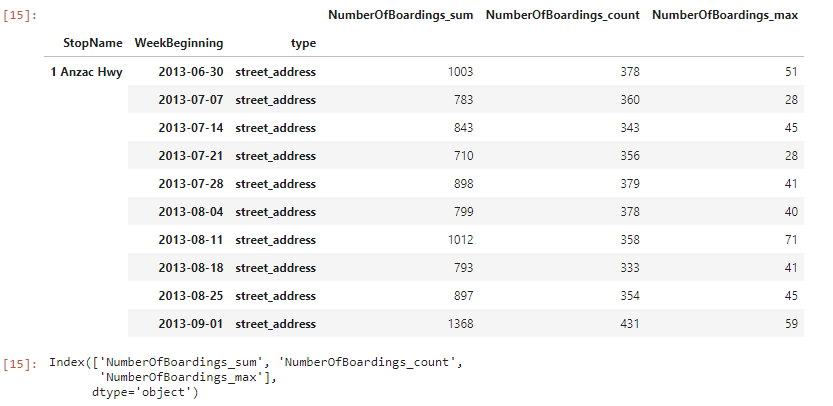
grouped = data.groupby(['StopName','WeekBeginning','type']).agg({'NumberOfBoardings': ['sum', 'count','max']})

grouped.columns = ["\_".join(x) for x in grouped.columns.ravel()]

grouped.head(10)

grouped.columns

**OUTPUT :**

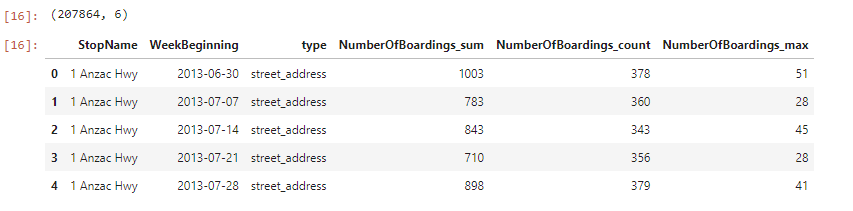


st\_week\_grp = pd.DataFrame(grouped).reset\_index()

st\_week\_grp.shape

st\_week\_grp.head()

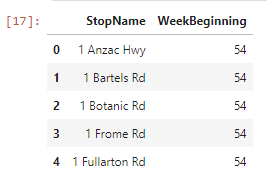
**OUTPUT :**



st\_week\_grp1 = pd.DataFrame(st\_week\_grp.groupby('StopName')["WeekBeginning"].count()).reset\_index()

st\_week\_grp1.head()

**OUTPUT :**

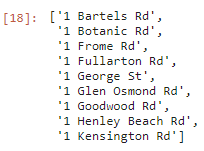


#Gathering only the Stop Name which having all 54 weeks of Dat

aa = list(st\_week\_grp1[st\_week\_grp1['WeekBeginning'] == 54]['StopName'])

Aa[1:10]

**OUTPUT :**

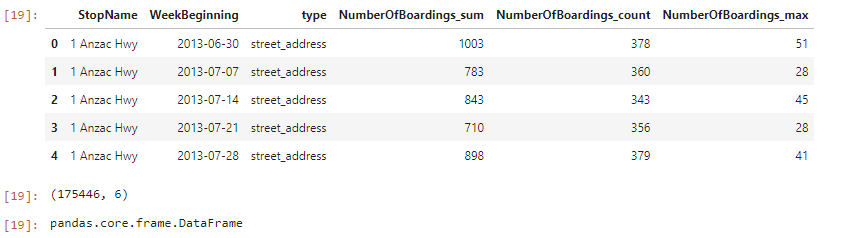


bb = st\_week\_grp[st\_week\_grp['StopName'].isin(aa)]

bb.head()

bb.shape

**OUTPUT :**



#removing the stoppage which are not having the data of whole 54 weeks

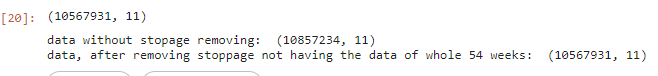
new\_data = data[data['StopName'].isin(aa)]

new\_data.shape

print("data without stopage removing: ", data.shape)

print("data, after removing stoppage not having the data of whole 54 weeks: ", new\_data.shape)

**OUTPUT :**



new\_data.head(2)

filtered\_data = new\_data[new\_data['dist\_from\_centre'] <= 100]

filtered\_data.shape

**OUTPUT :**



data = filtered\_data.copy()

data.shape

**OUTPUT :**



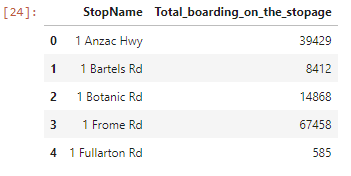
stopageName\_with\_boarding = bb.groupby(['StopName']).agg({'NumberOfBoardings\_sum': ['sum']})

stopageName\_with\_boarding = pd.DataFrame(stopageName\_with\_boarding.reset\_index())

stopageName\_with\_boarding.columns = ["StopName", "Total\_boarding\_on\_the\_stopage"]

stopageName\_with\_boarding.head()

**OUTPUT :**

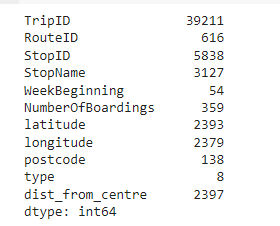


## **Data Exploration**

data.nunique()

#data.isnull().sum()

#data['WeekBeginning'].unique()



## **Data Visualization**

##can assign the each chart to one axes at a time

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]

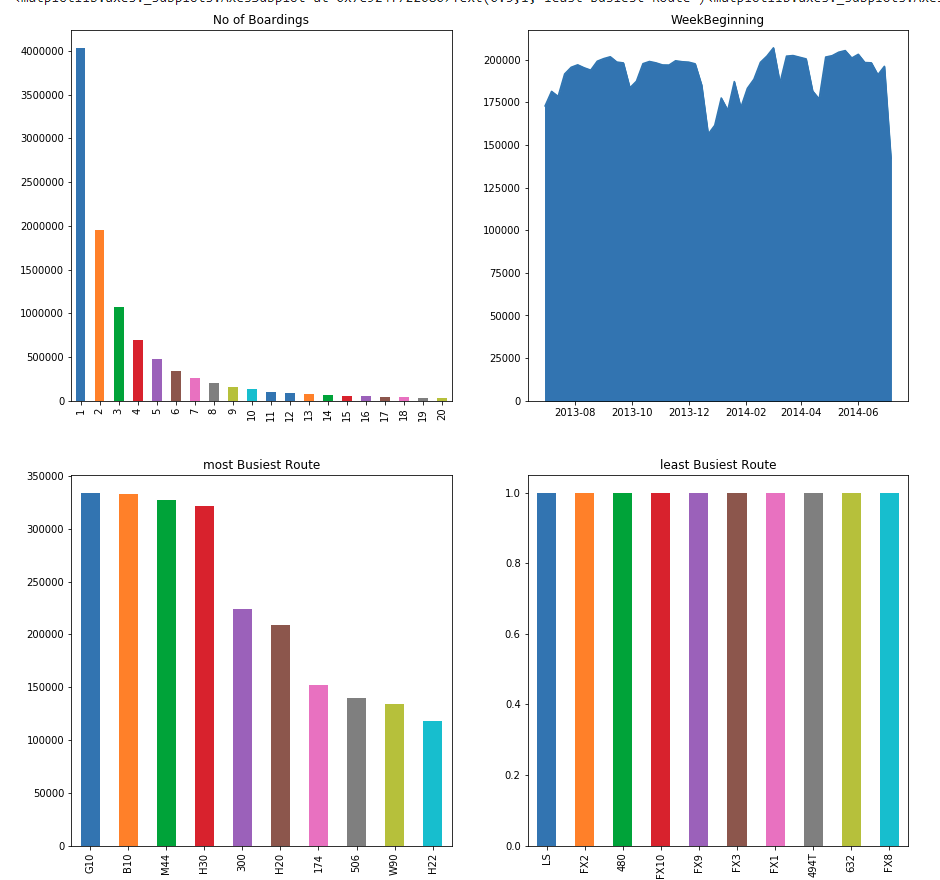
ax.set\_title("most Busiest Route")

data['RouteID'].value\_counts().head(10).plot.bar(ax=axrr[1][0])

ax=axrr[1][1]

ax.set\_title("least Busiest Route")

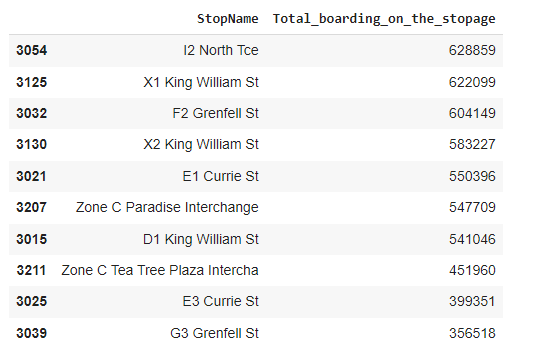
data['RouteID'].value\_counts().tail(10).plot.bar(ax=axrr[1][1])



stopageName\_with\_boarding = stopageName\_with\_boarding.sort\_values('Total\_boarding\_on\_the\_stopage', ascending = False)

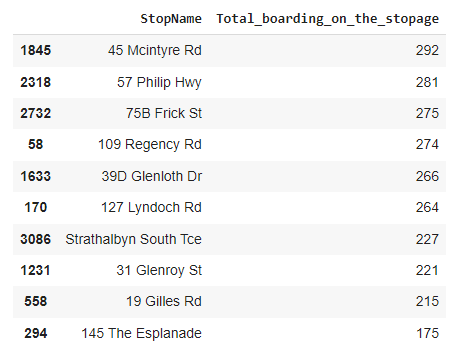
#stopage with most no of boarding

stopageName\_with\_boarding.head(10)



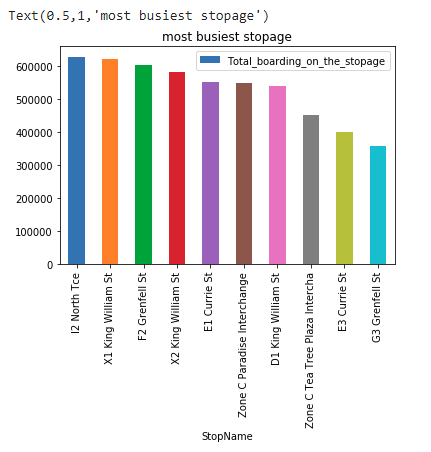
#stopage with least no of boarding

stopageName\_with\_boarding.tail(10)



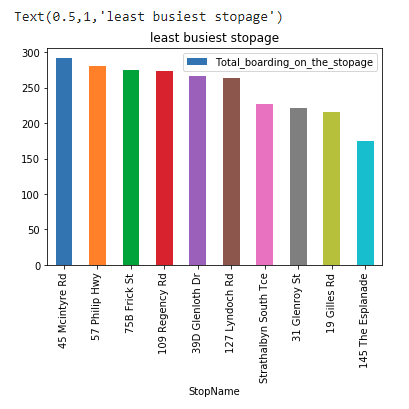
ax = stopageName\_with\_boarding.head(10).plot.bar(x='StopName', y='Total\_boarding\_on\_the\_stopage', rot=90)

ax.set\_title("most busiest stopage")



ax = stopageName\_with\_boarding.tail(10).plot.bar(x='StopName', y='Total\_boarding\_on\_the\_stopage', rot=90)

ax.set\_title("least busiest stopage")



data['WeekBeginning'].value\_counts().mean()



# data['dist\_from\_centre'].nunique()

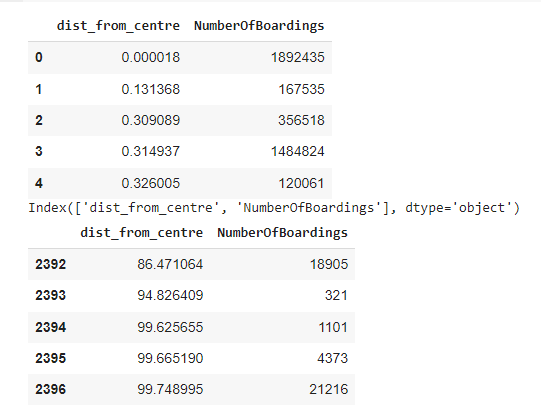
bb\_grp = data.groupby(['dist\_from\_centre']).agg({'NumberOfBoardings': ['sum']}).reset\_index()

bb\_grp.columns = bb\_grp.columns.get\_level\_values(0)

bb\_grp.head()

bb\_grp.columns

bb\_grp.tail()



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 William St')

data1 = [trace0]

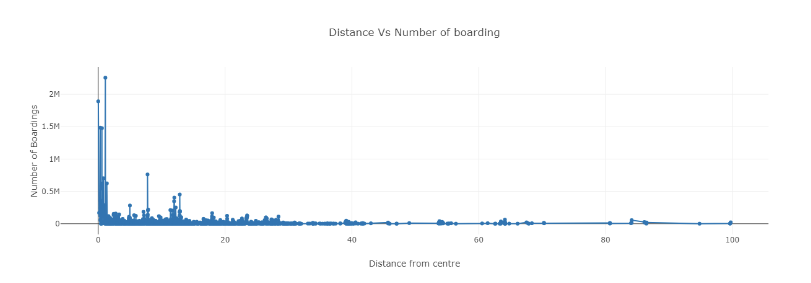
layout = dict(title = 'Distance Vs Number of boarding',

              xaxis = dict(title = 'Distance from centre'),

              yaxis = dict(title = 'Number of Boardings'))

fig = dict(data=data1, layout=layout)

iplot(fig)



#clustering Technique// based on the distance from city centre

x = data["dist\_from\_centre"]

distance\_10 = []

distance\_10\_50 = []

distance\_50\_100 = []

#distance\_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

    #elif(i>100 and i< 2000):

        #distance\_100\_more.append(i)

        #total += 1

    #elif(i>2000):

        #outlier.append(i)

        #outlier\_ += 1

print(outlier\_)

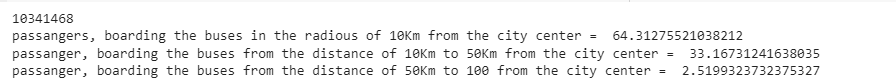


y = len(distance\_10)+len(distance\_10\_50)+len(distance\_50\_100)

print("passangers, boarding the buses in the radious of 10Km from the city center = ", (len(distance\_10)/total)\*100)

print("passanger, boarding the buses from the distance of 10Km to 50Km from the city center = ", (len(distance\_10\_50)/total)\*100)

print("passanger, boarding the buses from the distance of 50Km to 100 from the city center = ", (len(distance\_50\_100)/total)\*100)



**CONCLUSION:**

Machine learning in public transportation is an ongoing process that requires continuous learning and adaptation. By following these steps, transportation authorities can leverage technology to make public transit more efficient, responsive, and passenger-centric. Innovations in public transportation should aim to make it more efficient, sustainable, and accessible while meeting the changing needs of urban and rural populations.