**Public Transportation Analysis**

# Define Analysis Objectives:

Begin by clearly defining your analysis objectives. What specific aspects of public transportation are you aiming to improve? For example:

Assess on-time performance of buses/trains.

Evaluate passenger satisfaction and gather feedback.

Identify routes or time periods with the highest ridership.

# Data Collection:

Collect relevant transportation data to support your analysis. This may include:

* Timetables and schedules.
* GPS tracking data for vehicles.
* Passenger survey data.
* Maintenance records.
* Historical performance data.

Design:

Data Collection: Gather data from various sources, such as ticketing systems, GPS trackers on vehicles, and passenger surveys.Data Integration: Integrate and store the data in a suitable data warehouse, such as IBM Db2, or use a data lake for flexibility.

Data Transformation: Clean, preprocess, and transform the data into a format suitable for analysis.Data Modeling: Create a data model that reflects the structure of the public transport data and the business questions you want to answer.

IBM Cognos Setup: Install and configure IBM Cognos Analytics in your environment.Data Connection: Establish a connection between Cognos and your data source. Cognos supports various data sources, including relational databases and flat files.

Report and Dashboard Design: Design reports and dashboards in Cognos to visualize the data. You can use the Cognos Report Studio or Cognos Dashboards to create interactive reports.

Data Analysis: Use Cognos to create various analyses, such as route performance, passenger trends, and revenue analysis.

Data Visualization: Utilize Cognos’ visualization capabilities to create charts, graphs, and maps to represent your data effectively.Scheduled

Reporting: Set up automated reporting and distribution of reports to key stakeholders.

Sample Code:

Below is a simple example of how can use IBM Cognos to create a basic report:

<report xmlns="http://developer.cognos.com/schemas/report/10.2.1/" useStyleVersion="10.2.1" expressionLocale="en-us">

<list name="List1">

<listColumns>

<column name="column1">

<heading>

<text>Route Name</text>

</heading>

<source>

<model>

<dataItemLabel>Data Item</dataItemLabel>

<query>

<refQuery name="query1"/>

</query>

</model>

</source>

</column>

<column name="column2">

<heading>

<text>Total Passengers</text>

</heading>

<source>

<model>

<dataItemLabel>Data Item 2</dataItemLabel>

<query>

<refQuery name="query1"/>

</query>

</model>

</source>

</column>

</listColumns>

<noDataHandler>

<contents>

<block>

<contents>

<textItem>

<dataSource>

<model/>

</dataSource>

<dataItemLabel>Data Item</dataItemLabel>

</textItem>

</contents>

</block>

</contents>

</noDataHandler>

<style>

<defaultStyles>

<defaultStyle refStyle="pg"/>

</defaultStyles>

</style>

</list>

<queries>

<query name="query1">

<source>

<model/>

</source>

<selection>

<dataItem name="Data Item">

<expression>[Transport].[Route].[Route Name]</expression>

</dataItem>

<dataItem name="Data Item 2">

<expression>total([Transport].[Passenger].[Passenger Count])</expression>

</dataItem>

</selection>

</query>

</queries>

</report>

Code for public transport analysis

%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 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"))

*# Any results you write to the current directory are saved as output.*

['Public Transport Boarding Summary by Route, Trip, Stop and Week of Year.doc', '20140711.CSV']

Program2

import plotly.plotly as py

import plotly.graph\_objs as go

from plotly import tools

from plotly.offline import download\_plotlyjs, init\_notebook\_mode, plot, iplot

from bubbly.bubbly import bubbleplot

init\_notebook\_mode(connected=True)

from bokeh.plotting import figure, save

from bokeh.io import output\_file, output\_notebook, show

from bokeh.models import ColumnDataSource, GMapOptions,HoverTool

from bokeh.plotting import gmap

import tensorflow as tf

from tensorflow.python.keras.models import Sequential

from tensorflow.python.keras.layers import Input, Dense, GRU,LSTM, Embedding

from tensorflow.python.keras.optimizers import RMSprop

from tensorflow.python.keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard, ReduceLROnPlateau

data.shape

data.head(2)

Out:

(10857234, 6)

Out:

|  | TripID | RouteID | StopID | StopName | WeekBeginning | NumberOfBoardings |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 23631 | 100 | 14156 | 181 Cross Rd | 2013-06-30 00:00:00 | 1 |
| 1 | 23631 | 100 | 14144 | 177 Cross Rd | 2013-06-30 00:00:00 | 1 |

route.head(2)

out\_geo.head(2)

Out:

|  | route\_id | agency\_id | route\_short\_name | route\_long\_name | route\_desc | route\_type | route\_url | route\_color | route\_text\_color | RouteGroup |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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:

|  | 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 | -34.966 |  |  |  |  |  |

data['WeekBeginning'].unique()

out:

array([datetime.date(2013, 6, 30), datetime.date(2013, 7, 7),

datetime.date(2013, 7, 14), datetime.date(2013, 7, 21),

datetime.date(2013, 7, 28), datetime.date(2013, 8, 4),

datetime.date(2013, 8, 11), datetime.date(2013, 8, 18),

datetime.date(2013, 8, 25), datetime.date(2013, 9, 1),

datetime.date(2013, 9, 8), datetime.date(2013, 9, 15),

datetime.date(2013, 9, 22), datetime.date(2013, 9, 29),

datetime.date(2013, 10, 6), datetime.date(2013, 10, 13),

datetime.date(2013, 10, 20), datetime.date(2013, 10, 27),

datetime.date(2013, 11, 3), datetime.date(2013, 11, 10),

datetime.date(2013, 11, 17), datetime.date(2013, 11, 24),

datetime.date(2013, 12, 1), datetime.date(2013, 12, 8),

datetime.date(2013, 12, 15), datetime.date(2013, 12, 22),

datetime.date(2013, 12, 29), datetime.date(2014, 1, 5),

datetime.date(2014, 1, 12), datetime.date(2014, 1, 19),

datetime.date(2014, 1, 26), datetime.date(2014, 2, 2),

datetime.date(2014, 2, 9), datetime.date(2014, 2, 16),

datetime.date(2014, 2, 23), datetime.date(2014, 3, 2),

datetime.date(2014, 3, 9), datetime.date(2014, 3, 16),

datetime.date(2014, 3, 23), datetime.date(2014, 3, 30),

datetime.date(2014, 4, 6), datetime.date(2014, 4, 13),

datetime.date(2014, 4, 20), datetime.date(2014, 4, 27),

datetime.date(2014, 5, 4), datetime.date(2014, 5, 11),

datetime.date(2014, 5, 18), datetime.date(2014, 5, 25),

datetime.date(2014, 6, 1), datetime.date(2014, 6, 8),

datetime.date(2014, 6, 15), datetime.date(2014, 6, 22),

datetime.date(2014, 6, 29), datetime.date(2014, 7, 6)],

dtype=object)

## **Data Visualization**

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

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['type'].value\_counts().head(5).plot.bar(ax=axrr[2][0])

data['type'].value\_counts().tail(10).plot.bar(ax=axrr[2][1])

Out:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f1726f9e860>

Out:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f1615adbb38>

Out:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f1645050f28>

Out:

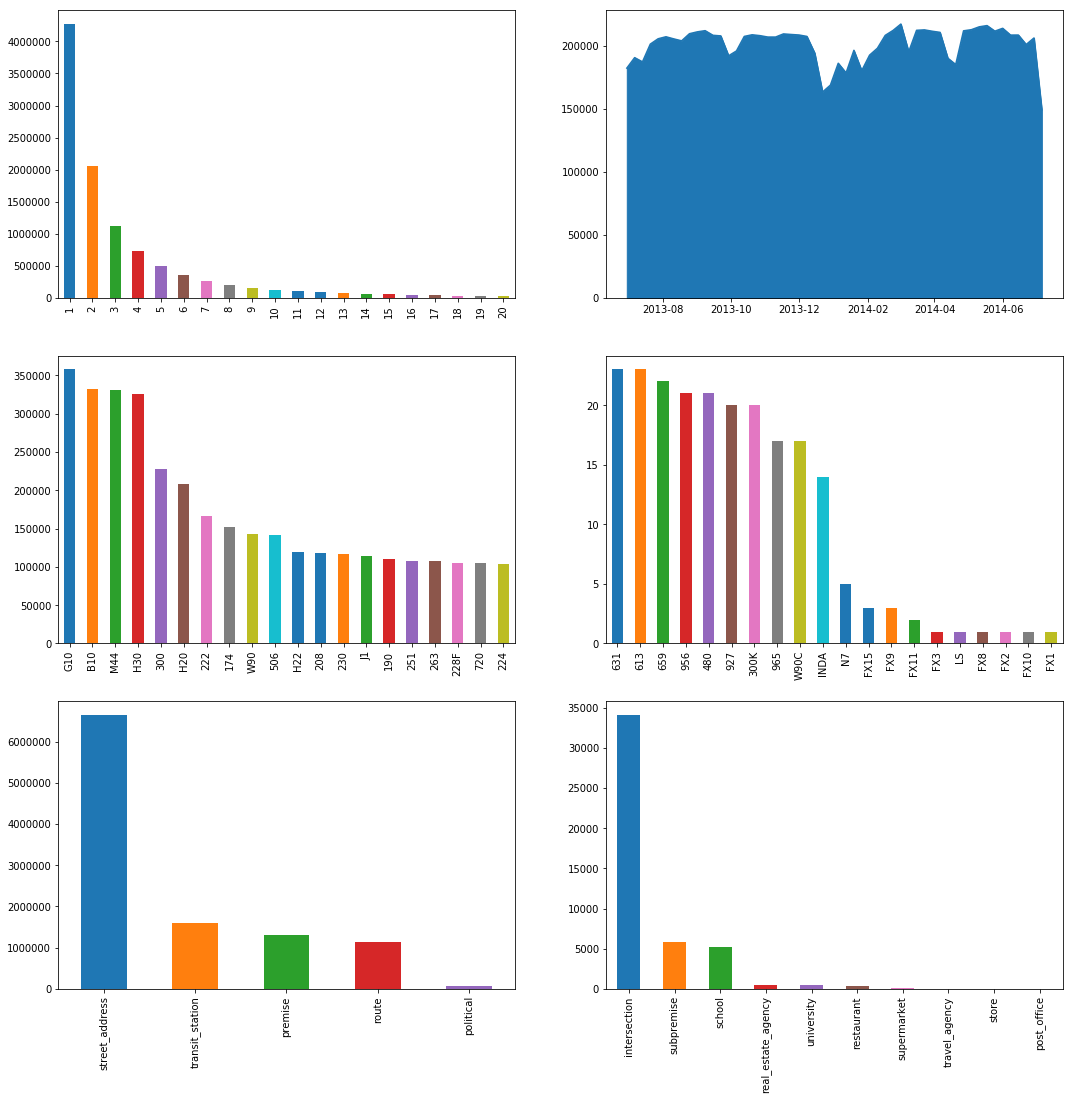
<matplotlib.axes.\_subplots.AxesSubplot at 0x7f171ef36588>

Out:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f171ef5dc50>

Out:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f171ef0d2e8>



**External Features :**

Some Important external data fields calculation

* **Is Holiday** Number of public holidays within that week
* **Distance From Centre** Distance measure from the city centre

For Calculating Distance between centre with other bus stops by using Longitude and Latitude we have used the Haversine formula

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 / 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: calc\_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()

In[12]: 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)

Adding the details regarding the Public holidays from June 2013 to June 2014

In[13]: 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[14]: data['WeekBeginning'] = pd.to\_datetime(data['WeekBeginning']).dt.date

In[15]: data['holiday\_label'] = data['WeekBeginning'].apply (lambda row: holid ay\_label(row))

Data Visualization :

### **Plot using Plotly**

In [38]:

linkcode

In[16]: *## for finding highest number of Boarding Bus stops*

bb\_grp = bb.groupby(['StopName']).agg({'NumberOfBoardings\_sum': ['sum']}). reset\_i ndex()['NumberOfBoardings\_sum'].sort\_values('sum')

bb\_grp[1000:1005]

bb.groupby(['StopName']).agg({'NumberOfBoardings\_sum': ['sum']}).reset\_in dex().il oc[[2325,1528,546,1043,1905]]

*# bb\_grp.iloc[[3054]]*

In[17] :

source\_1 = bb[bb['StopName'] == 'X2 King William St'].reset\_index(drop = True)

source\_2 = bb[bb['StopName'] == 'E1 Currie St'].reset\_index(drop = True)

source\_3 = bb[bb['StopName'] == 'I2 North Tce'].reset\_index(drop = True)

source\_4 = bb[bb['StopName'] == 'F2 Grenfell St'].reset\_index(drop = True)

source\_5 = bb[bb['StopName'] == 'D1 King William St'].reset\_index(drop = True)

In[16]: trace0 = go.Scatter(

x = source\_1['WeekBeginning'],

y = source\_1['NumberOfBoardings\_sum'],mode = 'lines+markers',name = 'X2 King William St')

trace1 = go.Scatter(

x = source\_2['WeekBeginning'],

y = source\_2['NumberOfBoardings\_sum'],mode = 'lines+markers',name = 'E1 Currie St')

trace2 = go.Scatter(

x = source\_3['WeekBeginning'],

y = source\_3['NumberOfBoardings\_sum'],mode = 'lines+markers',name = 'I2 North Tce')

trace3 = go.Scatter(

x = source\_4['WeekBeginning'],

y = source\_4['NumberOfBoardings\_sum'],mode = 'lines+markers',name = 'F2 Grenfell St')

trace4 = go.Scatter(

x = source\_5['WeekBeginning'],

y = source\_5['NumberOfBoardings\_sum'],mode = 'lines+markers',name = 'D1 King William St')

data = [trace0,trace1,trace2,trace3,trace4]

layout = dict(title = 'Weekly Boarding Total',

xaxis = dict(title = 'Week Number'),

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

shapes = [{*# Holidays Record: 2013-09-01*

'type': 'line','x0': '2013-09-01','y0': 0,'x1': '2013-09-02','y1': 18000,'line': {

'color': 'rgb(55, 128, 191)','width': 1,'dash': 'dashdot'},},

{*# 2013-10-07*

'type': 'line','x0': '2013-10-07','y0': 0,'x1': '2013-10-07','y1': 18000,'line': {

'color': 'rgb(55, 128, 191)','width': 1,'dash': 'dashdot'},},

{*# 2013-12-25*

'type': 'line','x0': '2013-12-25','y0': 0,'x1': '2013-12-26','y1': 18000,'line': {

'color': 'rgb(55, 128, 191)','width': 3,'dash': 'dashdot'},},

{*# 2014-01-27*

'type': 'line','x0': '2014-01-27','y0': 0,'x1': '2014-01-28','y1': 18000,'line': {

'color': 'rgb(55, 128, 191)','width': 1,'dash': 'dashdot'},},

{*# 2014-03-10*

'type': 'line','x0': '2014-03-10','y0': 0,'x1': '2014-03-11','y1': 18000,'line': {

'color': 'rgb(55, 128, 191)','width': 1,'dash': 'dashdot'},},

{*# 2014-04-18*

'type': 'line','x0': '2014-04-18','y0': 0,'x1': '2014-04-19','y1': 18000,'line': {

'color': 'rgb(55, 128, 191)','width': 3,'dash': 'dashdot'},},

{*# 2014-06-09*

'type': 'line','x0': '2014-06-09','y0': 0,'x1': '2014-06-10','y1': 18000,'line': {

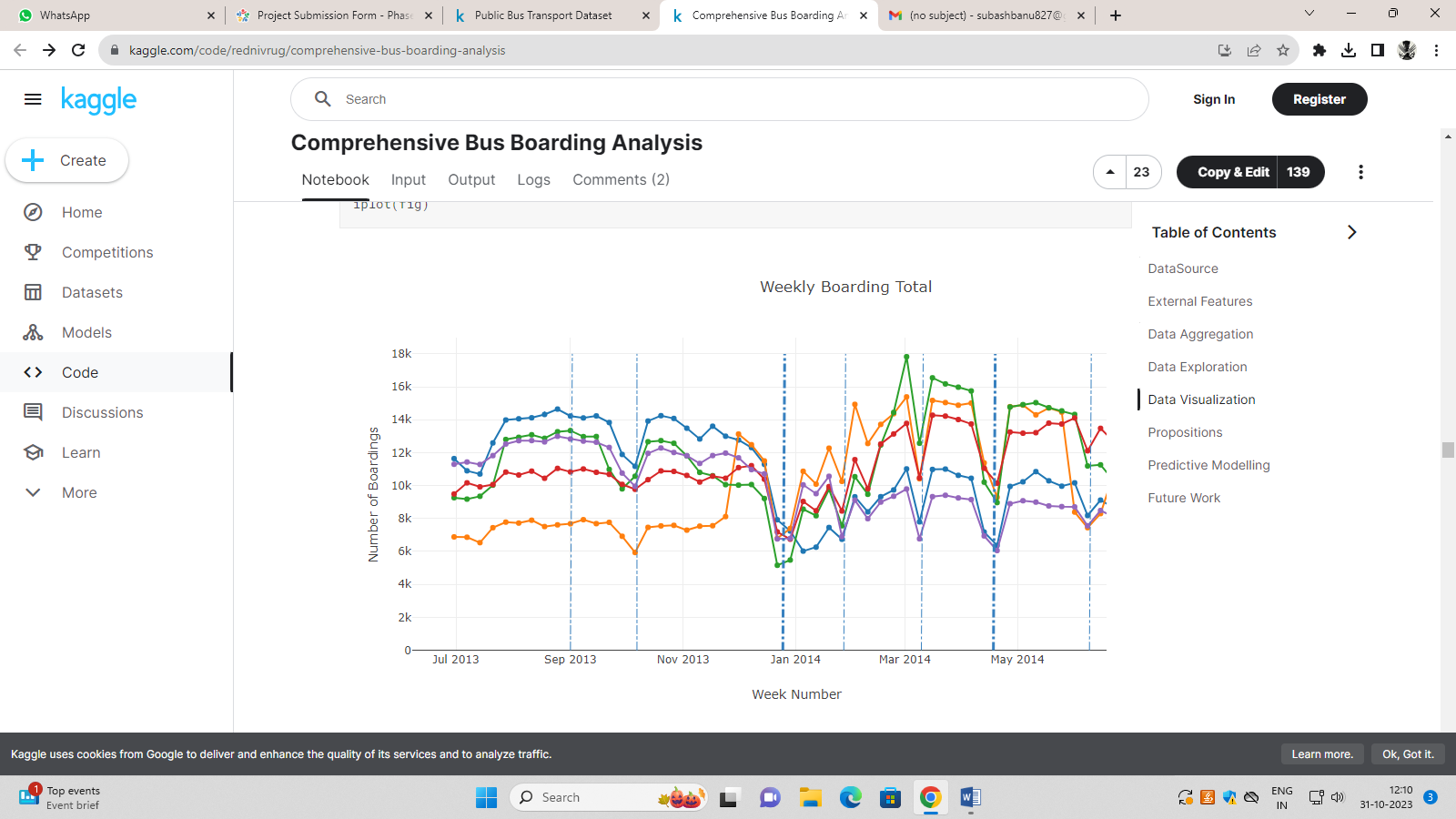
'color': 'rgb(55, 128, 191)','width': 1,'dash': 'dashdot'},},])

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

iplot(fig)

Jul 2013Sep 2013Nov 2013Jan 2014Mar 2014May 2014Jul 201402k4k6k8k10k12k14k16k18kExport to plot.ly »

X2 King William StE1 Currie StI2 North TceF2 Grenfell StD1 King William StWeekly Boarding TotalWeek NumberNumber of Boardings



**Inferences**:

* X2 King William St and stop near to that are the most busiest stops in the city. which having number of boardings per week more than 10k.
* Vertical lines are the indicator of holidays which came within that week.
* Whenever there is any Public holiday that week period have less than average number of people travelled from bus.

In [41]: source\_6 = bb[bb['StopName'] == '57A Hancock Rd'].reset\_index(drop = True)

source\_7 = bb[bb['StopName'] == '37 Muriel Dr'].reset\_index(drop = True)

source\_8 = bb[bb['StopName'] == '18B Springbank Rd'].reset\_index(drop = True)

source\_9 = bb[bb['StopName'] == '27E Sir Ross Smith Av'].reset\_index(drop = True)

source\_10 = bb[bb['StopName'] == '46A Baldock Rd'].reset\_index(drop = True)

## **Propositions**

linkcode

Rate of change in the traffic pattern in all different bus stops.

In [48]: d=[]

for i **in** bb['StopName'].unique():

d.append({'StopName': i,'Boarding\_sum':np.sum(bb[bb['StopName'] == i]['NumberOfBoardings\_sum'].pct\_change())/54,

'Boarding\_count':np.sum(bb[bb['StopName'] == i]['NumberOfBoardings\_count'].pct\_change())/54,

'Boarding\_max':np.sum(bb[bb['StopName'] == i]['NumberOfBoardings\_max'].pct\_change())/54})

pct\_chng = pd.DataFrame(d)

*#pct\_chng.head()*

pct\_chng['Boarding\_sum'].nlargest(5)

pct\_chng['Boarding\_sum'].nsmallest(5)

pct\_chng[pct\_chng['Boarding\_sum']<0].shape

pct\_chng.iloc[[3110,2134,214,1538,1290]]

**Inferences**:

* These 5 stops W Grote St,52 Taylors Rd,13 Tutt Av,37A Longwood Rd,32A Frederick Rd having the largest percent increase.
* There are 27 Bus stops where number of boardings have decreased.
* The number of busses can be found by taking the number of boarding divided by bus capacity which can take as 50.

## **Predictive Modelling**[**¶**](https://www.kaggle.com/code/rednivrug/comprehensive-bus-boarding-analysis#Predictive-Modelling-)

Get info like RouteID,latitude,longitude,postcode,dist\_from\_centre & holiday\_label 6 features from the main dataset

bb1 = pd.merge(bb, out\_geo, how='left', left\_on = 'StopName', right\_on = 'input\_string')

bb1['holiday\_label'] = bb1['WeekBeginning'].apply (lambda row: holiday\_label(row))

*##Final 11 features have been used for the forecastng.*

cols = ['StopName','WeekBeginning','type\_x','NumberOfBoardings\_sum','NumberOfBoardings\_count','NumberOfBoardings\_max','latitude','longitude','postcode','dist\_from\_centre','holiday\_label']

bb1=bb1[cols]

bb1.shape

bb1.head()

Modelling using regression models.

1. lightGbm Regressor
2. Gru

### **Using LightGbm**

from sklearn.ensemble import RandomForestRegressor

*# model = lgb.LGBMRegressor()*

model = RandomForestRegressor(n\_estimators=700, min\_samples\_leaf=3, max\_features=0.5,n\_jobs=-1)

*# model = lgb.LGBMRegressor(max\_depth=10,learning\_rate=0.0227,n\_estimators=195,num\_leaves=11,reg\_alpha=1.5764,reg\_lambda=0.0478,subsample=0.7776,colsample\_bytree=0.7761)*

model.fit(train\_x.values,train\_sum\_y['NumberOfBoardings\_sum'].values)

preds = model.predict(test\_x.values)

Out[60]:

RandomForestRegressor(bootstrap=True, criterion='mse', max\_depth=None,

max\_features=0.5, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=3, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=700, n\_jobs=-1,

oob\_score=False, random\_state=None, verbose=0, warm\_start=False)

plt.figure(figsize=(15,5))

plt.plot(test\_sum\_y['NumberOfBoardings\_sum'].values, label='true')

plt.plot(preds, label='pred')

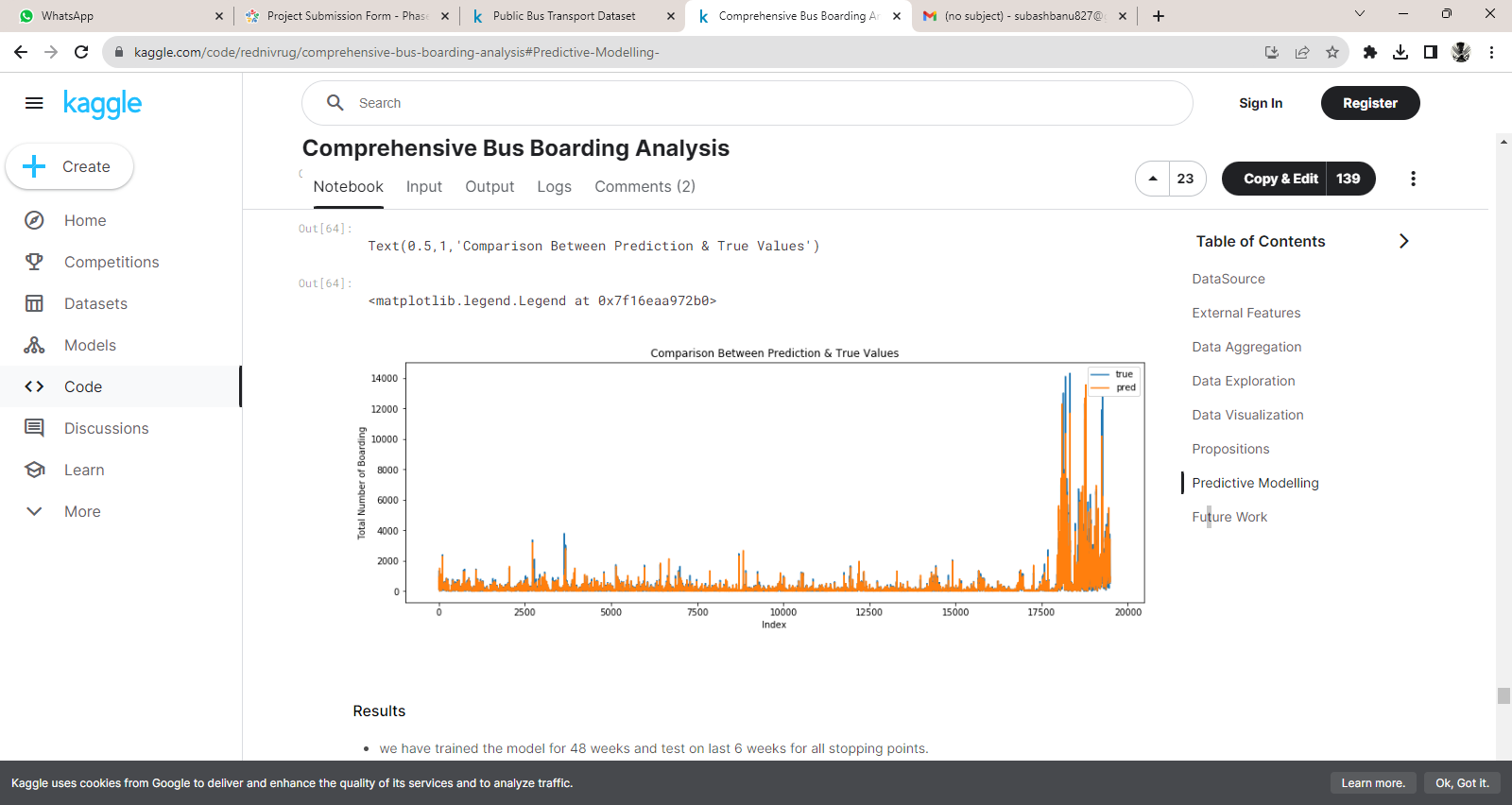
plt.ylabel("Total Number of Boarding")

plt.xlabel("Index")

plt.title("Comparison Between Prediction & True Values")

plt.legend()

plt.show()



### **Results**

* we have trained the model for 48 weeks and test on last 6 weeks for all stopping points.
* High Rmse value came because we didn't scale the values.so we got the actual prediction instead of scaled prediction

### **Using Gru**

target\_names = ['NumberOfBoardings\_sum', 'NumberOfBoardings\_count', 'NumberOfBoardings\_max']

train\_col = ['StopName','WeekBeginning','type\_x','latitude','longitude','postcode','dist\_from\_centre','holiday\_label']

*##want to predict 1 day in future.*

shift\_days = 6

shift\_steps = shift\_days \* 3249

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)

*#loss function define.*

warmup\_steps = 0

def loss\_mse\_warmup(y\_true, y\_pred):

*# [batch\_size, sequence\_length, num\_y\_signals].*

y\_true\_slice = y\_true[:, warmup\_steps:, :]

y\_pred\_slice = y\_pred[:, warmup\_steps:, :]

*# Calculate the MSE loss for each value in these tensors.*

loss = tf.losses.mean\_squared\_error(labels=y\_true\_slice,predictions=y\_pred\_slice)

loss\_mean = tf.reduce\_mean(loss)

return loss\_mean

*#early stopping and learning rate decrease callbacks*

callback\_early\_stopping = EarlyStopping(monitor='val\_loss',patience=5, verbose=1)

callbacks = [callback\_early\_stopping]

*# model.load\_weights(path\_checkpoint)*

*# result = model.evaluate(x=np.expand\_dims(x\_test\_scaled, axis=0),*

*# y=np.expand\_dims(y\_test\_scaled, axis=0))*

*# print("loss (test-set):", result)*