Public Transportation Analysis

2023 Naan Mudhalvan - IBM Data Analytics with Cognos
Group 1 - Project 8
College : NM001 - College of Engineering Guindy
Proj_200340_Team_2

Members: Abinithi R, Abirami S V, Adithya R U, Akshaya G R,
Sai Rishi A N
Faculty Mentor : Dr. G Geetha

PHASE 4

DEVELOPMENT PART 2

PROBLEM DEFINITION :

Analyse public transportation data to assess **service efficiency**, **on time performance**, and **passenger feedback.**
Provide insights that **support transportation improvement initiatives** and enhance the overall public transportation experience.

DATA ANALYSIS

Some Important external data fields calculation

- **IsHoliday** Number of public holidays within that week
- **DistanceFromCentre** 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

```
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)


# 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()


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'


data['WeekBeginning'] = pd.to_datetime(data['WeekBeginning']).dt.date
data['holiday_label'] = data['WeekBeginning'].apply (lambda row: holiday_label(row))

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

```
data = pd.merge(data, route, how='left', left_on = 'RouteID', right_on = 'route_id')

col = ['TripID','RouteID','StopID','StopName', 'WeekBeginning','NumberOfBoardings','
formatted_address','latitude','longitude','postcode','type','route_desc','dist_from_
centre','holiday_label']

data = data[col]

#saving the final dataset
data.to_csv('Weekly_Boarding.csv',index=False)
```

Aggregate the Data According to Weeks and Stop names

- **NumberOfBoardings_sum** Number of Boardings within particular week for each Bus stop
- **NumberOfBoardings_count** Number of times data is recorded within week
- **NumberOfBoardings_max** Maximum number of boarding done at single time within week

```
grouped = data.groupby(['StopName','WeekBeginning','type']).agg({'NumberOfBoardings'
          : ['sum', 'count','max']})
grouped.columns = ["_".join(x) for x in grouped.columns.ravel()]

st_week_grp = pd.DataFrame(grouped).reset_index()
st_week_grp.shape
st_week_grp.head()


#Gathering only the Stop Name which having all 54 weeks of Data
st_week_grp1 = pd.DataFrame(st_week_grp.groupby ('StopName') ['WeekBeginning']
.count()).reset_index()

aa=list(st_week_grp1[st_week_grp1['WeekBeginning'] == 54]['StopName'])
bb = st_week_grp[st_week_grp['StopName'].isin(aa)]

# save the aggregate data
bb.to_csv('st_week_grp.csv', index=False)
```

DATA EXPLORATION

Total Having 1 Year of Data from date 2013-06-30 till 2014-07-06 in a Weekly interval based.

Having Total of 4165 Stops in South Australian Metropolitan Area.

```
data.nunique()

data.shape
data.columns

data.head(3)
data.isnull().sum()

data['WeekBeginning'].unique()
```

VISUALIZATION

```
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])
```

Inferences:

- More than 40 lakhs times only single person board from the bus stop.
- There are average of 1.8 lakhs people travel every week by bus in adelaide metropolitan area.
- G10,B10,M44,H30 are the most busiest routes in the city while FX8,FX3,FX10,FX1,FX2 are the least.
- Most of the Bus stops are Street_Address Type while there are very few which are store or post office.

```
data['postcode'].value_counts().head(20).plot.bar()
bb_grp = data.groupby(['dist_from_centre']).agg({'NumberOfBoardings': ['sum']}).rese
t_index()

bb_grp.columns = bb_grp.columns.get_level_values(0)
bb_grp.head()
bb_grp.columns

trace0 = go.Scatter(
    x = bb_grp['dist_from_centre'],
    y = bb_grp['NumberOfBoardings'],mode = 'lines+markers',name = 'X2 King William S
t')

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)
```

Inferences:

- As we move away from centre the number of Boarding decreases
- There are cluster of bus stops near to the main Adelaide city as oppose to outside.so that's why most of boardings are near to center.

PLOT USING PLOTLY

```
# for finding highest number of Boarding Bus stops
bb_grp = bb.groupby(['StopName']).agg({'NumberOfBoardings_sum': ['sum']}).reset_inde
x()['NumberOfBoardings_sum'].sort_values('sum')
bb_grp[1000:1005]
bb.groupby(['StopName']).agg({'NumberOfBoardings_sum': ['sum']}).reset_index().iloc[
[2325,1528,546,1043,1905]]
```

```
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)


trace0 = go.Scatter(
    x = source_1['WeekBeginning'],
    y = source_1['NumberOfBoardings_sum'],mode = 'lines+markers',name = 'X2 King Wil
liam St')
trace1 = go.Scatter(
    x = source_2['WeekBeginning'],
    y = source_2['NumberOfBoardings_sum'],mode = 'lines+markers',name = 'E1 Currie S
t')
trace2 = go.Scatter(
    x = source_3['WeekBeginning'],
    y = source_3['NumberOfBoardings_sum'],mode = 'lines+markers',name = 'I2 North Tc
e')
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 Wil
liam 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)
```

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.

```
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)

trace0 = go.Scatter(
    x = source_6['WeekBeginning'],
    y = source_6['NumberOfBoardings_sum'],mode = 'lines+markers',name = '57A Hancock
Rd')
trace1 = go.Scatter(
    x = source_7['WeekBeginning'],
    y = source_7['NumberOfBoardings_sum'],mode = 'lines+markers',name = '37 Muriel D
r')
trace2 = go.Scatter(
    x = source_8['WeekBeginning'],
    y = source_8['NumberOfBoardings_sum'],mode = 'lines+markers',name = '18B Springb
ank Rd')
trace3 = go.Scatter(
    x = source_9['WeekBeginning'],
    y = source_9['NumberOfBoardings_sum'],mode = 'lines+markers',name = '27E Sir Ros
s Smith Av')
trace4 = go.Scatter(
    x = source_10['WeekBeginning'],
y = source_10['NumberOfBoardings_sum'],mode = 'lines+markers',name = '46A Baldock Rd
')

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': 80,'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': 80,'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': 80,'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': 80,'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': 80,'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': 80,'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': 80,'line': {
        'color': 'rgb(55, 128, 191)','width': 1,'dash': 'dashdot'},},])
fig = dict(data=data, layout=layout)
iplot(fig)
```

Inferences:

- Same decreasing affect of Holidays on number of people travelling through bus can be seen in other city bus stops also.
- The width of vertical blue line shows the number of holidays come within that week period.
- Two thickest blue lines shows Christmas and New year period while other one was easter & Good friday period.on both the occassion number of public holidays within week period was 3.

PLOT USING BUBBLY

```
bb1=bb.copy()

# Label encode the Date type for easy Plotting
le = LabelEncoder()
bb1['WeekBeginning'] = le.fit_transform(bb1['WeekBeginning'])

figure = bubbleplot(dataset=bb1, x_column='NumberOfBoardings_sum', y_column='NumberO
fBoardings_count',
    bubble_column='StopName', time_column='WeekBeginning', size_column='NumberOfBoar
dings_max',
    color_column='type',
    x_title="Total Boardings", y_title="Frequency Of Boardings",show_slider=True,
    title='Adelaide Weekly Bus Transport Summary 2D',x_logscale=True, scale_bubble=2
,height=650)

iplot(figure, config={'scrollzoom': True})
```

In the graph above, the size corresponds to the maximum number of people board at single time and the Total boardings and Frequency of boardings with stop name can be seen by hovering over the cursor on the bubbles.

The animated bubble charts convey a great deal of information since they can accomodate upto seven variables in total, namely:

- X-axis (Total Boardings per week)
- Y-axis (Frequency of Bus Boarding)
- Bubbles (Bus stop name)
- Time (in week period)
- Size of bubbles (maximum number of people board at single time)
- Color of bubbles (Type of Bus stop)

PROPOSITIONS

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

```
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['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

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()

#Replace all Nan by Mode
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_numeric)
```

Label Encode the Categorical data

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

Split into Train Test for Modelling further

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

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

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']]
```

MODELLING USING REGRESSION MODELS

USING LightGbm

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=700, min_samples_leaf=3, max_features=0.5
,n_jobs=-1)
model.fit(train_x.values,train_sum_y['NumberOfBoardings_sum'].values)
preds = model.predict(test_x.values)

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

```
test_sum_y.values[:15]
preds[:15]

fig, ax = plt.subplots(figsize=(6,10))
lgb.plot_importance(model, max_num_features=50, height=0.8, ax=ax)
ax.grid(False)
plt.title("LightGBM - Feature Importance", fontsize=15)
plt.show()
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