Analysis, Model Building, and Evaluation with IBM Cognos

Introduction:

In this technology project, we embarked on a journey of data analysis, model building, and evaluation using the powerful tools provided by IBM Cognos. Our goal was to perform various analyses and create data visualizations, ultimately culminating in the creation of a comprehensive document for assessment.

1) Data Collection and Preparation

Data serves as the foundation for any data-driven project, and this section delves into the critical steps involved in collecting and preparing data for our project.

Data Sourcing:

The first step in this phase was data sourcing. We specify the data sources used, whether they were internal databases, external APIs, or third-party datasets. We discuss the importance of data relevance and reliability in the context of our project's objectives.

Data Cleaning and Transformation:

Cleaning and transformation were pivotal for data quality. We outline the steps taken to address issues such as missing values, duplicates, and inconsistencies. Additionally, we describe how data was transformed to a common format, ensuring compatibility across different sources.

Data Integration:

Data integration is essential when dealing with disparate data sources. We elaborate on the methods and tools used to combine data from various origins. This process aimed to create a unified dataset, enabling comprehensive analysis.

Data Standardization and Normalization:

Standardization and normalization are integral for ensuring that data is in a consistent format. We delve into the specifics of these processes and how they contributed to the dataset’s uniformity.

2) Exploratory Data Analysis (EDA)

Exploratory Data Analysis is a fundamental phase in our project, where we dive into the dataset to uncover insights, patterns, and anomalies that will inform subsequent steps.

Descriptive Statistics and Summary:

In this subtopic, we discuss how descriptive statistics and summary metrics were utilized to gain an initial understanding of the dataset. We highlight the key statistics and measures used, including mean, median, standard deviation, and quartiles, and their relevance in summarizing data characteristics.

Data Visualization:

Visualization is a powerful tool for EDA. We elaborate on the types of visualizations employed, including histograms, scatter plots, and box plots. These visuals helped us interpret data distributions, relationships, and potential outliers. We discuss the insights derived from these visual representations.

Data Exploration:

This subtopic delves into the process of data exploration. We describe how we identified trends, patterns, and anomalies by drilling deeper into the data. It covers techniques such as grouping, sorting, and filtering, as well as the use of IBM Cognos for interactive data exploration.

Data Patterns and Anomalies:

We identify and describe data patterns observed during EDA. This could include trends, seasonality, or other recurring features within the data. Simultaneously, we address any anomalies or irregularities identified during the exploration, elucidating their potential implications.

Impact on Project Objectives:

We underscore the significance of the insights gained through EDA in relation to our project objectives. By identifying data patterns and anomalies, we discuss how we were able to adapt our project strategy and model building approach to better align with the project's goals.

3) Model Building

Model building is the core of our project, where we leverage advanced techniques to construct predictive models that address the project's objectives.

Algorithm Selection:

The choice of algorithms is a critical decision in model building. In this subtopic, we outline the selection process, explaining how we determined the most suitable algorithms for our project's objectives. We consider the trade-offs between accuracy, interpretability, and model complexity.

Feature Engineering:

Feature engineering is the art of creating informative features that enhance model performance. We discuss how we crafted relevant features from the raw data, including techniques like one-hot encoding, feature scaling, and the creation of interaction terms. These engineered features were instrumental in capturing patterns within the data.

Model Training and Validation:

This subtopic delves into the process of training and validating the models. We describe how we split the data into training and validation sets, as well as the use of techniques such as cross-validation to ensure model robustness. We touch on hyperparameter tuning and model evaluation as integral components of this phase.

Ensemble Models and Stacking:

Ensemble models, such as random forests and gradient boosting, were explored as part of our model building approach. We detail how we leveraged these techniques to combine multiple models for improved prediction accuracy. Additionally, we discuss the concept of model stacking and its application in our project.

Model Interpretability:

Model interpretability is a crucial consideration, especially when dealing with complex algorithms. We describe the methods and tools used to make our models interpretable, including feature importance analysis and partial dependence plots. This allowed us to not only make predictions but also understand the factors driving those predictions.

4) Model Evaluation

Model evaluation is a critical phase in our project, where we rigorously assess the performance and reliability of the predictive models.

Performance Metrics:

Performance metrics are central to evaluating model effectiveness. We discuss the selection and application of appropriate metrics, including accuracy, precision, recall, F1score, and ROC AUC. This subtopic highlights the importance of choosing metrics aligned with the project's objectives.

Cross-Validation:

Cross-validation is a vital technique for robust model assessment. We detail how we applied k-fold cross-validation to ensure that our models' performance estimates were reliable and not overfit to the training data.

Model Comparison:

Comparing different models is an essential step in model evaluation. We describe how we compared the performance of various algorithms and ensemble models. The comparison allowed us to select the most suitable models for the project.

Overfitting and Underfitting Analysis:

Overfitting and underfitting are common challenges in model building. In this subtopic, we elaborate on how we detected and mitigated these issues to ensure our models' generalizability.

Interpreting Model Results:

Interpreting model results is as crucial as model performance. We discuss how we examined and interpreted the results to extract actionable insights. This process involved analysing feature importance, confusion matrices, and learning curves.

5) Document Creation

Creating a comprehensive document is a pivotal step in our project, where we compile our findings, analysis, and model outcomes for effective communication and assessment.

Document Structure:

The structure of the document is vital for clarity and coherence. In this subtopic, we detail the document's organization, including sections for an executive summary, project objectives, data description, methodology, analysis results, model outcomes, and conclusions.

Data Insights and Visualizations:

This subtopic focuses on presenting data insights and visualizations in the document. We explain how we used descriptive statistics, graphs, and charts to convey the dataset's characteristics and trends. We also discuss the importance of clearly labeled and interpretable visualizations.

Model Methodologies:

Model methodologies and results are key components of the document. We elaborate on how we outlined the model building process, highlighting the algorithms, features, and techniques used. The document presents model performance results and their implications for the project objectives.

Interpreting Results for Stakeholders:

Effectively conveying results to stakeholders is vital for project success. We discuss how we translated complex model outcomes into understandable insights, providing actionable recommendations based on the analysis.

Visual Storytelling:

Visual storytelling enhances document engagement. We explain the use of storytelling techniques to convey data insights, model performance, and recommendations in a compelling and memorable manner.

Conclusion

Our project was a comprehensive exploration of data analysis, model building, and evaluation using IBM Cognos. Through the various project phases, we gained insights into the dataset and developed models that met the project's objectives. The document we've created serves as a testament to our efforts and the knowledge we've gained throughout this project.

Acknowledgments

We extend our gratitude to our project team and mentors for their support and guidance during this endeavour.

Submission

We are pleased to submit our project document, which encapsulates our journey through this technology project. We look forward to the assessment and feedback that will further enhance our skills and knowledge.

Public Transport Effectively Analysis

Analysis phase:

[ ]:

```

```

[1]:

```
%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

## For Multiple Output in single cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
warnings.filterwarnings('ignore')
```

[2]:

```
data = pd.read_csv('../input/unisys/ptsboardingsummary/20140711.CSV')
data.shape
data.head(10)
```

[2]: (10857234, 6)

[2]:

	TripID	RouteID	…	WeekBeginning	NumberOfBoardings
0	23631	100	…	2013-06-30 00:00:00	1
1	23631	100	…	2013-06-30 00:00:00	1
2	23632	100	…	2013-06-30 00:00:00	1
3	23633	100	…	2013-06-30 00:00:00	2
4	23633	100	…	2013-06-30 00:00:00	1
5	23634	100	…	2013-06-30 00:00:00	1
6	23634	100	…	2013-06-30 00:00:00	1
7	23634	100	…	2013-06-30 00:00:00	1
8	23634	100	…	2013-06-30 00:00:00	1

```
9   23634     100         …         2013-06-30 00:00:00                 1

[10 rows x 6 columns]
```

[3]:
```
out_geo = pd.read_csv('../input/outgeo/output_geo.csv')
out_geo.shape
out_geo.head()
```

[3]:
```
(4165, 10)
```

[3]:
```
           accuracy                        …
type
0           ROOFTOP                        …
street_address
1           ROOFTOP                        …
street_address
2           ROOFTOP                        …
street_address
3  GEOMETRIC_CENTER                        …
bus_station,establishment,point_of_interest,tr…
4           ROOFTOP                        …
street_address

[5 rows x 10 columns]
```

0.1 External Features

[4]:
```
#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
```

[5]:
```
out_geo['dist_from_centre'] = out_geo[['latitude','longitude']].apply(lambda x:␣
 ↪calc_dist(*x), axis=1)
```

[6]:
```
out_geo.head()
```

[6]:
```
             accuracy        …         dist_from_centre
0             ROOFTOP        …                 5.180961
1             ROOFTOP        …                 5.172525
2             ROOFTOP        …                 5.180709
3    GEOMETRIC_CENTER        …                 7.057549
4             ROOFTOP        …                 4.900099

[5 rows x 11 columns]
```

[7]:
```python
#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])
```

[8]:
```python
out_geo['type'].unique()
```

[8]:
```
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)
```

[9]:
```python
data['WeekBeginning'] = pd.to_datetime(data['WeekBeginning']).dt.date
data['WeekBeginning'][1]
```

[9]:
```
datetime.date(2013, 6, 30)
```

0.2 Data Aggregation

[10]:
```python
#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
```

[10]:
```
   TripID RouteID       …                  type dist_from_centre
0   23631     100       …        street_address         5.180961
1   23631     100       …        street_address         5.172525
2   23632     100       …        street_address         5.180709
3   23633     100       …       transit_station         7.057549
4   23633     100       …        street_address         4.900099

[5 rows x 17 columns]
```

[10]:
```
(10857234, 17)
```

[11]:
```
#Columns to keep for further analysis
col = ['TripID', 'RouteID', 'StopID', 'StopName',␣
↪'WeekBeginning','NumberOfBoardings',
      'latitude', 'longitude','postcode','type','dist_from_centre']
data = data[col]
```

[12]:
```
##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

[13]:
```
grouped = data.groupby(['StopName','WeekBeginning','type'])
#grouped.head()
```

[14]:
```
# 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()]
```

[15]:
```
grouped.head(10)
grouped.columns
```

[15]:
```
                                          NumberOfBoardings_sum          …
NumberOfBoardings_max
StopName     WeekBeginning type                                          …
1 Anzac Hwy 2013-06-30    street_address                   1003          …
51
             2013-07-07    street_address                    783          …
28
             2013-07-14    street_address                    843          …
45
             2013-07-21    street_address                    710          …
28
             2013-07-28    street_address                    898          …
41
             2013-08-04    street_address                    799          …
40
             2013-08-11    street_address                   1012          …
71
             2013-08-18    street_address                    793          …
41
             2013-08-25    street_address                    897          …
45
```

```
            2013-09-01    street_address                    1368          …
59

[10 rows x 3 columns]
```

[15]:
```
Index(['NumberOfBoardings_sum', 'NumberOfBoardings_count',
       'NumberOfBoardings_max'],
      dtype='object')
```

[16]:
```python
st_week_grp = pd.DataFrame(grouped).reset_index()
st_week_grp.shape
st_week_grp.head()
```

[16]:
```
(207864, 6)
```

[16]:
```
      StopName          …          NumberOfBoardings_max
0  1 Anzac Hwy          …                             51
1  1 Anzac Hwy          …                             28
2  1 Anzac Hwy          …                             45
3  1 Anzac Hwy          …                             28
4  1 Anzac Hwy          …                             41

[5 rows x 6 columns]
```

[17]:
```python
st_week_grp1 = pd.DataFrame(st_week_grp.groupby('StopName')["WeekBeginning"].
 ↪count()).reset_index()
st_week_grp1.head()
```

[17]:
```
         StopName  WeekBeginning
0     1 Anzac Hwy             54
1    1 Bartels Rd             54
2    1 Botanic Rd             54
3      1 Frome Rd             54
4  1 Fullarton Rd             54
```

[18]:
```python
#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]
```

[18]:
```
['1 Bartels Rd',
 '1 Botanic Rd',
 '1 Frome Rd',
 '1 Fullarton Rd',
 '1 George St',
 '1 Glen Osmond Rd',
 '1 Goodwood Rd',
 '1 Henley Beach Rd',
```

```
 '1 Kensington Rd']
```

[19]:
```
bb = st_week_grp[st_week_grp['StopName'].isin(aa)]
bb.head()
bb.shape

type(bb)
```

[19]:
```
       StopName          …          NumberOfBoardings_max
0   1 Anzac Hwy          …                             51
1   1 Anzac Hwy          …                             28
2   1 Anzac Hwy          …                             45
3   1 Anzac Hwy          …                             28
4   1 Anzac Hwy          …                             41

[5 rows x 6 columns]
```

[19]:
```
(175446, 6)
```

[19]:
```
pandas.core.frame.DataFrame
```

[20]:
```
#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)
```

[20]:
```
(10567931, 11)
```

```
data without stopage removing:  (10857234, 11)
data, after removing stoppage not having the data of whole 54 weeks:  (10567931,
11)
```

[21]:
```
new_data.head(2)
filtered_data = new_data[new_data['dist_from_centre'] <= 100]
filtered_data.shape
```

[21]:
```
   TripID RouteID       …                     type dist_from_centre
0   23631     100       …           street_address         5.180961
1   23631     100       …           street_address         5.172525

[2 rows x 11 columns]
```

[21]:
```
(10341468, 11)
```

[22]:
```
data = filtered_data.copy()
data.shape
```

[22]: (10341468, 11)

[23]:
```
#No of boarding for each stopage in all weeks
#bb["StopName"].groupby(NumberOfBoardings_sum)
stopageName_with_boarding = bb.groupby(['StopName']).
    ↪agg({'NumberOfBoardings_sum': ['sum']})

#stopageName_with_boarding.columns = ["_".join(x) for x in␣
    ↪stopageName_with_boarding.columns.ravel()]
#stopageName_with_boarding.head()
stopageName_with_boarding = pd.DataFrame(stopageName_with_boarding.
    ↪reset_index())
```

[24]:
```
#type(stopageName_with_boarding)
stopageName_with_boarding.columns = ["StopName",␣
    ↪"Total_boarding_on_the_stopage"]
#stopageName_with_boarding.shape
stopageName_with_boarding.head()
```

[24]:

	StopName	Total_boarding_on_the_stopage
0	1 Anzac Hwy	39429
1	1 Bartels Rd	8412
2	1 Botanic Rd	14868
3	1 Frome Rd	67458
4	1 Fullarton Rd	585

[25]:
```
## save the aggregate data
#bb.to_csv('st_week_grp.csv', index=False)
```

0.3 Data Exploration

[26]:
```
data.nunique()
#data.isnull().sum()
#data['WeekBeginning'].unique()
```

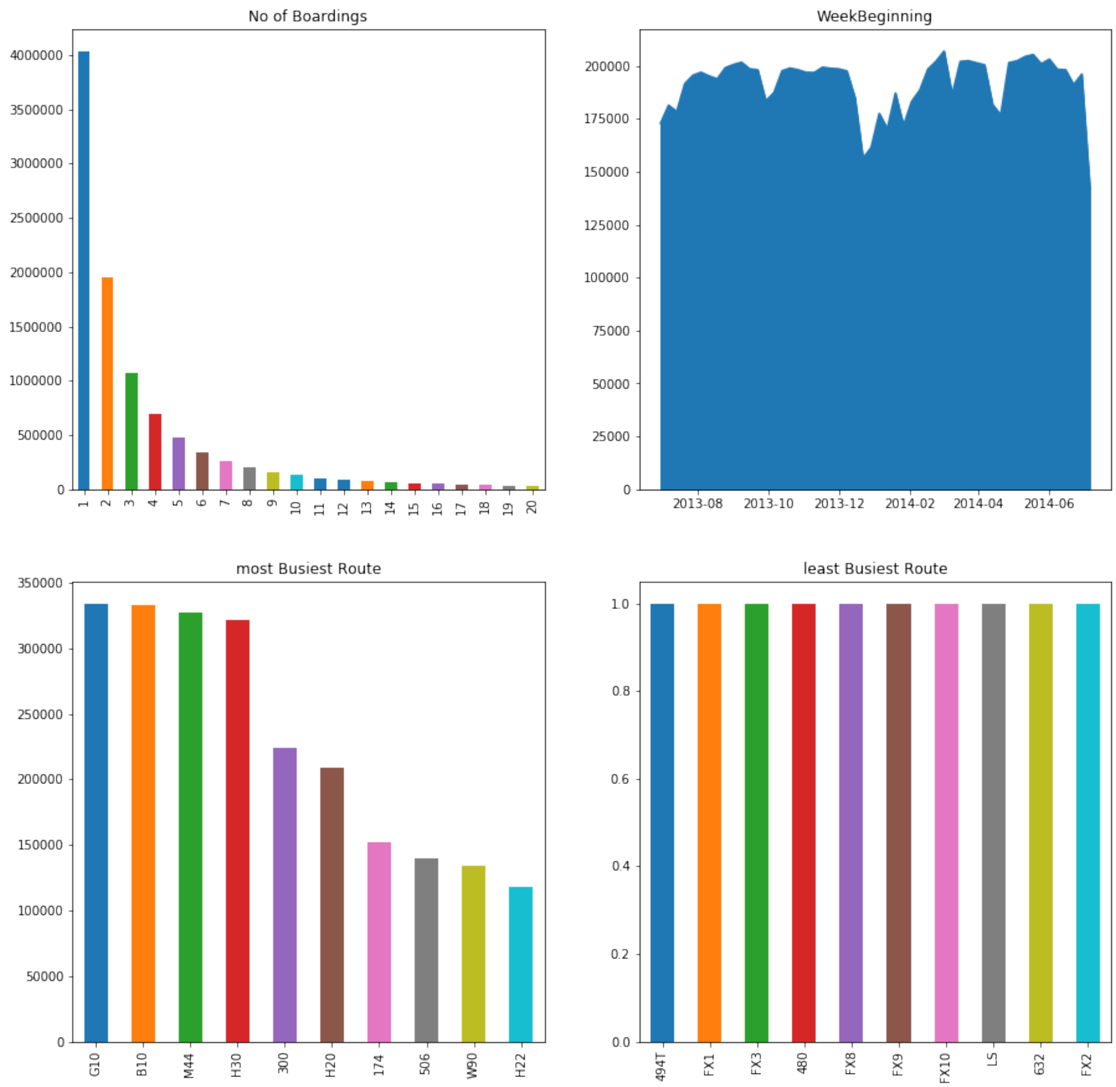
[26]:
```
TripID               39211
RouteID                616
StopID                5838
StopName              3127
WeekBeginning           54
NumberOfBoardings      359
latitude              2393
longitude             2379
```

```
postcode              138
type                    8
dist_from_centre     2397
dtype: int64
```

0.4 Data Visualization

```
[27]: ##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])
```

```
[27]: Text(0.5,1,'No of Boardings')

[27]: <matplotlib.axes._subplots.AxesSubplot at 0x7c5bd6604ba8>

[27]: Text(0.5,1,'WeekBeginning')

[27]: <matplotlib.axes._subplots.AxesSubplot at 0x7c5a5f56ab00>

[27]: Text(0.5,1,'most Busiest Route')

[27]: <matplotlib.axes._subplots.AxesSubplot at 0x7c5a5f548dd8>

[27]: Text(0.5,1,'least Busiest Route')

[27]: <matplotlib.axes._subplots.AxesSubplot at 0x7c5a8c6a1048>
```




```
[28]: 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)
```

[28]:

	StopName	Total_boarding_on_the_stopage
3054	I2 North Tce	628859
3125	X1 King William St	622099
3032	F2 Grenfell St	604149
3130	X2 King William St	583227
3021	E1 Currie St	550396
3207	Zone C Paradise Interchange	547709
3015	D1 King William St	541046
3211	Zone C Tea Tree Plaza Intercha	451960

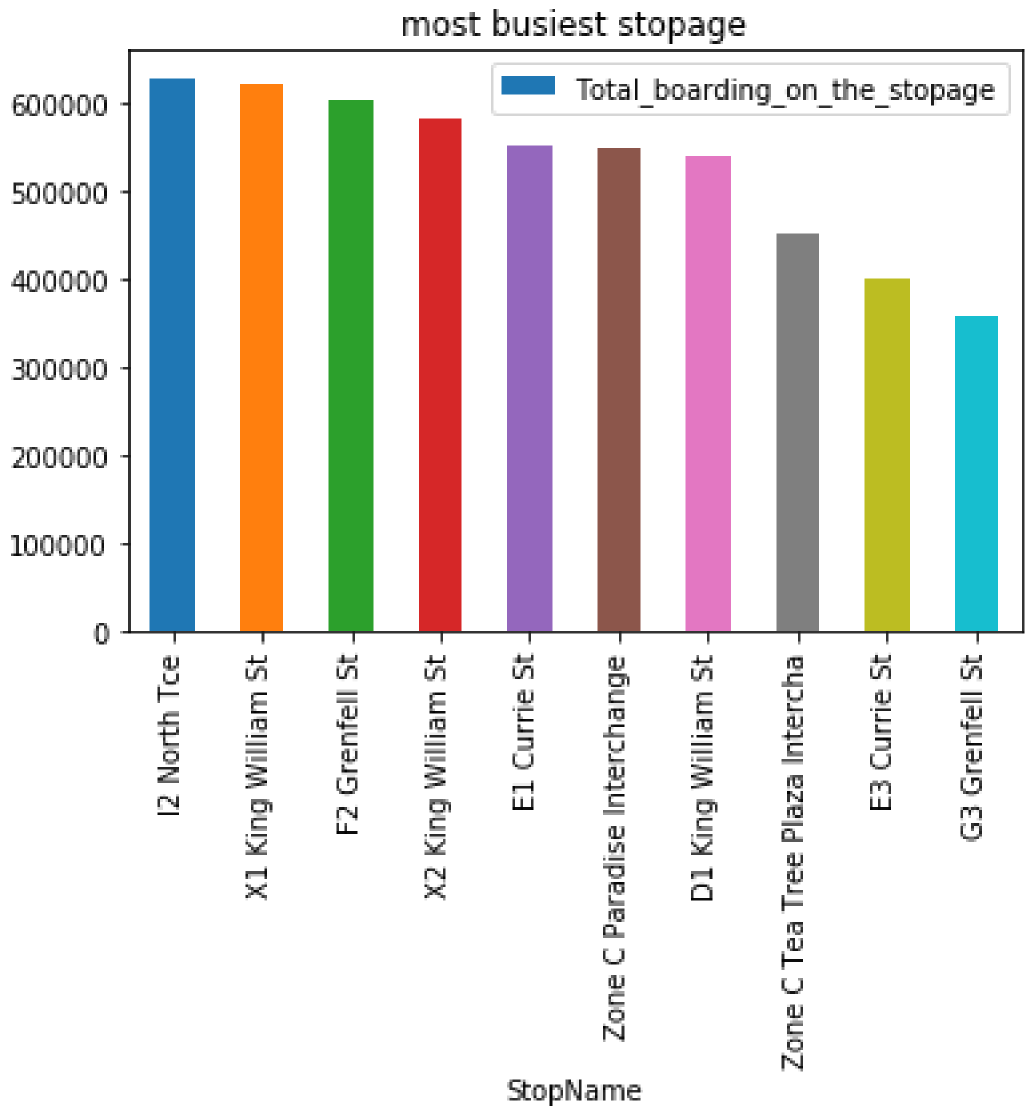
```
3025                       E3 Currie St                         399351
3039                     G3 Grenfell St                         356518
```

[29]:
```
#stopage with least no of boarding
stopageName_with_boarding.tail(10)
```

[29]:
```
                   StopName  Total_boarding_on_the_stopage
1845         45 Mcintyre Rd                            292
2318          57 Philip Hwy                            281
2732           75B Frick St                            275
58           109 Regency Rd                            274
1633        39D Glenloth Dr                            266
170          127 Lyndoch Rd                            264
3086  Strathalbyn South Tce                            227
1231          31 Glenroy St                            221
558           19 Gilles Rd                             215
294        145 The Esplanade                           175
```

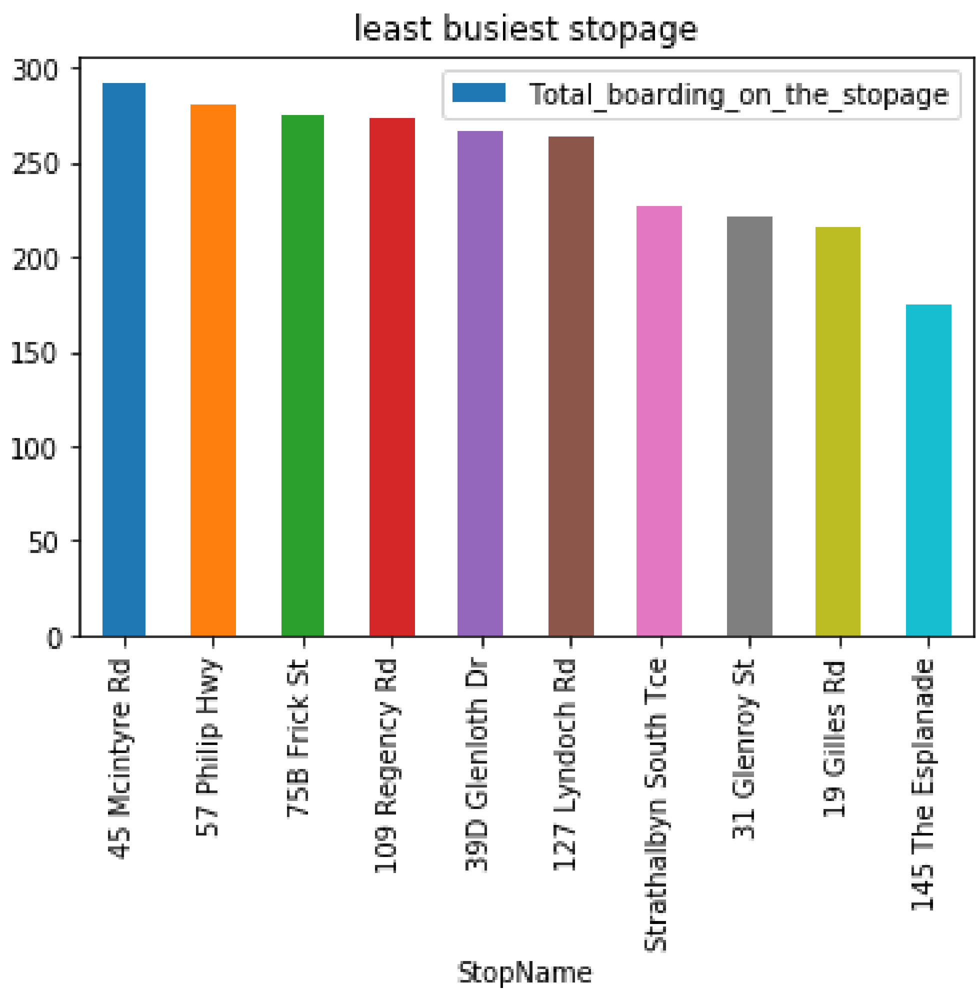
[30]:
```
ax = stopageName_with_boarding.head(10).plot.bar(x='StopName',␣
 ↪y='Total_boarding_on_the_stopage', rot=90)
ax.set_title("most busiest stopage")
```

[30]:
```
Text(0.5,1,'most busiest stopage')
```



[31]:
```
ax = stopageName_with_boarding.tail(10).plot.bar(x='StopName',␣
 ↪y='Total_boarding_on_the_stopage', rot=90)
ax.set_title("least busiest stopage")
```

[31]: Text(0.5,1,'least busiest stopage')



[32]:
```
data['WeekBeginning'].value_counts().mean()
```

[32]: 191508.66666666666

[33]:
```
# 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()
```

[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
```

[33]:
```
Index(['dist_from_centre', 'NumberOfBoardings'], dtype='object')
```

[33]:

	dist_from_centre	NumberOfBoardings
2392	86.471064	18905
2393	94.826409	321
2394	99.625655	1101
2395	99.665190	4373
2396	99.748995	21216

[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␣
 ↪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)
```

[35]:
```
#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
```

[36]:
```
print(outlier_)
```

```
0
```

[37]:
```
y = len(distance_10)+len(distance_10_50)+len(distance_50_100)
#+len(distance_100_more)
#print(y)
#print(total)
```

[38]:
```
print(total)
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)
#print("passanger, boarding the buses from the distance of 100Km and more from␣
↪the city center = ", (len(distance_100_more)/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
center =  2.5199323732375327
```

[39]:
```
#busiest route on weekly basis
#data.head(10)
# st_week_grp1 = pd.DataFrame(data.groupby(['StopName','WeekBeginning','type']).
↪agg({'NumberOfBoardings': ['sum', 'count']})).reset_index()
grouped_route = data.groupby(['RouteID']).agg({'NumberOfBoardings': ['sum',␣
↪'max']})
grouped_route.columns = ["_".join(x) for x in grouped_route.columns.ravel()]
```

[40]:
```
"""grouped_route = grouped_route.head().reset_index()
type(grouped_route)
grouped_route = grouped_route.sort_values("NumberOfBoardings_sum", ascending =␣
↪True)
```

```
#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)
#grouped_route["NumberOfBoardings_sum"] =␣
 ↪grouped_route["NumberOfBoardings_sum"] / 365
grouped_route.head(10)
grouped_route.shape"""
```

[40]:
```
'grouped_route =
grouped_route.head().reset_index()\ntype(grouped_route)\ngrouped_route =
grouped_route.sort_values("NumberOfBoardings_sum", ascending =
True)\n#stopageName_with_boarding =
stopageName_with_boarding.sort_values(\'Total_boarding_on_the_stopage\',
ascending = False)\n#stopage with most no of boarding\n#stopageName_with_boardin
g.head(10)\n#grouped_route["NumberOfBoardings_sum"] =
grouped_route["NumberOfBoardings_sum"] /
365\ngrouped_route.head(10)\ngrouped_route.shape'
```

….

[41]:
```
"""route_data = grouped_route[grouped_route['RouteID'] == "G10"]
route_data.head()"""
```

[41]:
```
'route_data = grouped_route[grouped_route[\'RouteID\'] ==
"G10"]\nroute_data.head()'
```

[ ]:
```

```