PUBLIC TRANSPORTATION ANALYSIS

PHASE 3: DEVELOPMENT PART 1

INTRODUCTION:

This project aims to improve public transportation by using data analysis and machine learning. We will explore the provided dataset, identify issues, and preprocess the data. Our goal is to predict service disruptions and analyze passenger feedback sentiment. With the power of machine learning, we'll uncover insights to enhance transportation services. This document outlines our step-by-step approach.

Data Exploration and Understanding

- Load the dataset using Pandas.
- Our focus will be on understanding the dataset's structure, consisting of 6 columns: TripID, RouteID, StopID, StopName, WeekBeginning, and NumberOfBoardings and understand the column meanings, and potential relationships between variables.
- Identify data quality issues, missing values, and outliers.

Data Preprocessing

- Select relevant columns for analysis (e.g., TripID, RouteID, StopName).
- Handle missing data, duplicates, and irrelevant entries.
- Convert data types if needed

Predicting Service Disruptions

- Innovation: Define how service disruption is determined from given features
- Select a set of features and Service Disruption as target feature
- Create DecisionTreeClassifier and train on 80% of dataset
- Test the classifier on remaining 20% of dataset

Sentiment Analysis for Passenger Feedback

A. Data Preprocessing

- For sentiment analysis, we need to extract and clean the text data containing passenger feedback.
- Load the dataset using Pandas.
- Select relevant columns for sentiment analysis (e.g., TripID, StopName).
- Remove duplicates and any irrelevant entries.
- Handle missing data, if any.

B. Text Preprocessing

- The text data may contain noise and irrelevant information. Text preprocessing is essential to ensure the accuracy of sentiment analysis.
- Tokenization: Split text into words.
- Lowercasing: Convert all text to lowercase.
- Removing special characters and punctuation.
- Stopword Removal: Eliminate common words (e.g., "the," "and") that do not carry sentiment.
- Lemmatization or stemming to reduce words to their base form.

C. Model Selection VADER Model for Sentiment Analysis:

- VADER is a specialized NLP model for sentiment analysis.
- It provides polarity and intensity scores.
- Suitable for real-time analysis and informal text.
- Ideal for public transportation feedback analysis.

D. Feature Engineering

- Create additional features or transformations that could enhance the analysis, such as time-based aggregations, seasonality, or weather data.
- Machine Learning Model Development
- Random Forest is an ensemble learning method that can be used for public transportation analysis as it can handle complex, multifaceted data.
- It combines multiple decision trees for enhanced accuracy and robustness.
- The Random Forest model has high accuracy, can handle large datasets, reduces overfitting, is robust to outliers and handles non-linearity.

E. Model Training and Validation

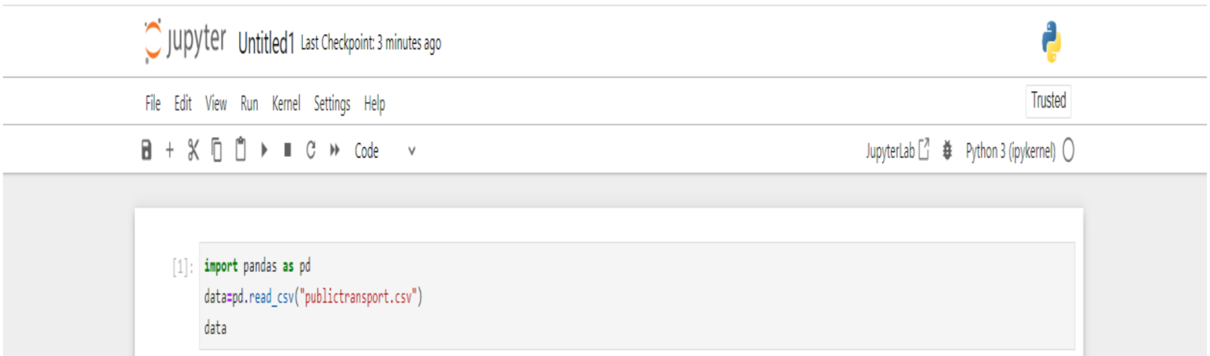
- Split the dataset into training and testing sets.
- Train the models for both service disruption prediction and overall analysis.
- Evaluate the model's performance using relevant metrics.
- Fine-tune the models if necessary.

F. Integration with IBM Cognos

- Integrate the machine learning and sentiment analysis results into IBM Cognos for streamlined data analytics and reporting.

G. Data Visualization and Reporting

- Create dashboards and reports in IBM Cognos to display insights from the analysis.
- Utilize charts, graphs, and maps to make the results easily interpretable for decision-makers.



Out[4]:

	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
2	23632	100	14132	175 Cross Rd	2013-06-30 00:00:00	1
3	23633	100	12266	Zone A Arndale Interchange	2013-06-30 00:00:00	2
4	23633	100	14147	178 Cross Rd	2013-06-30 00:00:00	1
...	...	...	...	...	...	...
10857229	13346	W91C	14629	21 Cashel St	2014-07-06 00:00:00	1
10857230	13346	W91C	14708	22 Cashel St	2014-07-06 00:00:00	3
10857231	13346	W91C	13709	2 Greenhill Rd	2014-07-06 00:00:00	1
10857232	13346	W91C	14029	10 East Av	2014-07-06 00:00:00	1
10857233	13346	W91C	13824	6 Leader St	2014-07-06 00:00:00	1

10857234 rows × 6 columns

In [10]:
```
data.head(10)
```

Out[10]:

	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
2	23632	100	14132	175 Cross Rd	2013-06-30 00:00:00	1
3	23633	100	12266	Zone A Arndale Interchange	2013-06-30 00:00:00	2
4	23633	100	14147	178 Cross Rd	2013-06-30 00:00:00	1
5	23634	100	13907	9A Marion Rd	2013-06-30 00:00:00	1
6	23634	100	14132	175 Cross Rd	2013-06-30 00:00:00	1
7	23634	100	13335	9A Holbrooks Rd	2013-06-30 00:00:00	1
8	23634	100	13875	9 Marion Rd	2013-06-30 00:00:00	1
9	23634	100	13045	206 Holbrooks Rd	2013-06-30 00:00:00	1

In [7]:
```
data.shape
```

Out[7]:
```
(10857234, 6)
```

In [8]:
```
data.columns
```

Out[8]:
```
Index(['TripID', 'RouteID', 'StopID', 'StopName', 'WeekBeginning',
       'NumberOfBoardings'],
      dtype='object')
```

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
TripID               0
RouteID              0
StopID               0
StopName             0
WeekBeginning        0
NumberOfBoardings    0
dtype: int64
```

In [11]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10857234 entries, 0 to 10857233
Data columns (total 6 columns):
 #   Column             Dtype
---  ------             -----
 0   TripID             int64
 1   RouteID            object
 2   StopID             int64
 3   StopName           object
 4   WeekBeginning      object
 5   NumberOfBoardings  int64
dtypes: int64(3), object(3)
memory usage: 497.0+ MB
```

In [16]:
```
df=data
```

In [19]:
```
a=df.TripID.value_counts()
a
```

Out[19]:
```
57020    2819
57018    2741
27478    2733
57041    2718
57029    2691
         ...
59297       1
3061        1
3414        1
3415        1
61163       1
Name: TripID, Length: 39282, dtype: int64
```

In [20]:
```
b=df.RouteID.value_counts()
b
```

Out[20]:
```
G10     358005
B10     332694
M44     331442
H30     326004
300     228373
         ...
FX1          1
FX10         1
FX8          1
FX3          1
FX2          1
Name: RouteID, Length: 619, dtype: int64
```

In [21]:

```
c=df.StopID.value_counts()
c
```

Out[21]:

```
13354    44089
13277    43339
13364    43265
13330    36992
13279    33800
         ...
17107        1
15420        1
15243        1
17805        1
17807        1
Name: StopID, Length: 7397, dtype: int64
```

In [22]:

```
d=df.WeekBeginning.value_counts()
d
```

Out[22]:

```
2014-03-02 00:00:00    217162
2014-05-18 00:00:00    215932
2014-05-11 00:00:00    214947
2014-06-01 00:00:00    213789
2014-05-04 00:00:00    212681
2014-03-23 00:00:00    212552
2014-03-16 00:00:00    212188
2014-02-23 00:00:00    212103
2013-09-08 00:00:00    211914
2014-04-27 00:00:00    211782
2014-05-25 00:00:00    211534
2014-03-30 00:00:00    211460
2013-09-01 00:00:00    210968
2014-04-06 00:00:00    210557
2013-08-25 00:00:00    209497
2013-11-17 00:00:00    209341
2013-11-24 00:00:00    208881
2013-10-20 00:00:00    208655
2013-12-01 00:00:00    208470
2014-06-15 00:00:00    208457
2014-06-08 00:00:00    208417
2013-09-15 00:00:00    208241
2014-02-16 00:00:00    208178
2013-10-27 00:00:00    207971
2013-09-22 00:00:00    207769
2013-12-08 00:00:00    207353
2013-10-13 00:00:00    207351
2013-08-04 00:00:00    207082
2013-11-03 00:00:00    206863
2013-11-10 00:00:00    206853
2014-06-29 00:00:00    206138
2013-07-28 00:00:00    205492
2013-08-11 00:00:00    205385
2013-08-18 00:00:00    203852
2013-07-21 00:00:00    201257
2014-06-22 00:00:00    200950
2014-02-09 00:00:00    197978
2014-01-19 00:00:00    196344
2013-10-06 00:00:00    195830
2014-03-09 00:00:00    195200
2013-12-15 00:00:00    194102
2014-02-02 00:00:00    192507
2013-09-29 00:00:00    192023
2013-07-07 00:00:00    190543
2014-04-13 00:00:00    190060
2013-07-14 00:00:00    187192
2014-01-05 00:00:00    186105
2014-04-20 00:00:00    185080
2013-06-30 00:00:00    182229
2014-01-26 00:00:00    180259
2014-01-12 00:00:00    178456
2013-12-29 00:00:00    168771
2013-12-22 00:00:00    163331
2014-07-06 00:00:00    149202
Name: WeekBeginning, dtype: int64
```

In [24]:
```
e=df.NumberOfBoardings.value_counts()
e
```

Out[24]:
```
1      4270812
2      2057245
3      1128820
4       731537
5       502763
        ...
547          1
539          1
443          1
474          1
342          1
Name: NumberOfBoardings, Length: 400, dtype: int64
```

In [29]:
```
data['WeekBeginning'] = pd.to_datetime(data['WeekBeginning']).dt.date
data['WeekBeginning'][1]
```

Out[29]:
```
datetime.date(2013, 6, 30)
```

In [38]:
```
grouped = data.groupby(['StopName','WeekBeginning',]).agg({'NumberOfBoardings': ['sum', 'count','max']})
grouped
```

Out[38]:

		NumberOfBoardings		
		sum	**count**	**max**
StopName	**WeekBeginning**			
1 Anzac Hwy	**2013-06-30**	1003	378	51
	2013-07-07	783	360	28
	2013-07-14	843	343	45
	2013-07-21	710	356	28
	2013-07-28	898	379	41
...	**...**	...	...	...
Zone I Salisbury Interchange	**2014-06-08**	822	117	44
	2014-06-15	965	113	39
	2014-06-22	896	111	58
	2014-06-29	1052	113	39
	2014-07-06	534	90	21

207864 rows × 3 columns

In [40]:
```
st_week_grp = pd.DataFrame(grouped).reset_index()
st_week_grp1 = pd.DataFrame(st_week_grp.groupby('StopName')["WeekBeginning"].count()).reset_index()
st_week_grp1.head()
```

Out[40]:

	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

In [49]:
```
stopListName = list(st_week_grp1[st_week_grp1['WeekBeginning'] == 54]['StopName'])
stopListName[1:30]
```

Out[49]:
```
['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',
 '1 King William Rd',
 '1 Port Rd',
 '1 Sir Donald Bradman Dr',
 '1 Sir Edwin Smith Av',
 '1 Unley Rd',
 '10  Holbrooks Rd',
 '10  Marion Rd',
 '10  Portrush Rd',
 '10 Airport Rd',
 '10 Anzac Hwy',
 '10 Ashley St',
 '10 Belair Rd',
 '10 Churchill Rd',
 '10 East Av',
 '10 Fullarton Rd',
 '10 Garden Tce',
 '10 Glen Osmond Rd',
 '10 Goodwood Rd',
 '10 Greenhill Rd',
 '10 Harrow Tce']
```

In [59]:
```
stopageName_with_boarding = data.groupby(['StopName']).agg({'NumberOfBoardings': ['sum']}).reset_index()
```

Out[60]:

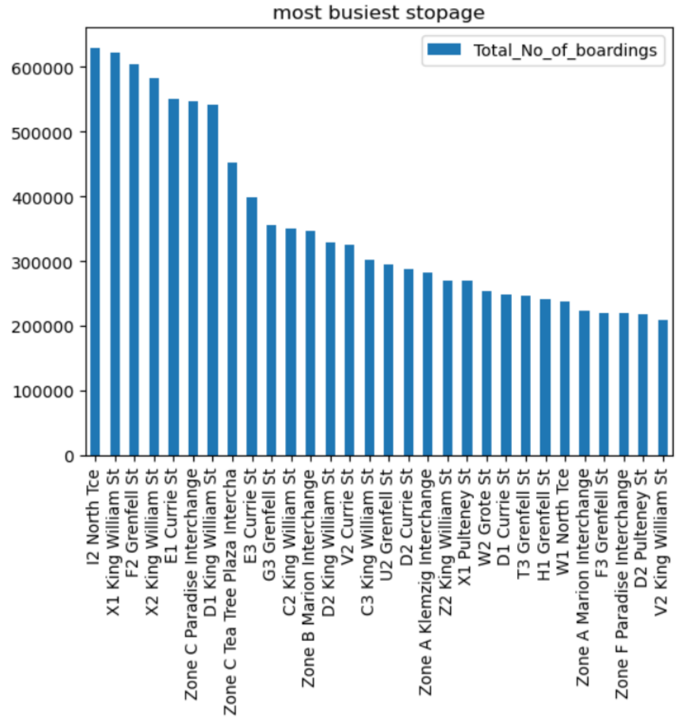
	stopName	Total_No_of_boardings
0	1 Anzac Hwy	39429
1	1 Bartels Rd	8412
2	1 Botanic Rd	14868
3	1 Frome Rd	67458
4	1 Fullarton Rd	585

In [63]:
```
stopageName_with_boarding = stopageName_with_boarding.sort_values("Total_No_of_boardings", ascending = False)
#stopage with most no of boarding
stopageName_with_boarding.head(10)
```

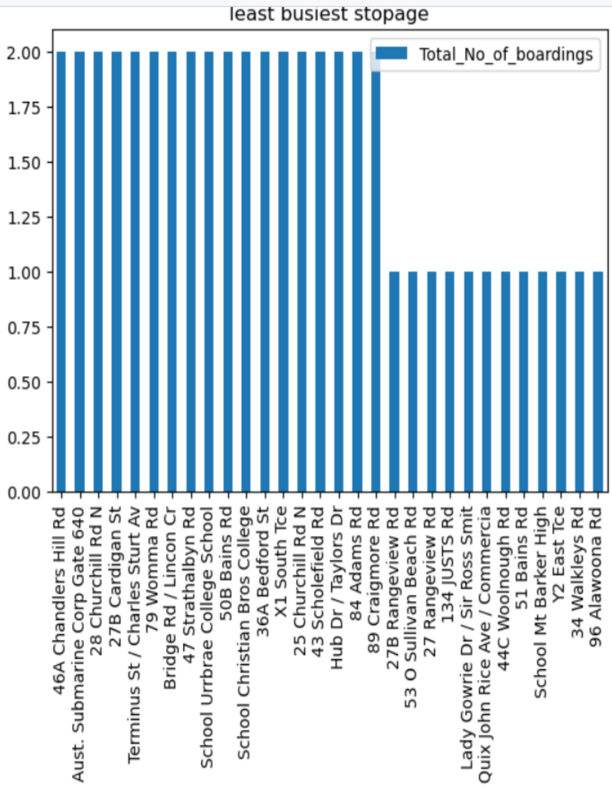
Out[63]:

	stopName	Total_No_of_boardings
3841	I2 North Tce	628859
4023	X1 King William St	622099
3807	F2 Grenfell St	604149
4029	X2 King William St	583227
3791	E1 Currie St	550396
4120	Zone C Paradise Interchange	547709
3784	D1 King William St	541046
4124	Zone C Tea Tree Plaza Intercha	451960
3796	E3 Currie St	399351
3819	G3 Grenfell St	356518

In [76]:
```
busiestStop = stopageName_with_boarding.head(30).plot.bar(x="stopName", y="Total_No_of_boardings", rot=90)
plt.title("most busiest stopage")
plt.legend()
```



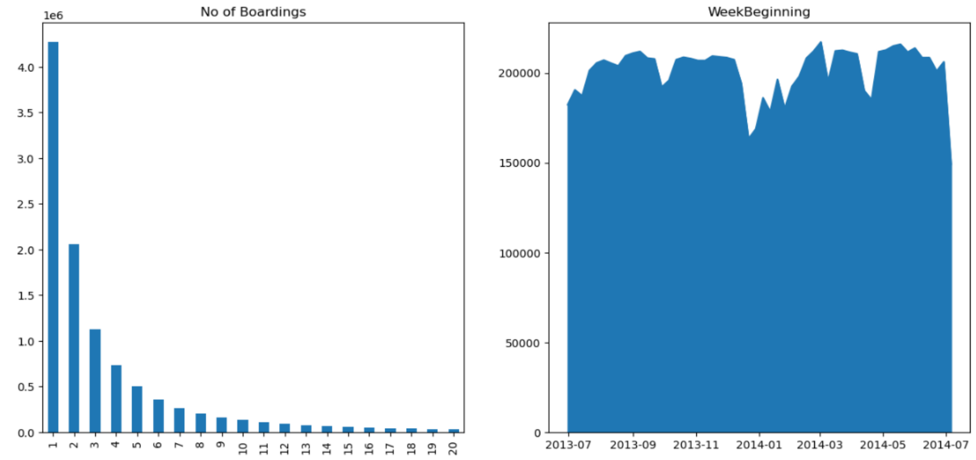
In [75]:
```
leastBusiestStop = stopageName_with_boarding.tail(30).plot.bar(x='stopName', y='Total_No_of_boardings', rot=90)
plt.title("least busiest stopage")
plt.legend()
```

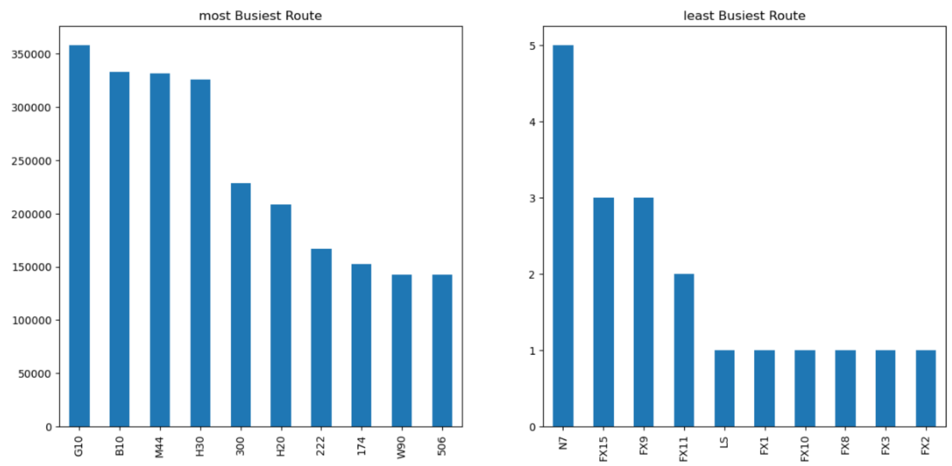


In [30]:

```
import matplotlib.pyplot as plt
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])
```

Out[30]: `<Axes: title={'center': 'least Busiest Route'}>`





CONCLUSION:

By following these steps, we can effectively enhance public transportation analysis by predicting service disruptions and analyzing passenger sentiment, ultimately leading to improvements in the transportation system.