SUBMITTED BY: JAYAPRIYADHARSHINI V

PUBLIC TRANSPORTATION EFFICENCY ANALYSIS

NM ID: au211521106068

ABSTRACT:

Reducing traffic, minimizing environmental impact, and promoting sustainable urban development all depend on effective public transit networks. In this study, we investigate how data analysis methods might be applied to improve the effectiveness of public transportation networks using IBM Cognos, a potent business intelligence tool. We examine vast amounts of transportation data using Cognos' capabilities in an effort to find trends, enhance system performance, and optimize routes.

The first step of the study is gathering and combining data from a variety of transportation sources, such as passenger counts, vehicle locations, and schedule adherence. The raw data is cleaned up and transformed into a consistent format that can be analyzed by using Cognos' data cleaning and transformation tools.

After that, we create thorough data models in Cognos Framework Manager by creating linkages and hierarchies that offer a clear picture of the data environment for transportation. Using Cognos Report Studio and Cognos Dashboard, these models form the basis for producing informative reports and dynamic dashboards.

Advanced data analysis methods, such clustering, statistical analysis, and predictive modeling, provide us important new insights into the behavior of passengers and provide information on popular routes, peak travel periods, and service interruptions. The current transportation system's inefficiencies are found using these findings.Targeted optimization strategies are created and put into practice based on the analysis's findings. These tactics could include rerouting cars to avoid traffic jams, modifying bus schedules, or maximizing rail frequencies.

Cognos makes it possible to monitor the functioning of the transportation system in real time, allowing for quick corrections and ongoing development.The study's conclusions offer data-driven suggestions to legislators, city planners, and transportation authorities to improve the effectiveness of public transit systems. This research advances smart and sustainable urban mobility by utilizing IBM Cognos for data analysis, guaranteeing that public transportation systems are not only effective but also adaptable to the changing needs of metropolitan areas.

Introduction:

As cities continue to face the growing challenge of improving their public transportation systems, data analytics emerges as a promising solution. By harnessing the power of data, public transport authorities can now analyze various factors that affect efficiency and make informed decisions to enhance the overall performance. This article explores the critical role of data analytics in improving public transport efficiency and its potential to transform the way we commute.

Problem definition:

"Improving Public Transportation efficiency and Reliability"

Public transportation is a vital component of urban infrastructure, yet it faces persistent challenges that hinder its effectiveness. The problem of public transportation encompasses the need to enhance accessibility and reliability for passengers while addressing key issues such as congestion, environmental impact, and equitable service provision.

Key components of this problem include:

1. Accessibility: Ensuring that public transportation is available and convenient for all members of the community, including those with disabilities, the elderly, and those without access to private vehicles.

2. Reliability: Minimizing delays, service interruptions, and unpredictability in public transportation schedules to provide passengers with consistent and dependable service.

3. Congestion: Managing and reducing traffic congestion on urban roadways by promoting the use of public transportation as a viable alternative to private vehicles.

4. Environmental Impact: Mitigating the environmental footprint of public transportation by promoting cleaner technologies and implementing sustainable practices.

Definition of Public Transport Efficiency:

Public transport efficiency refers to the ability of a public transportation system to provide reliable, timely, and convenient services to commuters while maximizing resources and minimizing environmental impact. It involves analyzing factors such as punctuality, frequency, connectivity, capacity utilization, and customer satisfaction to identify areas for improvement and optimize the system's performance. Data analytics plays a crucial role in measuring and understanding these factors, enabling authorities to make data-driven decisions for enhancing overall efficiency and creating a seamless commuting experience.

DESIGN THINKING:

1.Problem Identification:

   - Identify specific issues within public transportation, such as delays, overcrowding, or route optimization, that data analytics can help address.

2.Data Collection:

   - Gather relevant data from various sources, including GPS tracking, ticketing systems, maintenance records, and customer feedback.

3. Data Cleaning and Preparation:

   - Clean and preprocess the collected data to remove inconsistencies, missing values, and errors.

   - Format data for analysis and ensure it's in a suitable structure for modeling.

4. Exploratory Data Analysis (EDA):

   - Conduct EDA to gain insights into the data. Visualize data patterns, correlations, and anomalies.

   - Identify key variables and factors that may contribute to transportation issues.

5. Model Selection:

- Choose appropriate data analytics and machine learning models based on the problem you're addressing.

- Common models include regression for demand forecasting, clustering for route optimization, and time series analysis for scheduling.

6. Feature Engineering:

-Create or transform features that are relevant to your modelling task.

7. Model Training:

- Train the selected models on historical data to learn patterns and relationships.

- Use a portion of the data for training and reserve another portion for testing and validation.

8. Model Evaluation:

- Assess the performance of your models using appropriate metrics. For example, Mean Absolute Error (MAE) for regression or F1-score for classification tasks.

- Fine-tune models and hyperparameters to improve their accuracy

9.Deployment:

-Implement the data analytics solutions into the public transportation system.

10. Monitoring and Maintenance:

- Continuously monitor the performance of your data analytics solutions in a real-world environment.

- Implement alerts and triggers to identify and address issues as they arise.

11.Feedback Loop:

- Gather user feedback and data on the effectiveness of your solutions.

- Use this feedback to make necessary adjustments, updates, and improvements to your data analytics models and processes.

12. Scale and Iterate:

- If successful, consider scaling the solutions to cover a larger portion of the transportation network.

- Continue to iterate and refine your models as new data becomes available or as transportation needs evolve.

13. Policy and Decision Making:

- Share insights and recommendations derived from data analytics with transportation authorities and policymakers to inform decisions related to infrastructure, routes, and services.

14. Public Communication:

- Communicate improvements and changes to the public, ensuring transparency and addressing concerns or feedback

STEPS FOR IMPLEMENTING MACHINE LEARNING ALGORITHM:

**1. Model Training:**

* Train the selected machine learning models using the training data.

**2. Model Evaluation:**

* Evaluate the models using the testing data. Common evaluation metrics include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) for regression problems, and accuracy, precision, recall for classification problems.

**3. Hyperparameter Tuning:**

* Fine-tune the hyperparameters of your models to improve their performance. This can be done using techniques like grid search or random search.

**4. Deployment:**

* Once you have a well-performing model, deploy it to make predictions on new, unseen data. This could involve setting up a real-time prediction system or integrating it into existing software infrastructure.

[1]:

%matplotlibinline

**importnumpyasnp***#linearalgebra*

**importpandasaspd***#data processing, CSV fileI/O (e.g.pd.read\_csv)*

**importmatplotlib.pyplotaspltimportdatetime**

**importos**

**frommathimport**sqrt

**importwarnings**

*##For Multiple Output in single cell*

**fromIPython.core.interactiveshellimport**InteractiveShellInteractiveShell.ast\_node\_interactivity="all"warnings.filterwarnings('ignore')

[2]:

data=pd.read\_csv('../content/20140711.CSV')data.shape

data.head(10)

[2]: (31767,6)

[2]: TripID RouteID StopID StopName WeekBeginning \0 23631 100.0 14156.0 181CrossRd 2013-06-3000:00:00

1 23631 100.0 14144.0 177CrossRd 2013-06-3000:00:00

2 23632 100.0 14132.0 175CrossRd 2013-06-3000:00:00

3 23633 100.0 12266.0 ZoneAArndaleInterchange 2013-06-3000:00:00

4 23633 100.0 14147.0 178CrossRd 2013-06-3000:00:00

5 23634 100.0 13907.0 9A MarionRd 2013-06-3000:00:00

6 23634 100.0 14132.0 175CrossRd 2013-06-3000:00:00

7 23634 100.0 13335.0 9A HolbrooksRd 2013-06-3000:00:00

8 23634 100.0 13875.0 9 MarionRd 2013-06-3000:00:00

9 23634 100.0 13045.0 206HolbrooksRd 2013-06-3000:00:00

NumberOfBoardings0 1.0

1 1.0

2 1.0

3 2.0

|  |  |
| --- | --- |
| 4 | 1.0 |
| 5 | 1.0 |
| 6 | 1.0 |
| 7 | 1.0 |
| 8 | 1.0 |
| 9 | 1.0 |

[3]:

out\_geo=pd.read\_csv('../content/output\_geo.csv')out\_geo.shape

out\_geo.head()

[3]: (4165,10)

[3]: accuracy formatted\_address \

1. ROOFTOP 181CrossRd,WestbourneParkSA5041,Australia
2. ROOFTOP 177CrossRd,WestbourneParkSA5041,Australia
3. ROOFTOP 175CrossRd,WestbourneParkSA5041,Australia
4. GEOMETRIC\_CENTER ZoneAArndaleInterchange-Southside,Kilke…
5. ROOFTOP 178CrossRd,MalvernSA5061,Australia

google\_place\_id input\_string latitude \

1. ChIJKT7I9rbPsGoRVHMHkIy-Oyk 181CrossRd-34.966656
2. ChIJ-VFZ87bPsGoRyfVgC5qbPpE 177CrossRd-34.966607
3. ChIJIztlirbPsGoR38KRk76kPFI 175CrossRd-34.966758
4. ChIJn0C1hCPGsGoRIWvCdhF1RIg ZoneAArndaleInterchange-34.875160
5. ChIJycNiylvOsGoRdhfq9GKnpq0 178CrossRd-34.964960

longitude number\_of\_resultspostcodestatus \

|  |  |  |  |
| --- | --- | --- | --- |
| 0 138.592148 | 1 | 5041 | OK |
| 1 138.592301 | 1 | 5041 | OK |
| 2 138.592715 | 1 | 5041 | OK |
| 3 138.551628 | 1 | 5009 | OK |
| 4 138.611477 | 1 | 5061 | OK |

type

1. street\_address
2. street\_address
3. street\_address
4. bus\_station,establishment,point\_of\_interest,tr…
5. street\_address

[4]:

*#DistanceFromCentre:Distancemeasurefromthecitycentre*

*#ForCalculatingDistancebetweencentrewithotherbusstopsbyusing*␣

↪*Longitude and Latitude*

*#wehave used the Haversine formula*

**frommath import**sin,cos,sqrt,atan2,radians

**def**calc\_dist(lat1,lon1):

*##approximateradius of earthin 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 | formatted\_address | | \ |
|  |  | 0 | ROOFTOP | 181CrossRd,WestbourneParkSA5041,Australia | |  |
|  |  | 1 | ROOFTOP | 177CrossRd,WestbourneParkSA5041,Australia | |  |
|  |  | 2 | ROOFTOP | 175CrossRd,WestbourneParkSA5041,Australia | |  |
|  |  | 3 | GEOMETRIC\_CENTER | ZoneAArndaleInterchange-Southside,Kilke… | |  |
|  |  | 4 | ROOFTOP | 178CrossRd,MalvernSA5061,Australia | |  |
|  | | | google\_place\_id input\_string | | latitude | \ |
| 0 | | | ChIJKT7I9rbPsGoRVHMHkIy-Oyk 181CrossRd | | -34.966656 |  |
| 1 | | | ChIJ-VFZ87bPsGoRyfVgC5qbPpE 177CrossRd | | -34.966607 |  |
| 2 | | | ChIJIztlirbPsGoR38KRk76kPFI 175CrossRd | | -34.966758 |  |
| 3 | | | ChIJn0C1hCPGsGoRIWvCdhF1RIg ZoneAArndaleInterchange | | -34.875160 |  |
| 4 | | | ChIJycNiylvOsGoRdhfq9GKnpq0 178CrossRd | | -34.964960 |  |

longitude number\_of\_resultspostcodestatus \

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 138.592148 | 1 | 5041 | OK |  |
| 1 138.592301 | 1 | 5041 | OK |  |
| 2 138.592715 | 1 | 5041 | OK |  |
| 3 138.551628 | 1 | 5009 | OK |  |
| 4 138.611477 | 1 | 5061 | OK |  |
|  |  |  |  | type dist\_from\_centre |

|  |  |
| --- | --- |
| 0 street\_address | 5.180961 |
| 1 street\_address | 5.172525 |
| 2 street\_address | 5.180709 |
| 3 bus\_station,establishment,point\_of\_interest,tr… | 7.057549 |
| 4 street\_address | 4.900099 |

[7]:

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

*##Fillthe 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]:

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]:

data['WeekBeginning']=pd.to\_datetime(data['WeekBeginning']).dt.datedata['WeekBeginning'][1]

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

STEPS FOR CLEANING DATASET USING PYTHON:

1.Handle Missing Values: Locate and deal with any missing data points. Rows with missing data can either be eliminated or filled in utilizing methods like mean, median, or interpolation, depending on the circumstances.

2. Remove Duplicates: - Search for and eliminate records that are identical, particularly if the dataset was produced from various sources. Analysis results may be skewed by duplicates.

3. Data Type Conversion:- Ascertain that the data types are suitable for analysis. For instance, dates should be formatted as dates, and categorical variables should have the proper labels. Numerical data should also be presented in its proper format.

4. Check for Outliers: - Find anomalies in the numerical data that can have a big impact on the analysis. Choose whether to eliminate outliers or to change the data to lessen their influence.

5.Normalize and Standardize Data:- If you intend to employ methods that are sensitive to the scale of variables, standardize or normalize numerical data. It is possible to apply normalization (scaling values between 0 and 1) or standardization (subtracting the mean and dividing by the standard deviation).

6. Validate Categorical Data: Verify that categorical variables only contain valid values by validating them. Correct categories that are inconsistent or misspelled.

7. Address Inconsistent Data: Keep an eye out for errors made when entering data, particularly in text areas. For instance, when referring to the same area, "NY," "New York," and "New York City" should all be used consistently.

8. Check Integrity Constraints - Verify that the connections between various columns make sense. For instance, the time of arrival should be later than the time of departure.

9. Extract pertinent data from text fields using parsing and extraction software. For instance, if a field contains both a date and an hour, separate them into different columns.

10. Validate the coordinates in the dataset if it contains geographic information to make sure they are within the expected range for the area of interest.

11. Validate data across various fields via cross-field validation. Make sure the estimated speed is within acceptable bounds, for instance, if you have a distance field and a time field.

12. note Changes: Keep a note of any modifications that were made during cleaning. This documentation is useful for reproducibility and transparency.

13. Examine the Clean Dataset:On the cleaned dataset, run preliminary analysis to make sure the data behaves as predicted. This process helps identify any problems that might have gone unnoticed during cleaning.

OBJECTIVES OF ANALYSIS OF DATASET:

1. Recognize Trends and Patterns:

The goal is to find trends, patterns, and connections in the data.

Why: To understand the relationships between variables and how they alter as a function of time, place, or other dimensions.

1. Make Forecasts or Predictions:

The goal is to create predictive models that can predict future trends or results.

Why: To foresee events in the future, make wise decisions, and develop plans based on what is likely to happen.

1. Process improvement

The goal is to locate process bottlenecks, inefficiencies, or potential improvement areas.

Why: To boost output, cut costs, and increase overall effectiveness in a variety of industries including manufacturing, logistics, or service provision.

1. Targeting and segmentation

Segment the data into groups based on traits and choose particular groups to focus marketing or intervention efforts on.

Why? To better comprehend various customer categories, hone marketing tactics, and raise client satisfaction.

1. Detection of Anomalies:

Goal: Spot any odd trends or anomalies in the data.

Why: To catch fraud, mistakes, flaws, or other odd occurrences that may need further care or investigation.

1. Resource allocation optimization

Determine the most efficient way to distribute resources like money, labor, or time.

Why: To ensure that resources are distributed where they are most needed and to make the most of the impact of those resources.

1. Analyze any policies or interventions:

Determine the effects of particular interventions, laws, or changes in variables.

Why: To assess the efficacy of activities performed, such as policy reforms, educational initiatives, or public health interventions.

1. Analysis of consumer behavior

Understanding consumer behavior, tastes, and shopping habits is the goal.

Why: To increase product and service offerings, better the consumer experience, and customize marketing methods.

1. Churn Forecast:

Determine which clients are most likely to discontinue utilizing a service.

Why: To put retention tactics in place, lower client attrition, and keep a steady customer base.

1. Systems of recommendations:

Create algorithms to provide users with recommendations for goods, services, or content.

Why: To promote customer engagement, boost revenues, and improve user experience on channels like social media, e-commerce, and streaming services.

1. Analysis of social networks:

Analyze connections and communications within networks of people or organizations.

Why? To comprehend social structures, recognize powerful nodes, and investigate information flow throughout networks.

1. Scientific Research:

Make scientific findings by analyzing experimental or observational data.

Why: To further scientific understanding, support theories, and derive important conclusions from empirical facts.

%matplotlib inline

importnumpyasnp *#linearalgebra*

importpandasaspd *#dataprocessing,CSV fileI/O (e.g. pd.read\_csv)*

importmatplotlib.pyplotaspltimportdatetime

importos

from math import sqrtimportwarnings

*##ForMultipleOutputinsinglecell*

from IPython.core.interactiveshell import InteractiveShellInteractiveShell.ast\_node\_interactivity= "all"warnings.filterwarnings('ignore')

data=pd.read\_csv('/content/20140711.xslb.csv')data.shape

data.head(10)(82769,6)

TripIDRouteID StopID StopName

WeekBeginning\

0 23631 100 14156 181CrossRd30-06-2013

00:00

1 23631 100 14144 177CrossRd30-06-2013

00:00

2 23632 100 14132 175CrossRd30-06-2013

00:00

3 23633 100 12266Zone AArndaleInterchange 30-06-2013

00:00

4 23633 100 14147 178CrossRd30-06-2013

00:00

5 23634 100 13907 9AMarionRd30-06-2013

00:00

6 23634 100 14132 175CrossRd30-06-2013

00:00

7 23634 100 13335 9AHolbrooksRd30-06-2013

00:00

8 23634 100 13875 9MarionRd30-06-2013

00:00

9 23634 100 13045 206HolbrooksRd30-06-2013

00:00

NumberOfBoardings0 1.0

1 1.0

2 1.0

3 2.0

4 1.0

5 1.0

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 6 | 1.0 |
| 7 | 1.0 |
| 8 | 1.0 |
|  | 9 | 1.0 |

out\_geo= pd.read\_csv('/content/output\_geo.csv')out\_geo.shape

out\_geo.head()(4165,10)

accuracy formatted\_address

\

0 ROOFTOP 181 Cross Rd, Westbourne Park SA 5041, Australia1 ROOFTOP 177 Cross Rd, Westbourne Park SA 5041, Australia2 ROOFTOP 175 Cross Rd, Westbourne Park SA 5041, Australia3GEOMETRIC\_CENTERZoneA ArndaleInterchange-Southside, Kilke...

4 ROOFTOP 178 Cross Rd, Malvern SA5061, Australia

google\_place\_id input\_string latitude

\

1. ChIJKT7I9rbPsGoRVHMHkIy-Oyk 181CrossRd-34.966656
2. ChIJ-VFZ87bPsGoRyfVgC5qbPpE 177 Cross Rd -34.9666072ChIJIztlirbPsGoR38KRk76kPFI 175 Cross Rd -34.9667583ChIJn0C1hCPGsGoRIWvCdhF1RIgZone A Arndale Interchange -34.8751604ChIJycNiylvOsGoRdhfq9GKnpq0 178CrossRd-34.964960

longitudenumber\_of\_resultspostcodestatus \

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 138.592148 | 1 | 5041 | OK |  |
| 1 | 138.592301 | 1 | 5041 | OK |
| 2 | 138.592715 | 1 | 5041 | OK |
| 3 | 138.551628 | 1 | 5009 | OK |
| 4 | 138.611477 | 1 | 5061 | OK |
|  |  |  |  |  |
|  |  |  |  |  | type |
| 1. street\_address 2. street\_address 3. street\_address 4. bus\_station,establishment,point\_of\_interest,tr... 5. street\_address | | | | | |

*#DistanceFromCentre:Distancemeasurefromthecitycentre*

*#For Calculating Distance between centre with other bus stops by usingLongitude andLatitude*

*#wehaveused theHaversineformula*

from math import sin, cos, sqrt, atan2, radiansdef calc\_dist(lat1,lon1):

*##approximateradius of earthinkm*

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))returnR\*c

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

out\_geo.head()

accuracy formatted\_address

\

0 ROOFTOP 181 Cross Rd, Westbourne Park SA 5041, Australia1 ROOFTOP 177 Cross Rd, Westbourne Park SA 5041, Australia2 ROOFTOP 175 Cross Rd, Westbourne Park SA 5041, Australia3GEOMETRIC\_CENTERZoneA ArndaleInterchange-Southside, Kilke...

4 ROOFTOP 178 Cross Rd, Malvern SA5061, Australia

google\_place\_id input\_string latitude

\

1. ChIJKT7I9rbPsGoRVHMHkIy-Oyk 181CrossRd-34.966656
2. ChIJ-VFZ87bPsGoRyfVgC5qbPpE 177 Cross Rd -34.9666072ChIJIztlirbPsGoR38KRk76kPFI 175 Cross Rd -34.9667583ChIJn0C1hCPGsGoRIWvCdhF1RIgZone A Arndale Interchange -34.8751604ChIJycNiylvOsGoRdhfq9GKnpq0 178CrossRd-34.964960

longitudenumber\_of\_resultspostcodestatus \

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 138.592148 | 1 | 5041 | OK |
| 1 | 138.592301 | 1 | 5041 | OK |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | |
| 2 | 138.592715 | 1 | 5041 | OK |
| 3 | 138.551628 | 1 | 5009 | OK |  |  |
| 4 | 138.611477 | 1 | 5061 | OK |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | type | dist\_from\_centre |

|  |  |
| --- | --- |
| 0 street\_address | 5.180961 |
|  |  |
| 1 street\_address | 5.172525 |
|  |  |
| 2 street\_address | 5.180709 |
|  |  |
| 3bus\_station,establishment,point\_of\_interest,tr... | 7.057549 |
|  |  |
| 4 street\_address | 4.900099 |

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

*##Fill the missing valueswithmode*

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

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

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

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)

data.info()

<class 'pandas.core.frame.DataFrame'>RangeIndex: 82769 entries, 0 to 82768Data columns(total6columns):

#Column Non-NullCount Dtype

* 1. TripID 82769non-nullint64
  2. RouteID 82769non-nullint64
  3. StopID 82769non-nullint64
  4. StopName 82769non-null object
  5. WeekBeginning 82768non-null object

5 NumberOfBoardings82768 non-nullfloat64dtypes:float64(1),int64(3), object(2)

memory usage:3.8+MBdata.head(3)

TripIDRouteIDStopID StopName WeekBeginningNumberOfBoardings

0 23631 100 14156181 Cross Rd30-06-2013 00:00

1.0

1 23631 100 14144177 Cross Rd30-06-2013 00:00

1.0

2 23632 100 14132175 Cross Rd30-06-2013 00:00

1.0

data.tail(3)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TripID | RouteID | StopID | StopName | WeekBeginning | \ |
| 82766 13354 | 100 | 14152 | 179Cross Rd | 01-06-201400:00 |  |
| 82767 13354 | 100 | 12352 | Woodville | 01-06-201400:00 |  |
| 82768 13354 | 100 | 13767 | 8H | NaN |  |

NumberOfBoardings

82766 1.0

82767 1.0

82768 NaN

data.describe()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | TripID | RouteID | StopID | NumberOfBoardings |
| count | 82769.000000 | 82769.0 | 82769.000000 | 82768.000000 |
| mean | 28094.843166 | 100.0 | 13475.915923 | 3.604545 |
| std | 18674.361134 | 0.0 | 745.937586 | 7.038205 |
| min | 5605.000000 | 100.0 | 12213.000000 | 1.000000 |
| 25% | 5651.000000 | 100.0 | 12839.000000 | 1.000000 |
| 50% | 44679.000000 | 100.0 | 13627.000000 | 2.000000 |
| 75% | 44704.000000 | 100.0 | 14099.000000 | 4.000000 |
| max | 44729.000000 | 100.0 | 17881.000000 | 181.000000 |

data.isna().sum()TripID 0

RouteID 0

StopID 0

StopName 0

WeekBeginning 1

NumberOfBoardings 1

dtype: int64

data.dropna(inplace=True)data.isna().sum()

TripID 0

RouteID 0

StopID 0

StopName 0

WeekBeginning 0

NumberOfBoardings 0

dtype: int64

data[data.duplicated()]

EmptyDataFrame

Columns: [TripID, RouteID, StopID, StopName, WeekBeginning,NumberOfBoardings]

Index:[]

data['TripID']=data['TripID'].astype(float)data.head(3)

TripIDRouteIDStopID StopName WeekBeginningNumberOfBoardings

023631.0 100 14156181 CrossRd30-06-201300:00

1.0

123631.0 100 14144177 CrossRd30-06-201300:00

1.0

223632.0 100 14132175 CrossRd30-06-201300:00

1.0

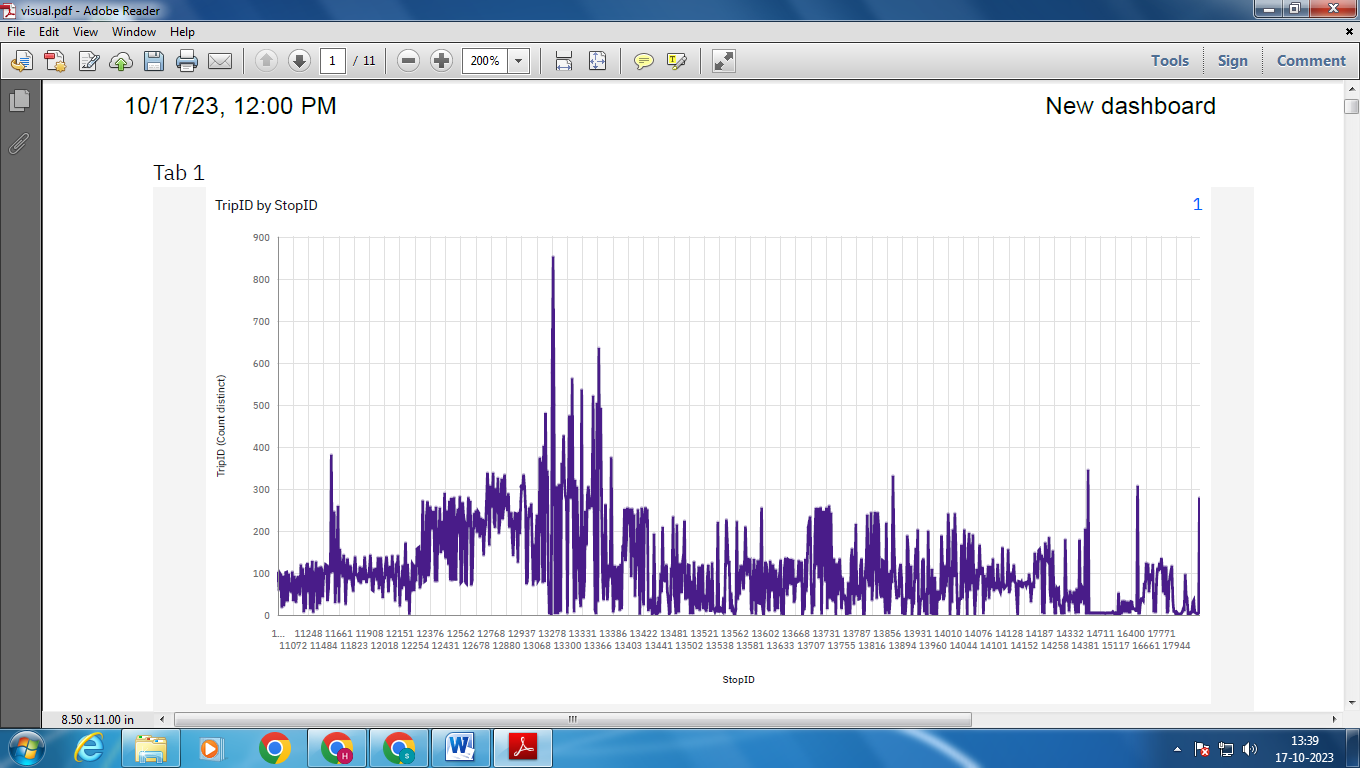
data['StopName'] = data['StopName'].str.strip()data['WeekBeginning'] =pd.to\_datetime(data['WeekBeginning'])from sklearn.preprocessingimportStandardScaler

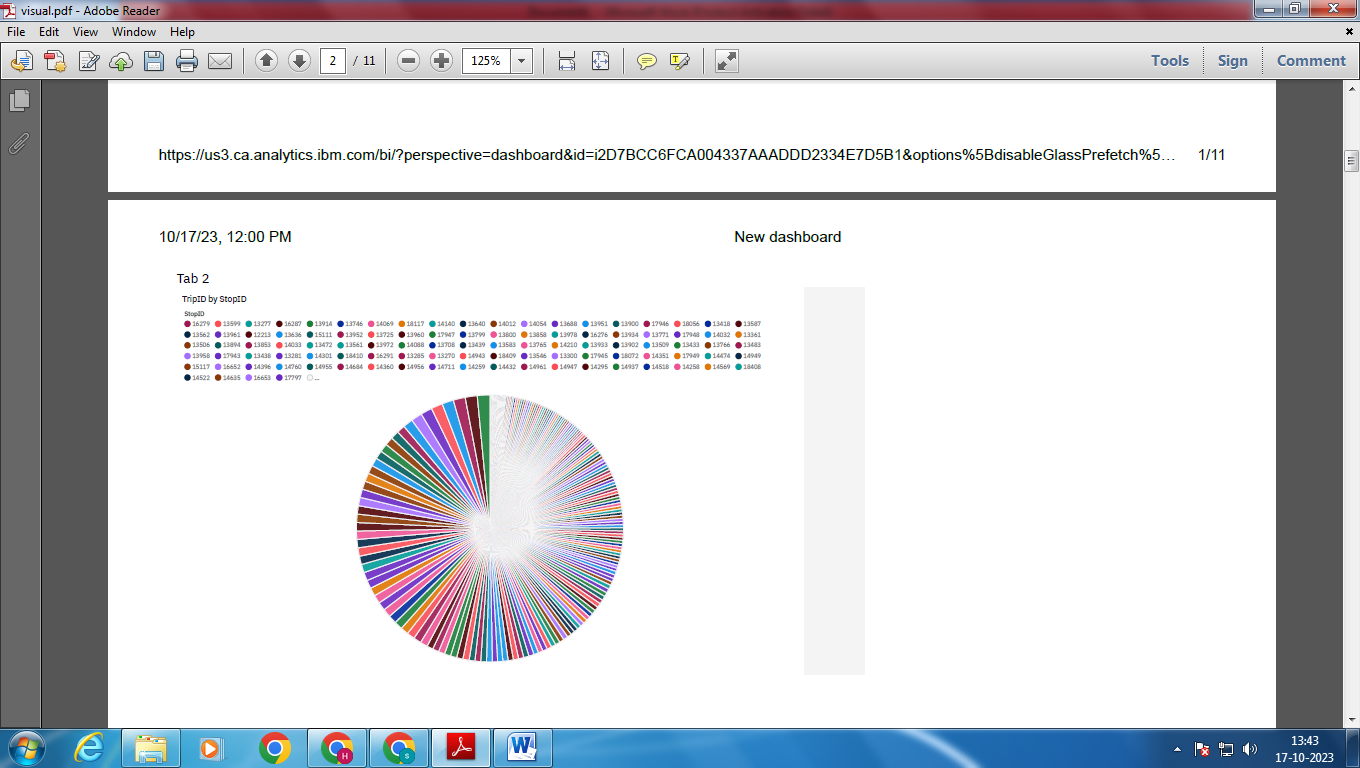
scaler=StandardScaler()

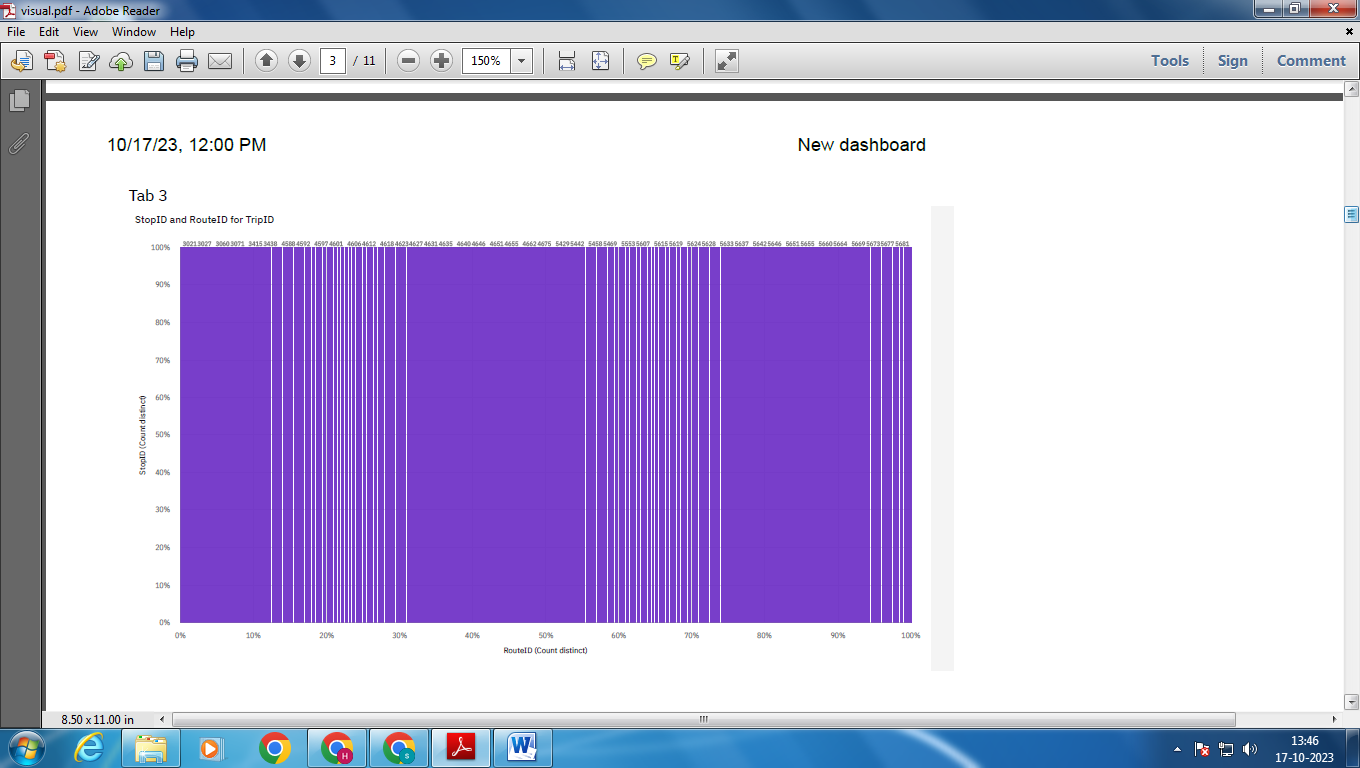
data[['StopID']] = scaler.fit\_transform(data[['StopID']])data.to\_csv('cleaned\_data.csv',index=False)

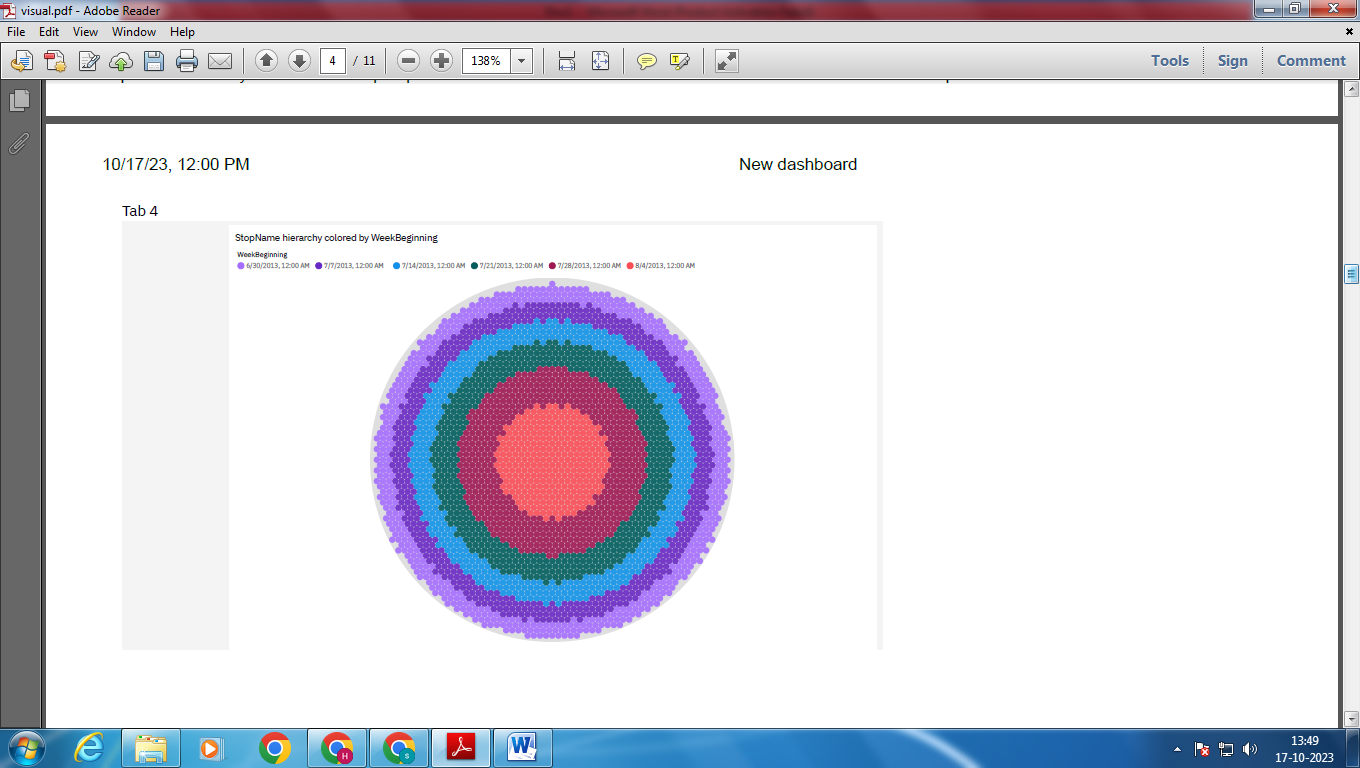
from google.colab import drivedrive.mount('/content/drive')

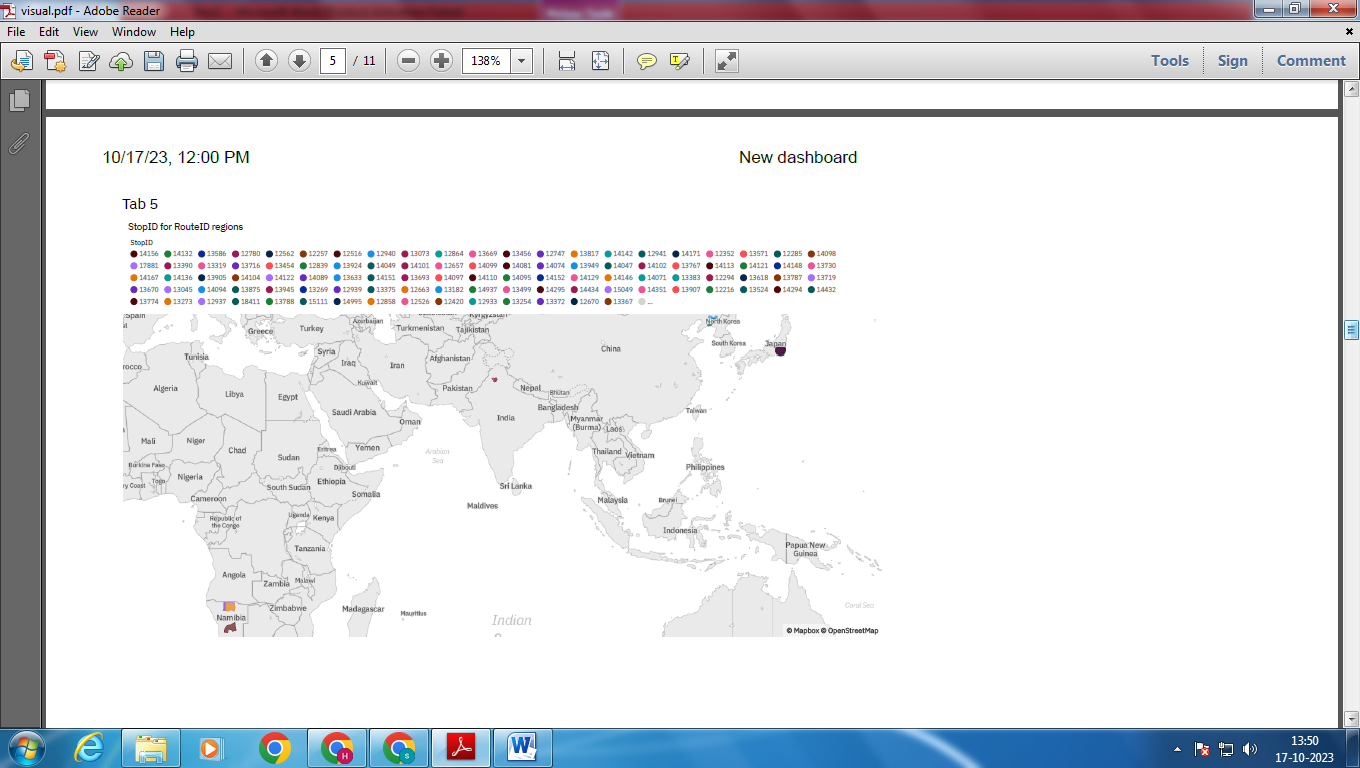
Mounted at/content/drive

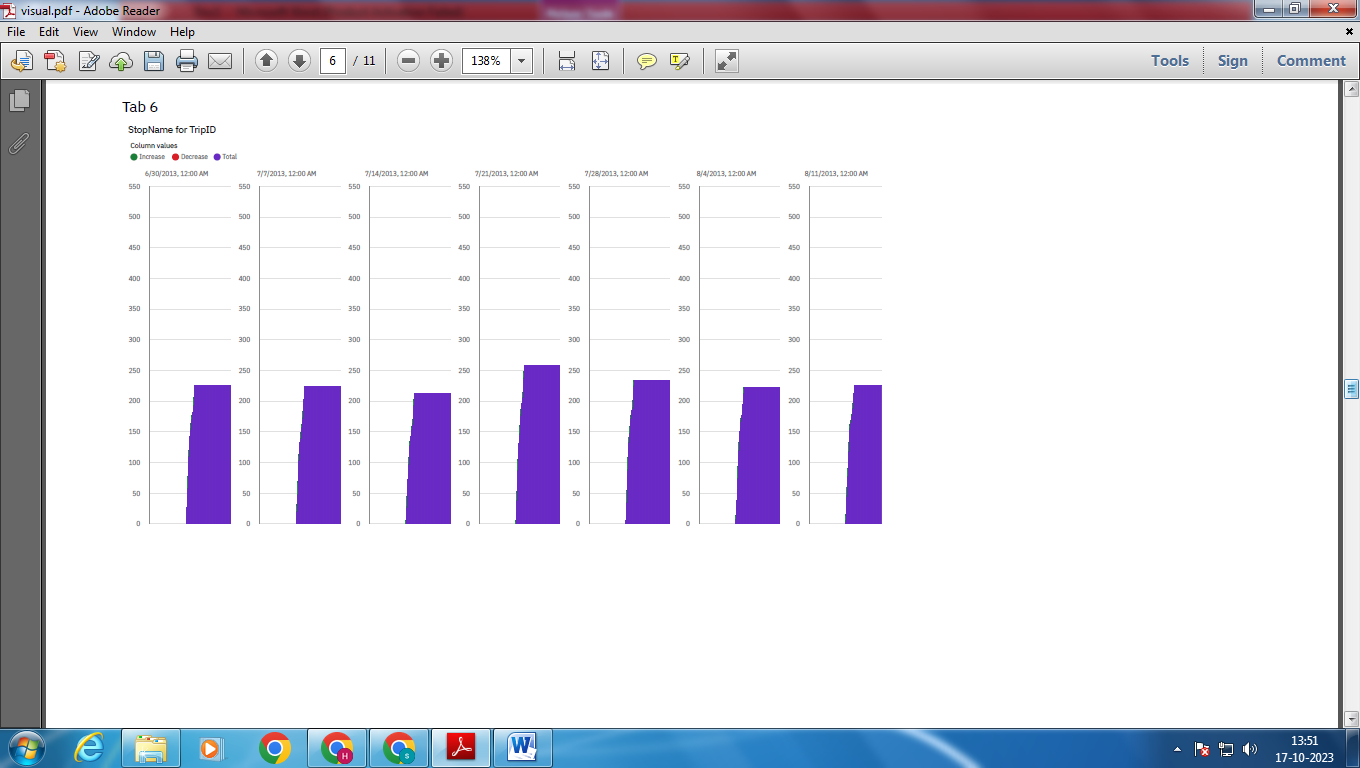


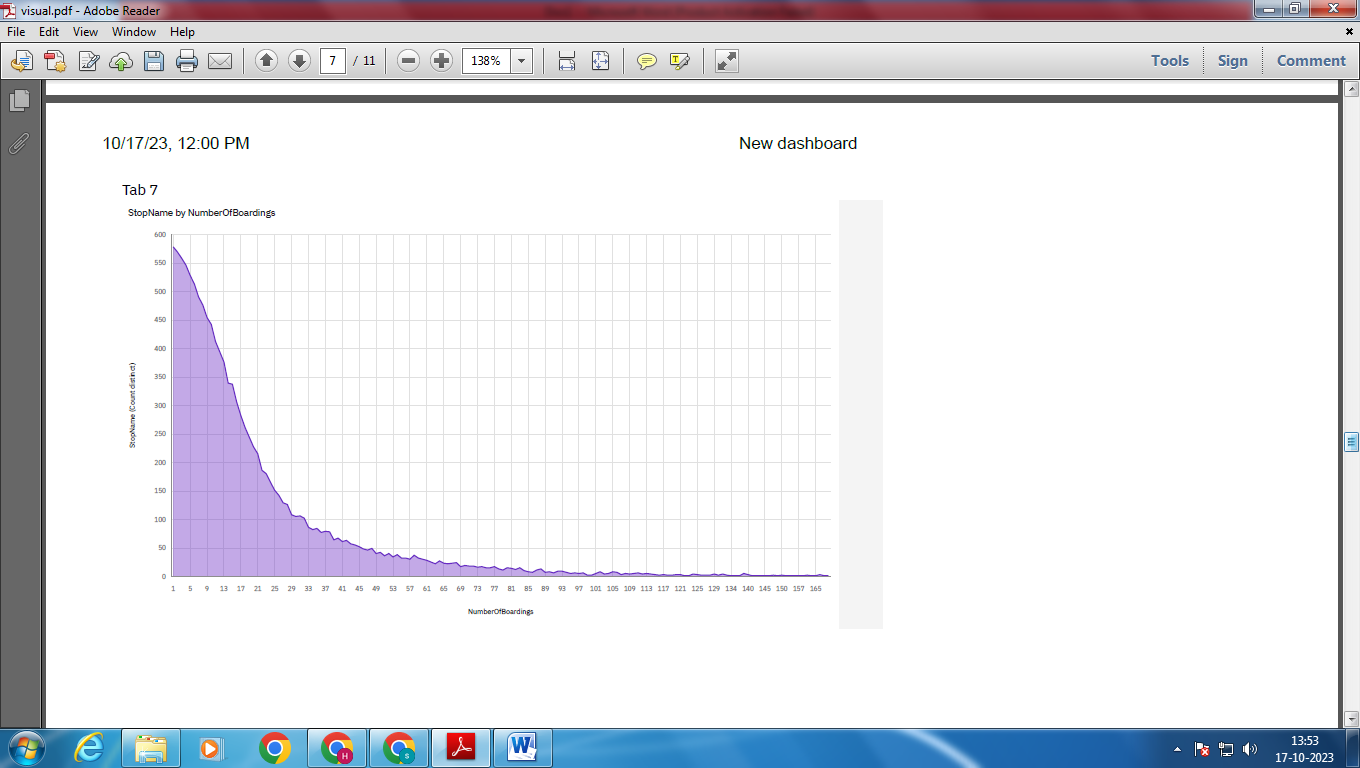


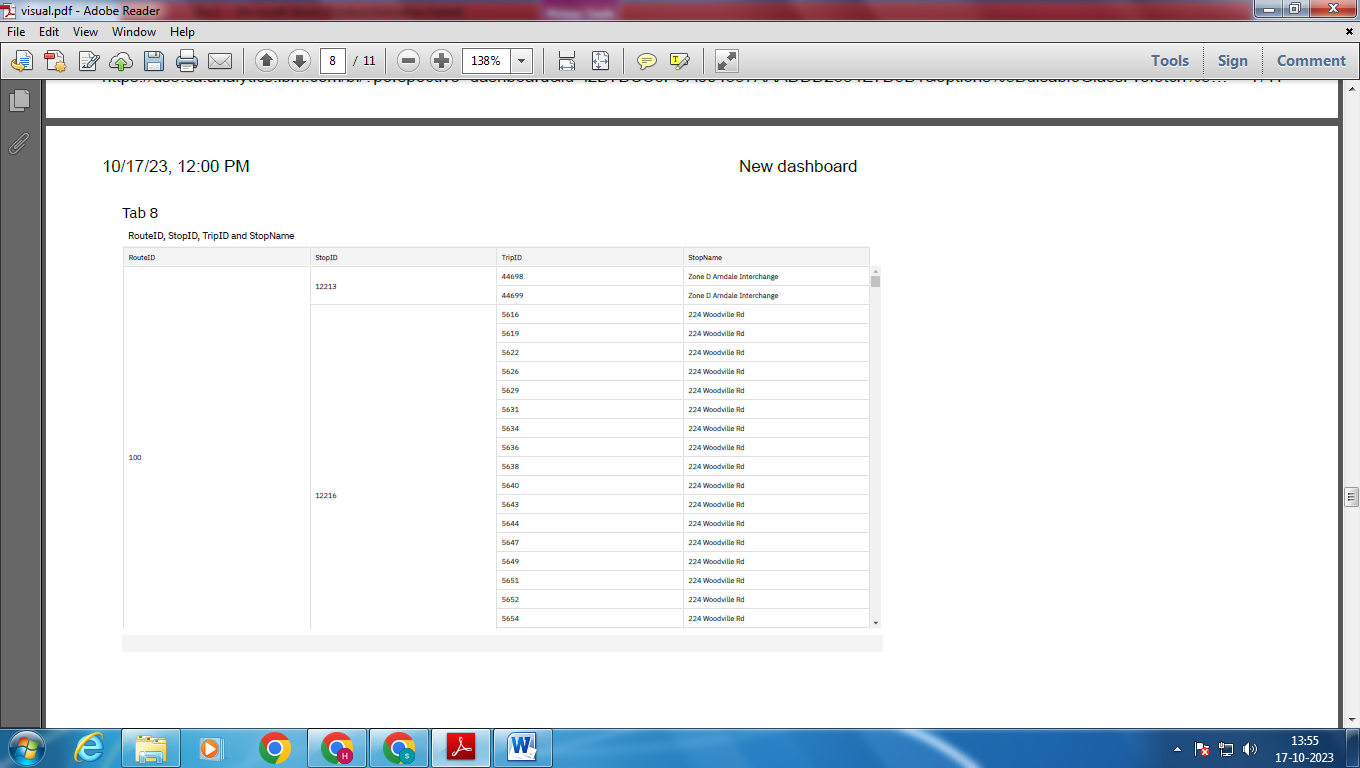


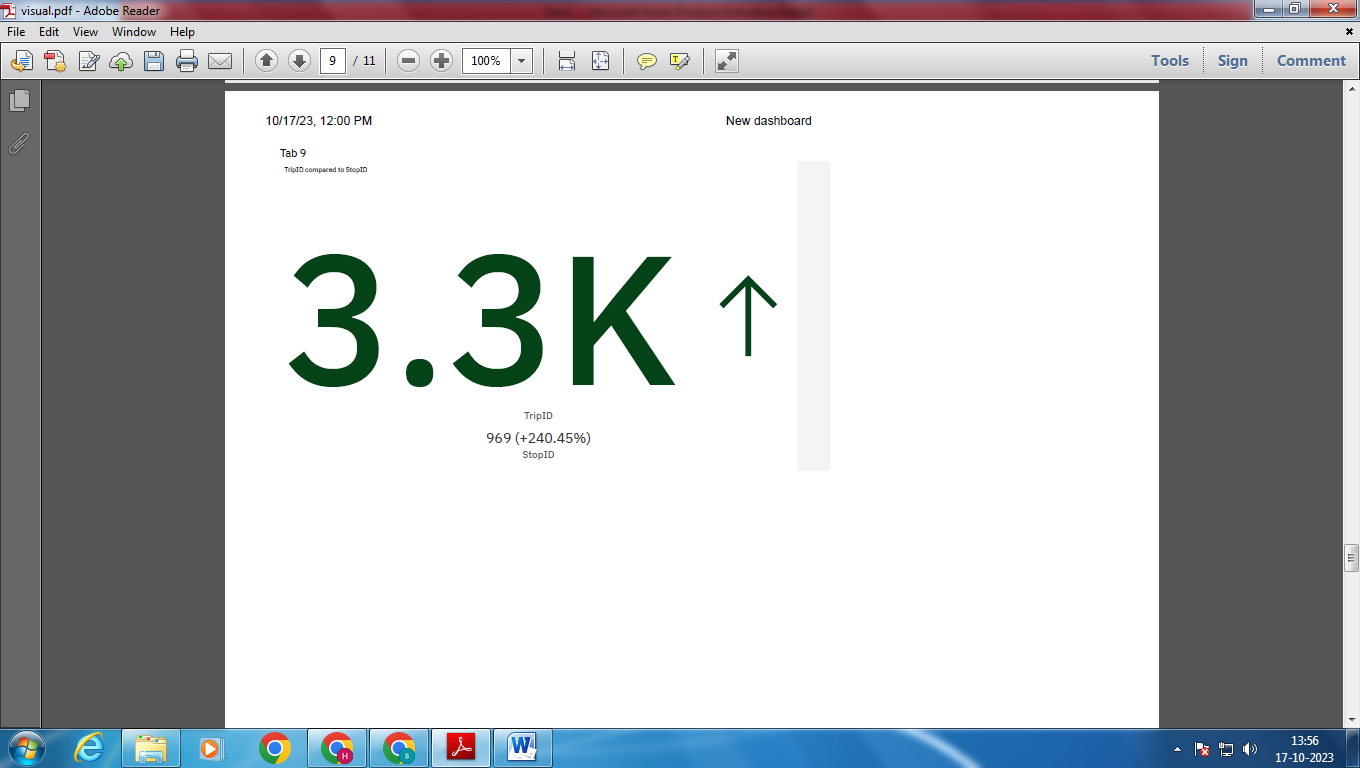


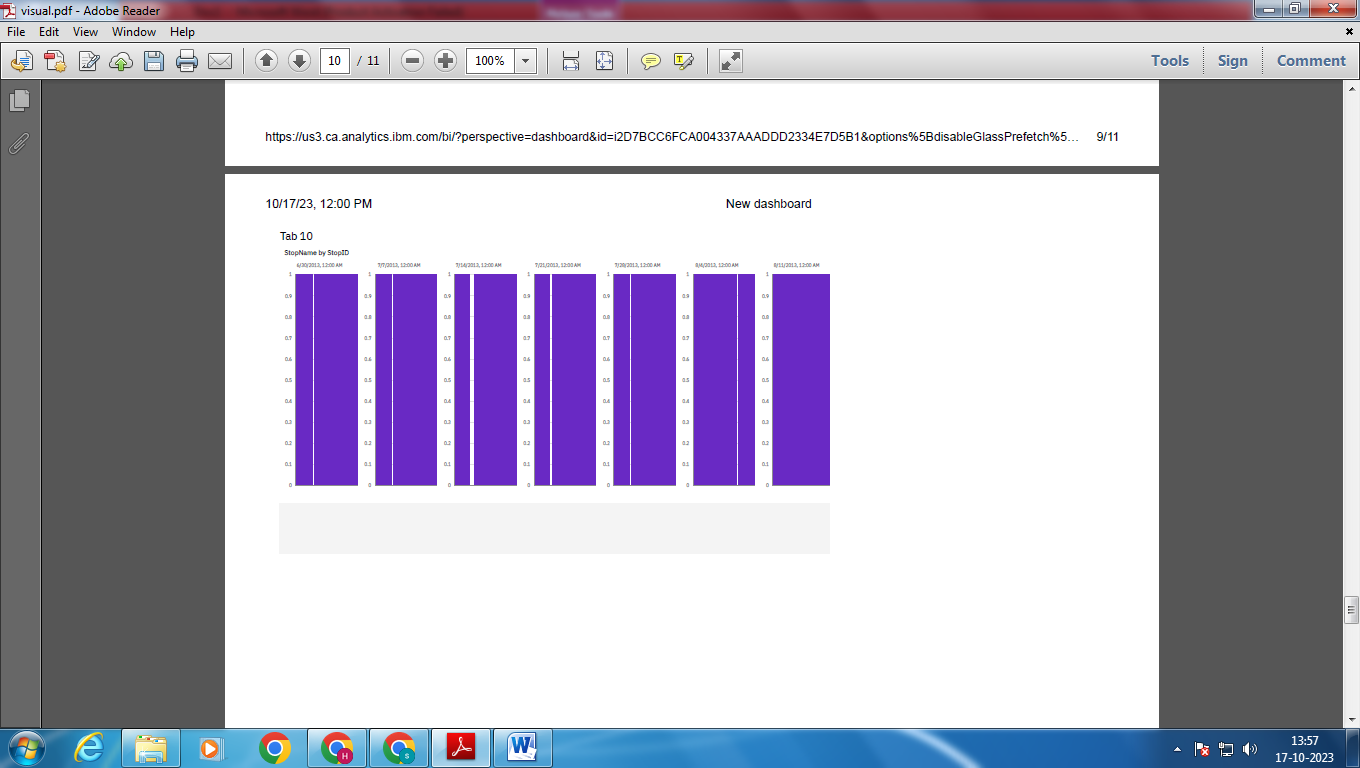












**CODE FOR ANALYSIS OF DATA:**

%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('/content/cleaned\_data.csv')

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

data= pd.merge(data,out\_geo,how='left',left\_on ='StopName',right\_on ='input\_string')  
data.head(5)  
data.shape

(20431, 16)

*#Columns to keep for further analysis*  
col = ['TripID', 'RouteID', 'StopID', 'StopName', 'WeekBeginning','NumberOfBoardings',  
'latitude', 'longitude','postcode','type']  
data = data[col]

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

*# 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()]

<ipython-input-9-ac1d5add8db2>:3: FutureWarning: Index.ravel returning ndarray is deprecated; in a future version this will return a view on self.  
 grouped.columns = ["\_".join(x) for x in grouped.columns.ravel()]

grouped.head(10)  
grouped.columns

Index(['NumberOfBoardings\_sum', 'NumberOfBoardings\_count',  
 'NumberOfBoardings\_max'],  
 dtype='object')

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

StopName WeekBeginning type NumberOfBoardings\_sum \  
0 10 Holbrooks Rd 07-07-2013 street\_address 73.0   
1 10 Holbrooks Rd 08-04-2013 street\_address 245.0   
2 10 Holbrooks Rd 08-11-2013 street\_address 223.0   
3 10 Holbrooks Rd 09-01-2013 street\_address 194.0   
4 10 Holbrooks Rd 09-08-2013 street\_address 198.0   
  
 NumberOfBoardings\_count NumberOfBoardings\_max   
0 23 9.0   
1 26 100.0   
2 26 89.0   
3 25 73.0   
4 26 79.0

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

StopName WeekBeginning  
0 10 Holbrooks Rd 12  
1 10 Marion Rd 12  
2 10A Marion Rd 12  
3 11 Marion Rd 12  
4 11 Portrush Rd 12

*#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]

[]

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

(0, 6)

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)

data without stopage removing: (20431, 10)  
data, after removing stoppage not having the data of whole 54 weeks: (0, 10)

new\_data.head(2)  
filtered\_data = new\_data[new\_data['latitude'] <=100]  
filtered\_data.shape

(0, 10)

data = filtered\_data.copy()  
data.shape

(0, 10)

*#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())

stopageName\_with\_boarding.columns = ["StopName", "Total\_boarding\_on\_the\_stopage"]  
*#stopageName\_with\_boarding.shape*  
stopageName\_with\_boarding.head()

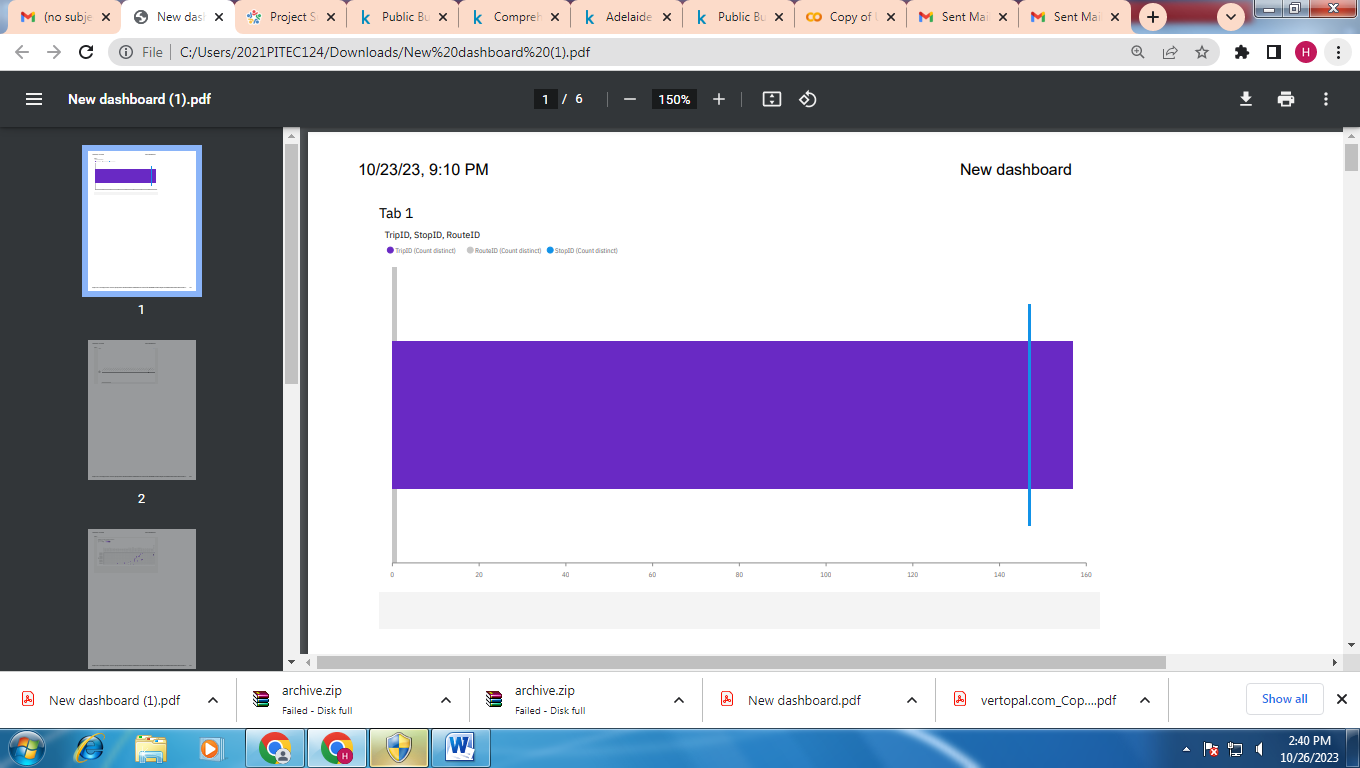
Empty DataFrame  
Columns: [StopName, Total\_boarding\_on\_the\_stopage]  
Index: []

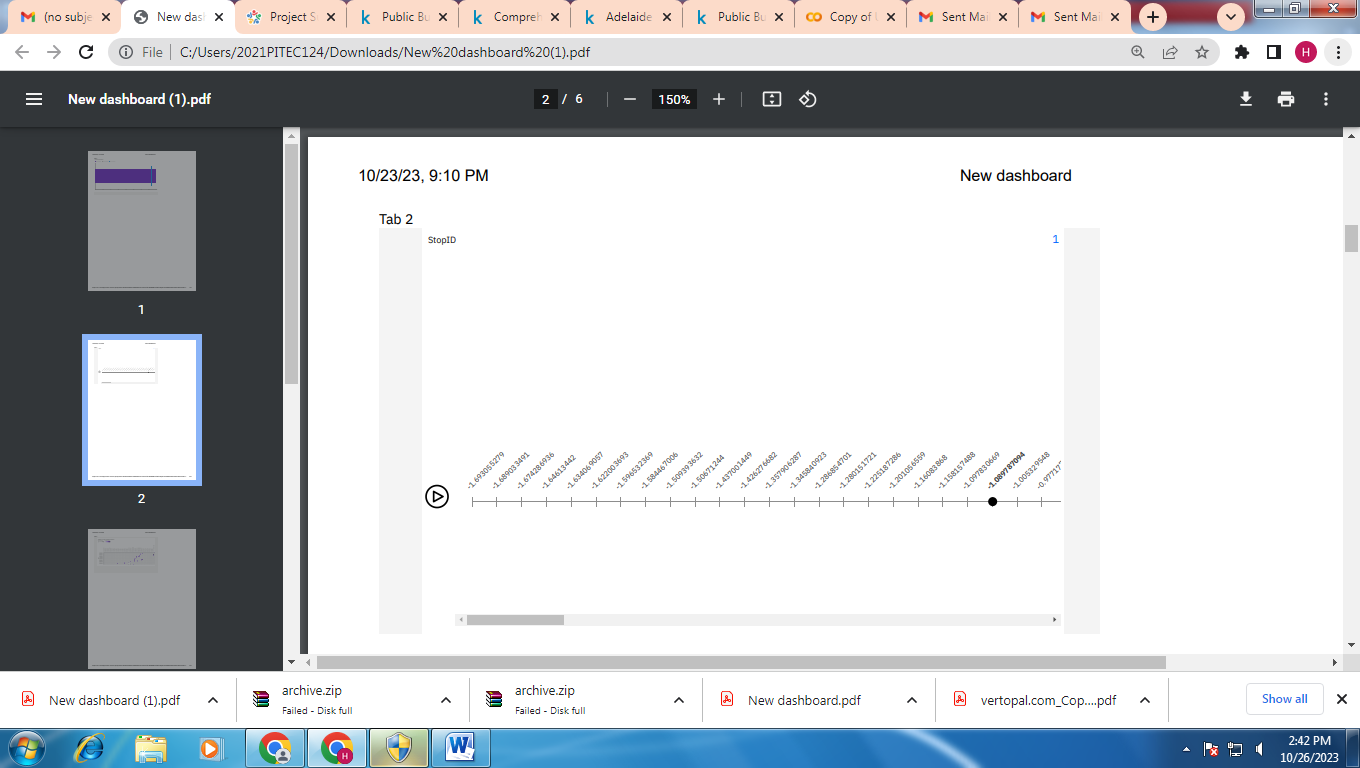
*## save the aggregate data*  
*#bb.to\_csv('st\_week\_grp.csv', index=False)*

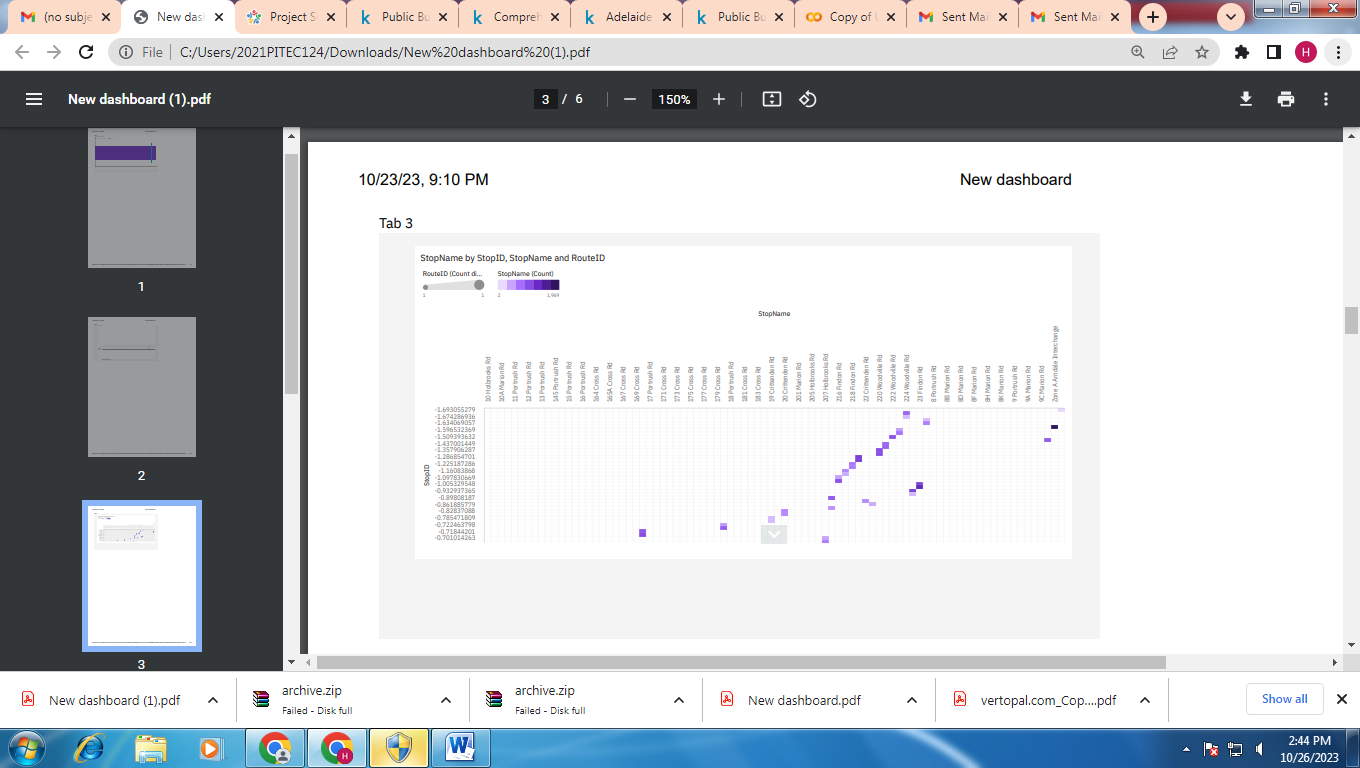
data.nunique()  
*#data.isnull().sum()*  
*#data['WeekBeginning'].unique()*

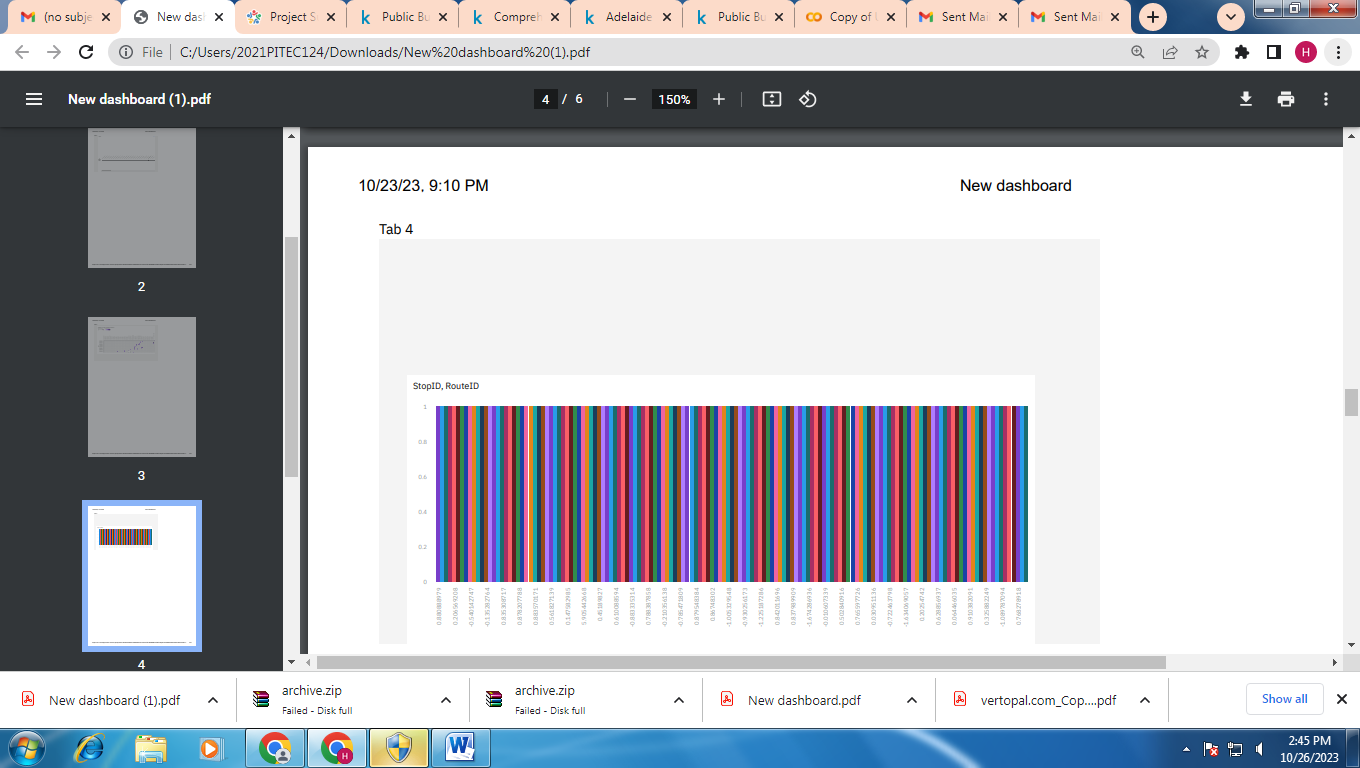
TripID 0  
RouteID 0  
StopID 0  
StopName 0  
WeekBeginning 0  
NumberOfBoardings 0  
latitude 0  
longitude 0  
postcode 0  
type 0  
dtype: int64

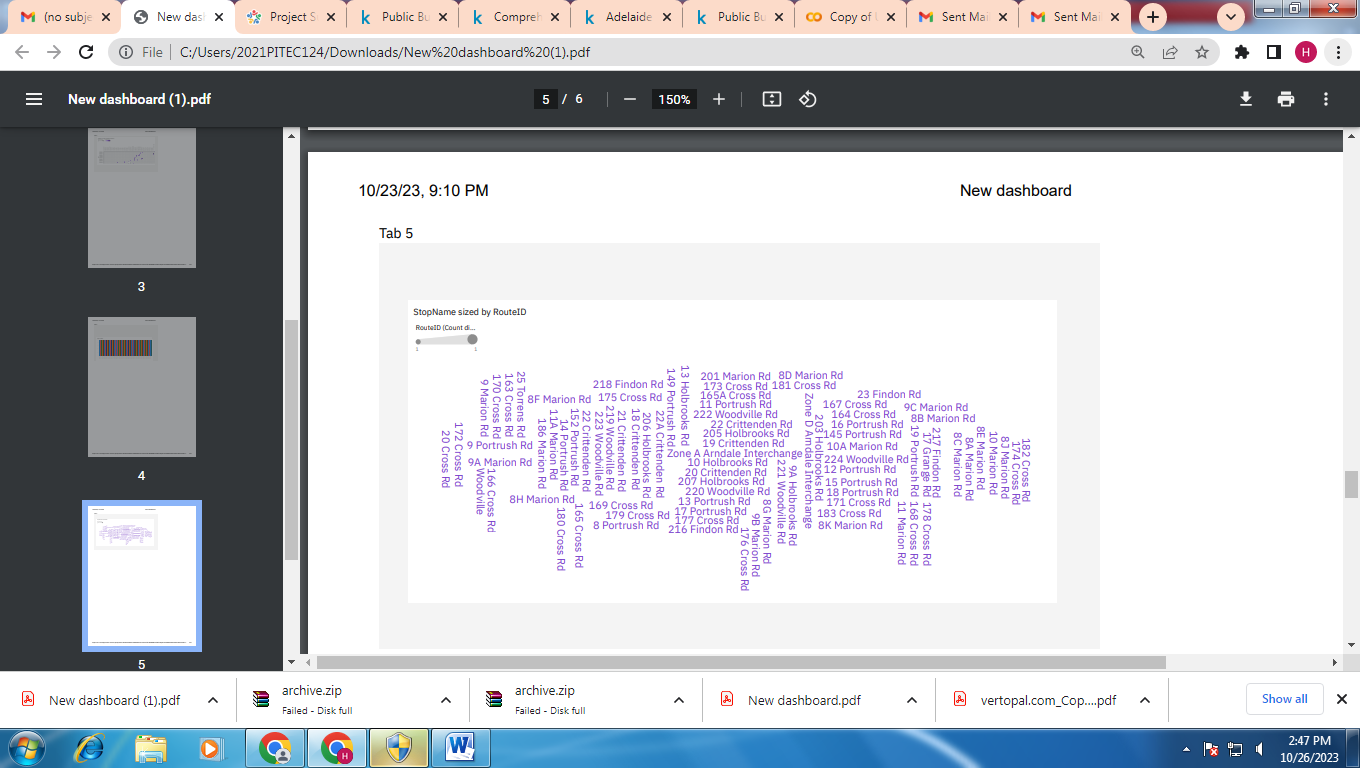
DATA VISUALISATION USING IBM COGNOS:











IMPROVEMENT INITIATIVES:

1. Perform data analysis:

Make use of Cognos' sophisticated analytical features to carry out in-depth study. Utilize data mining strategies, statistical approaches, and predictive analytics to find patterns and trends in transportation data.

2. Determine Areas for Improvement:

Examine the findings to see where the transportation system needs to be improved, such as any bottlenecks or underutilized routes.

3. Implement optimization strategies:

Formulate and execute optimization tactics by drawing on the understandings obtained from the data examination. This can entail changing the schedules of public transportation, rerouting traffic, or timing traffic signals optimally.

4. Observe and Assess:

Use real-time data to continuously assess the performance of the transportation system. Analyze the success of the tactics that have been put into practice and make any necessary changes for ongoing progress.

5. Share Research Results:

Share the conclusions and revelations from the data analysis with the appropriate parties, including the city planners, transit authorities.

CONCLUSION:

To sum up, utilizing IBM Cognos data analytic tools to assess the effectiveness of public transit is a big step in the direction of creating more intelligent, adaptable, and sustainable urban mobility networks. We have gotten essential insights into the complex patterns of passenger behavior, vehicle movement, and system performance through the methodical gathering, integration, and analysis of different transportation data.