

**ANL252**

**Python for Data Analytics**

**Group-Based Assignment**

**July 2022 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Alexandre Oudea** | **K1882534** |
| **Vivian Wee Si Min** | **E2010380** |
| **Van Lal Rem** | **Z2172902** |
| **Lim Shu Han** | **B2082598** |

**Tutorial Group: ­­­­­­­­­­­­ T 09**

**Instructor’s Name: Prof. Munish Kumar**

**Submission Date: 28/08/2022**

**Question 1(a)**

To replace the missing values “?”, “--” and “-“ into the default missing value marker NaN for readability and convenience, the dataset was read as:

GBA\_data = pd.read\_csv(' GBA\_data',na\_values = ['?','-','--'])

**Question 1(b)**

The sum of NaN values for each variable was then shown using the .isnull() and .sum() functions to reveal a count of 504 NaN values for **origin** and **destination** each, 10 for **type**, **yob** and **age**, and 3 for **gender**.

As NaN values cannot be casted to change type, we treated them by replacing them with either the mode or mean for their respective variable. Variables such as type and gender are nominal and non-numeric, thus the only possible replacement is with their modes. While origin and destination are numeric, they are also at the nominal level of measurement, thus their modes are most appropriate for replacement as the means and medians are insignificant. Finally, for ratio variables yob and age, we decided to replace NaN values with their means to preserve the average and not skew their distributions, which could impact future analysis. This was achieved via the following:

GBA\_data['origin'] = GBA\_data['origin'].fillna(GBA\_data['origin'].mode()[0])

GBA\_data['destination'] = GBA\_data['destination'].fillna(GBA\_data['destination'].mode()[0])

GBA\_data['type'] = GBA\_data['type'].fillna(GBA\_data['type'].mode()[0])

GBA\_data['gender'] = GBA\_data['gender'].fillna(GBA\_data['gender'].mode()[0])

yob\_mean = GBA\_data['yob'].mean()

age\_mean = GBA\_data['age'].mean()

GBA\_data['yob'].fillna(value = yob\_mean, inplace = True)

GBA\_data['age'].fillna(value = age\_mean, inplace = True)

A final check on whether all NaN values were successfully replaced revealed no remaining missing values in the dataframe.

**Question 1(c)**

**Different Data Types**

Using df.info, it was found that column indexes 0, 1, 7 and 8 were floats, while indexes 2, 3, 5, 6 and 9 were objects. Only index 4 (id) contained integers. As such, the .astype function was used to cast the respective columns to the correct data type for accurate classification and further analysis, as follows:

GBA\_data['origin'] = GBA\_data['origin'].astype(int,copy=True)

GBA\_data['destination'] = GBA\_data['destination'].astype(int,copy=True)

GBA\_data['start'] = pd.to\_datetime(GBA\_data['start'])

GBA\_data['end'] = pd.to\_datetime(GBA\_data['end'])

GBA\_data['id'] = GBA\_data['id'].astype(str,copy=True)

GBA\_data['type'] = GBA\_data['type'].astype(str,copy=True)

GBA\_data['subscriber'] = GBA\_data['subscriber'].astype(str,copy=True)

GBA\_data['yob'] = GBA\_data['yob'].astype(int,copy=True)

GBA\_data['age'] = GBA\_data['age'].astype(int,copy=True)

GBA\_data['gender'] = GBA\_data['gender'].astype(str,copy=True)

A “duration” column was also introduced to measure the duration of each commuter’s trip in minutes. This column was then cast as integers for further analysis:

GBA\_data['duration'] = GBA\_data['end'] - GBA\_data['start']

GBA\_data['duration'] = GBA\_data['duration'].dt.total\_seconds().div(60).astype(int)

**Removing Duplicate Rows and Rows with Date after August 2022**

After replacing the missing values in 1(b) with their respective modes and means, a check for duplicate rows was conducted using the following code:

GBA\_data = GBA\_data.drop\_duplicates(keep=False)

print(GBA\_data.duplicated().sum())

An output of 0 was obtained, indicating no duplicate rows in the dataframe after replacement of missing values.

A filter on Excel for start and end times also indicated timestamps with value dates after August 2022, such as in December 2022 and January 2023. These rows were removed using the code below to preserve the integrity of the dataset, with 27 August 2022 as the reference date for removal.

GBA\_data['start'] = pd.to\_datetime(GBA\_data['start'])

GBA\_data = GBA\_data[~(GBA\_data['start'] > '2022-08-27')]

**Outliers**

Apart from missing values, outliers are also present in the data. Plotly was used to create box plots for the variables age and duration, revealing the presence of outliers for both variables, as seen in Figures 1 and 2 below.

**Figure 1**

*Box plot for age variable*

*Chart, timeline, bar chart

Description automatically generated*

**Figure 2**

*Box plot for duration variable*

Graphical user interface, application, Teams

Description automatically generated

There exists some outliers for age and duration above the upper fences of 71 and 25 respectively. The presence of extreme outliers is apparent in Figure 2. Rows containing these outliers need to be removed in order not to skew the results of analysis and also because they could indicate, especially in the case for duration, measurement errors or issues with data entry. As such, outliers were removed for both age and duration using the interquartile range method in the below code:

def outliers(df, col):

Q1 = df[col].quantile(0.25)

Q3 = df[col].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

ls = df.index[(df[col] < lower\_bound) | (df[col] > upper\_bound)]

index\_list = []

for column in ['duration','age']:

index\_list.extend(outliers(GBA\_data,column))

def remove(df,ls):

ls = sorted(set(ls))

df = df.drop(ls)

return df

GBA\_cleaned = remove(GBA\_data, index\_list)

A final check was conducted using GBA\_cleaned.describe() to assess the effects of the outlier removal process and whether the cleaned data is satisfactory.

**Question 1(d)**

GBA\_cleaned['hour\_stamp'] = GBA\_cleaned['start'].dt.hour

ComFreq\_hr = GBA\_cleaned.groupby(['hour\_stamp'])[['id']].count().reset\_index()

ComFreq\_hr.rename(columns={'id':'count'},inplace=True)

ComFreq\_hr['count'].sum()

def get\_time():

GBA\_cleaned['hour\_stamp'] = GBA\_cleaned['start'].dt.hour

ComFreq\_hr = GBA\_cleaned.groupby(['hour\_stamp'])[['id']].count().reset\_index()

ComFreq\_hr.rename(columns={'id':'count'},inplace=True)

ComFreq\_hr.insert(1,'12H format', ['12AM','1AM','2AM','3AM','4AM','5AM','6AM',

'7AM','8AM','9AM','10AM','11AM','12PM','1PM',

'2PM','3PM','4PM','5PM','6PM','7PM','8PM','9PM',

'10PM','11PM'])

ComFreq\_hr = ComFreq\_hr[['12H format','count']]

results = ComFreq\_hr[ComFreq\_hr['count']==ComFreq\_hr['count'].max()]

return results.iloc[0,0]

print('The highest number of commuters start their journey at', get\_time())

**Question 1(e)**

**Insight 1**

A clustered bar chart/histogram was created to analyse the ages of users clustered by their profile types. As there were many instances of duplicate IDs in the dataset, due to some users taking multiple trips at different times, instances of duplicate IDs were first removed to keep only 1 record for each individual user. A histogram was then plotted with age as the x variable, coloured by type, as seen in Figure 3 below.

Code used:

GBA\_Unique\_ID = GBA\_data.drop\_duplicates(subset="id")

CHART\_1 = px.histogram(GBA\_Unique\_ID, x='age',

color='type', barmode='group', title='1 Frequency by Age of consumers broken down by Type',

height=400)

CHART\_1.show()

**Figure 3**

Clustered bar chart showing frequency of age clustered by type

A screenshot of a computer

Description automatically generated with low confidence

It is observed that the concessionary age range is from 55 to 65. As there exists a fair proportion of regular users aged 50 to 54 as well, the shared mobility service could consider extending concessionary pass benefits to users aged above 50 to attract more concession sign ups from this age group to possibly increase total revenue. Additionally, the firm can also focus marketing efforts on the demographic age group of 30 to 40, as this group constitutes the majority of both ad-hoc and regular users. Should the service be able to successfully convert more ad-hoc users to regular users, it should also experience an increase in revenue.

**Insight 2**

A scatter plot with the average duration of trips as the Y-axis and the start day of the week as the X-axis, differentiated by user profile types and the average age of users, to visualise peak and lull periods of service usage. This was achieved using the following code and resulted in the scatter plot output in Figure 4 and Figure 5 below. The 2 figures are of the same output graph, except with different views due to the interactive selection of user types.

Code used:

GBA\_cleaned['start'] = pd.to\_datetime(GBA\_cleaned['start'])

GBA\_cleaned['start\_date'] = GBA\_cleaned['start'].dt.date

GBA\_cleaned['Day of the week'] = GBA\_cleaned['start'].dt.day\_name()

GBA\_cleaned=GBA\_cleaned.join(GBA\_cleaned.groupby(['Day of the week','type'])['age'].mean(),

on=['Day of the week','type'], rsuffix='\_avg')

GBA\_cleaned=GBA\_cleaned.join(GBA\_cleaned.groupby(['Day of the week','type'])['duration'].mean(),

on=['Day of the week','type'], rsuffix='\_avg')

GBA\_cleaned = GBA\_cleaned.loc[:,~GBA\_cleaned.columns.duplicated()].copy()

GBA\_cleaned = GBA\_cleaned.sort\_values(by=['start\_date'],ascending=True)

CHART\_2 = px.scatter(GBA\_cleaned,

x='Day of the week',

y='duration\_avg',

color='type',

size='age\_avg',

title= '2 Average Duration by day of the week - Bubble size based on average age')

CHART\_2

**Figure 4**

*Scatter plot showing start day of the week by average duration of trips (all user types)*

Chart, scatter chart

Description automatically generated

Figure 5 below provides a better zoomed-in view of the user types “Regular” and “Concession”.

**Figure 5**

*Scatter plot showing start day of the week by average duration of trips (“Regular” and “Concession” only)*

Chart, scatter chart

Description automatically generated

As seen in Figure 4, trips taken on Thursdays and Fridays had the longest average durations for all user types, indicating them as the peak period of usage. For ad-hoc users, while trips taken on Saturdays were relatively long as well, trips taken from Sundays to Wednesdays were, on average, the shortest. As seen in figure 5, trips for concession users were relatively short on Tuesdays and Saturdays and trips for regular customers were similarly short throughout Saturdays to Wednesdays. Additionally, based on the size of the bubbles, albeit the visual lack of difference in average age across days, there was a visible difference across user types, with the concession users having the largest average age. This reaffirms our earlier observations in Insight 1.

Assuming that all users pay for the shared mobility service based on the duration of their trips (e.g. pay-per-minute pricing schemes), this presents marketing opportunities for the firm to ramp up off-peak promotions for its service on Saturdays to Wednesdays, especially on Tuesdays, Saturdays (for regular customers) and Sundays (ad-hoc customers). Given the above assumption, successfully increasing the duration of trips on off-peak periods would increase the firm’s revenue as well.

**Insight 3**

2 stacked bar graphs were created to identify the 5 most popular start and end location identifiers respectively, stacked by user profile type. The 2 output graphs are depicted in Figures 5 and 6 below.

Code for 1st graph (5 most popular origins):

x = pd.DataFrame(GBA\_cleaned.groupby(['origin', 'type']).size())

x.columns = ['counts']

x=x.reset\_index()

a=x.set\_index('origin')

b = pd.DataFrame(GBA\_cleaned.groupby(['origin']).size())

c = b.sort\_values(by=0, ascending=False).head().index

d = a.loc[c,:].reset\_index()

CHART\_3A = px.bar(d,x='origin',y='counts', color='type', text\_auto=True,

title = "3A Most popular Origin against Subscription Type")

CHART\_3A.update\_traces(textposition='inside')

CHART\_3A.update\_layout(uniformtext\_minsize=12, uniformtext\_mode='show')

CHART\_3A.update\_layout(xaxis={'tickmode':'array','tickvals':c,'ticktext':c})

CHART\_3A.update\_xaxes(

showgrid=True,

ticks="outside",

tickson="boundaries",

ticklen=20)

CHART\_3A.show()

Code for 2nd graph (5 most popular destinations):

x = pd.DataFrame(GBA\_cleaned.groupby(['destination', 'type']).size())

x.columns = ['counts']

x=x.reset\_index()

a=x.set\_index('destination')

b = pd.DataFrame(GBA\_cleaned.groupby(['destination']).size())

c = b.sort\_values(by=0, ascending=False).head().index

d = a.loc[c,:].reset\_index()

CHART\_3B = px.bar(d,x='destination',y='counts', color='type', text\_auto=True,

title = "3B Most popular Destination against Subscription Type")

CHART\_3B.update\_traces(textposition='inside')

CHART\_3B.update\_layout(uniformtext\_minsize=12, uniformtext\_mode='show')

CHART\_3B.update\_layout(xaxis={'tickmode':'array','tickvals':c,'ticktext':c})

CHART\_3B.update\_xaxes(

showgrid=True,

ticks="outside",

tickson="boundaries",

ticklen=20)

CHART\_3B.show()

**Figure 5**

*Stacked bar graph showing frequencies of top 5 popular origins (by user type)*

Chart, bar chart

Description automatically generated

**Figure 6**

*Stacked bar graph showing frequencies of top 5 popular destinations (by user type)*

Chart, bar chart

Description automatically generated

In Figure 5, locations 30, 58, 81, 67 and 15 were identified as the most common origin points, in ranked order. In Figure 6, locations 30, 67, 15, 21 and 81 were identified as the most popular destinations, in ranked order. With this information in mind, the firm should prioritize placing enough mobility vehicles at the popular start locations to ensure that users are able to find available vehicles for use at these origin points. The firm should also ensure ample parking facilities and space at the most popular destinations to allow users to have a smooth experience ending their journeys, while being socially responsible to avoid situations such as crowding of vehicles blocking public areas.

These measures should be prioritized at origin points 13, 58 and 31, as well as destinations 30 and 67, which are the most popular locations for the service’s regular customers, to ensure continuous customer satisfaction and loyalty, as unpleasant experiences would encourage regulars to switch to other competitors, resulting in a large loss of revenue due to the high lifetime value of regular and loyal customers.