****

**ANL252**

**PYTHON FOR DATA ANALYTICS**

**Group-based Assignment**

# **July 2022 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Toh Li Ting** | **M2172239** |
| **Jeffrey Lim Dao Fong** | **Y2111639** |
| **Kris Lennings** | **J1910196** |

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

**Declaration Page**

**We, members of Group 7, do hereby declare that we each contributed to this assignment and that we collectively agree to a shared grade.**

|  |  |  |
| --- | --- | --- |
| **Name** | **Contribution** | **Signature** |
| **Toh Li Ting (Team Lead)** | **All questions are attempted and completed as a group.** |  |
| **Jeffrey Lim Dao Fong** | **All questions are attempted and completed as a group.** |  |
| **Kris Lennings** | **All questions are attempted and completed as a group.** |  |

**Qn1a**

**i.**

First, we must import the packages and modules listed below into the Jupyter environment.

*# import pandas as pd: This module provides users to access the pandas module functions that enables them to read a csv or xlxs data file.*

*# import numpy as np: This module allows users to work with the high multidimensional array and matrices contained in the data file.*

*# from matplotlib import pyplot as plt. This module allows users to create figures, plots and labels in the Jupyter environment*

*# import seaborn as sns. This module allows users to visualise data by creating graphs and plots.*

*import pandas as pd*

*import numpy as np*

*from matplotlib import pyplot as plt*

*import seaborn as sns*

**ii.**

Next, we'll load the GBA Data as a data frame (df) and examine its structure. Before that, we must create a list of missing values in Jupyter as specified in the assignment. These will allow Python to detect the other types of missing values in the GBA dataset.

*# Make a list of missing value types i.e "-", "--", "?" and convert it to NaN.*

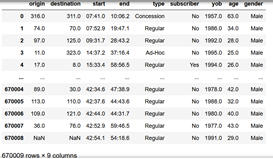
*missing\_values = ["-", "--", "?"]*

*df = pd.read\_csv('D:\GBA\_Data.csv', na\_values = missing\_values)*

*# Drop the ID column*

*df.drop(df.columns[4], axis=1, inplace=True)*

*df*



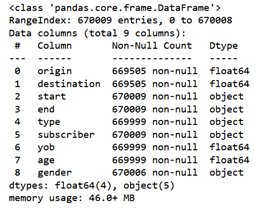
*Table 1. Table output with missing data (NaN)*

**Qn1b**

**i.**

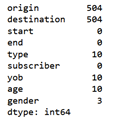
Following that, we must investigate the data types and look for any missing data in the data frame (df). This is to assist users in comprehending the dataset's data types and objects. We can see from the output that there are 504 missing values for the 'origin' and 'destination' columns, 10 missing values for the 'type', 'yob', and 'age' columns, and 3 missing values for the 'gender' column.

*df.info()*



*Figure 1. Output obtained for df.info() code*

*df.isna().sum()*

**

*Figure 2. Output obtained for df.isna().sum() code*

**ii.**

By accessing the category counts contained in the 'type', 'yob' and 'gender' columns, we can replace the missing values by using the 'Most Frequent' variable count.

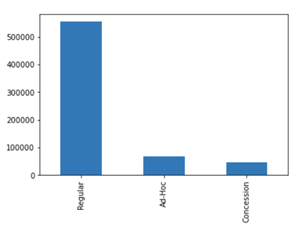
Replacing the missing values in the 'type' column

*df['type'].value\_counts()*

**

*Figure 3. Frequency count of each category item in the type column*

*df['type'].value\_counts().plot.bar()*

**

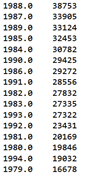
*Figure 4. Bar chart for each category item in the type column*

*df['type'].fillna(df['type'].value\_counts().index[0]*

*,inplace = True)*

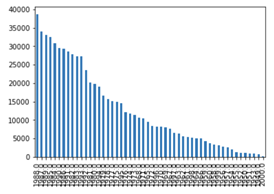
Replacing the 10 missing values in the 'yob' column

*df['yob'].value\_counts()*



*Figure 5. Frequency count of each category item in the yob column*

*df['yob'].value\_counts().plot.bar()*

**

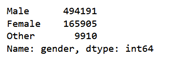
*Figure 6. Bar chart for each category item in the yob column*

*df['yob'].fillna(df['yob'].value\_counts().index[0]*

*,inplace = True)*

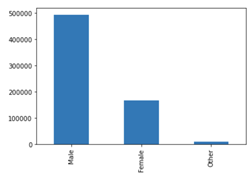
Replacing the 3 missing values in the 'gender' column. Other is considered as Transexual identity.

*df['gender'].value\_counts()*

**

*Figure 7. Frequency count of each category item in the gender column*

*df['gender'].value\_counts().plot.bar()*

**

*Figure 8. Bar chart for each category item in the gender column*

*df['gender'].fillna(df['gender'].value\_counts().index[0]*

*,inplace = True)*

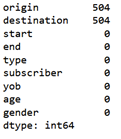
Replacing the 10 missing values in the 'age' column. Because the 'age' column contains continuous data, we can use the 'Mean' to replace these values.

*x = df["age"].mean()*

*df["age"].fillna(x, inplace = True)*

In this stage, we have successfully replaced the missing values "-," "--," and "?" stated in the assignment.

*df.isna().sum()*

**

*Figure 9. Output obtained for df.isna().sum() code*

**Qn1c**

**i.**

The first data quality issue is the handling of 'Blanks' in the 'origin' and 'destination' columns. We should not discard or delete these records because they account for only 0.07522% (<5% of the total dataset). As a result, we can replace these 'Blanks' with the values 'Mean' or 'Median' depending on the skewness values obtained from each of the data categories in the 'type' column.

*# from scipy.stats import skew: This module allows users to work on various probability distributions and summary statistics contained in the dataset.*

*from scipy.stats import skew*

Check the data skewness in the 'origin' and 'destination' columns under the 'Regular' category for 'type'

*df['type'].value\_counts()*

**

*Figure 10. The type column contains three categorical items.*

*cat\_reg = df.groupby('type')*

*cat\_reg.get\_group('Regular').dropna()*

*reg = cat\_reg.get\_group('Regular').dropna()*

*skew(reg['origin'].dropna())*

**

*Figure 11. Skewness of ‘origin’ data when NA values are removed and grouped by ‘Regular’ category.*

*skew(reg['destination'].dropna())*

**

*Figure 12. Skewness of ‘destination’ data when NA values are removed and grouped by ‘Regular’ category.*

Check the data skewness in the 'origin' and 'destination' columns under the 'Ad-Hoc' category for 'type'

*cat\_ad = df.groupby('type')*

*cat\_ad.get\_group('Ad-Hoc').dropna()*

*ad = cat\_ad.get\_group('Ad-Hoc').dropna()*

*skew(ad['origin'].dropna())*

**

*Figure 13. Skewness of ‘origin’ data when NA values are removed and grouped by ‘Ad-Hoc’ category.*

*skew(ad['destination'].dropna())*

**

*Figure 14. Skewness of ‘destination’ data when NA values are removed and grouped by ‘Ad-Hoc’ category.*

Check the data skewness in the 'origin' and 'destination' columns under the 'Concession' category for 'type'

*cat\_con = df.groupby('type')*

*cat\_con.get\_group('Concession').dropna()*

*con = cat\_con.get\_group('Concession').dropna()*

*skew(con['origin'].dropna())*

**

*Figure 15. Skewness of ‘origin’ data when NA values are removed and grouped by ‘Concession’ category.*

*skew(con['destination'].dropna())*

**

*Figure 16. Skewness of ‘destination’ data when NA values are removed and grouped by ‘Concession’ category.*

Since all of the above data show a positive skewed distribution, we will use 'Median' to replace the 'Blank' value in each of the categories contained in the 'origin' and 'destination' columns.

*df['origin'].fillna(df.groupby(["type"])['origin'].transform('median')*

*,inplace = True)*

*df['destination'].fillna(df.groupby(["type"])['destination'].transform('median')*

*,inplace = True)*

In this stage, we have successfully treated and replaced the 'Blank' values in the dataset.

*df.isna().sum()*



*Figure 17. Output obtained for df.isna().sum() code*

**ii.**

The second data quality issue is the presence of outliers in the dataset. Outliers in data can reduce normality and influence statistical estimates, resulting in biased results. From the analysis, we can see that 'age' and 'yob' have some outliers in their dataset.

*def plotvariable(df, variable):*

*plt.figure(figsize=(16,4))*

*#histogram*

*plt.subplot(1,2,1)*

*plt.hist(df[variable], alpha=0.5)*

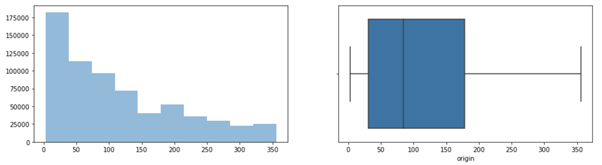
*#boxplot*

*plt.subplot(1,2,2)*

*sns.boxplot(df[variable])*

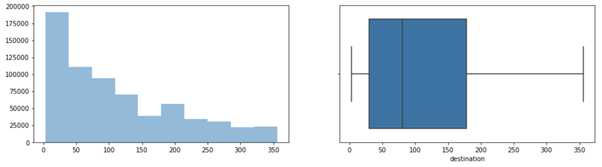
*Plt.show*

*plotvariable(df, 'origin')*

****

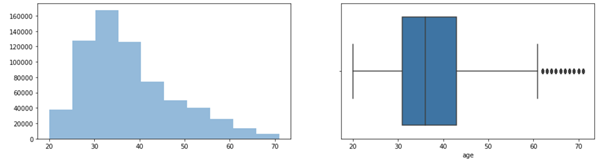
*Figure 19. Histogram and Box plot for ‘origin’*

*plotvariable(df, 'destination')*

****

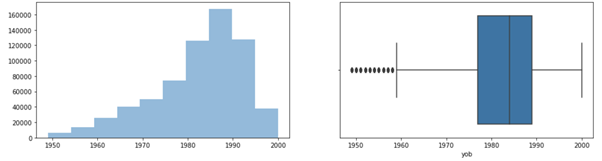
*Figure 20. Histogram and Box plot for ‘destination’*

*plotvariable(df, 'age')*

**

*Figure 21. Histogram and Box plot for ‘age’*

*plotvariable(df, 'yob')*

**

*Figure 22. Histogram and Box plot for ‘yob’*

To remove outliers from the data set, we can use the IQR calculation method to locate their location. Once detected, we can proceed to remove them with the following codes.

*# define a function called outliers which returns a list of index outliers*

*def outliers(df, ft):*

*Q1 = df[ft].quantile(0.25)*

*Q3 = df[ft].quantile(0.75)*

*IQR = Q3 - Q1*

*lower\_bound = Q1 - 1.5 \* IQR*

*upper\_bound = Q3 + 1.5 \* IQR*

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

*return ls*

*#create the index list*

*index\_list = []*

*for feature in ['age', 'yob']:*

*index\_list.extend(outliers(df, feature))*

*#generate the index list*

*index\_list*

*#define a function called to "remove" which return a cleaned dataframe without outliers*

*def remove(df, ls):*

*ls = sorted(set(ls))*

*df = df.drop(ls)*

*return df*

*df\_cleaned = remove(df, index\_list)*

*df\_cleaned.shape*



*Figure 23. Shape of new data frame*

In this stage, we have successfully removed the outliers contained in the dataset.

**iii.**

The third data quality issue is the conversion of the categorical columns ‘subscriber’, 'type' and 'gender' into dummy variables. This is because Python can only process numerical values in a dataset to perform machine learning.

Before dummifying the variables for the current dataset, we need to make a backup copy of the cleaned dataset for data exploration.

*df\_cleaned2 = df\_cleaned.copy()*

Next, we will apply the following codes.

*df\_cleaned = pd.get\_dummies(df, columns=['subscriber','type','gender'])*

*df\_cleaned.head()*

****

*Figure 18. Data frame output with dummy variables*

In this stage, we have successfully created the dummy variables for the columns ‘subscriber’, ‘type’ and ‘gender’.

**Qn1d** However,

**i.**

To accomplish this, we must write a function to assist us in converting the 'start' column from a string object to a time object and splitting the object by its hour and minute values.

# function to convert string in hh:mm.ss to time object

*def convert\_to\_time(time\_str):*

*# strip the part after .*

*time\_str = time\_str.split('.')[0]*

*# split the string into hh:mm*

*hh, mm = time\_str.split(':')*

*# wrap hour value to 24-hour format*

*hh = int(hh) % 24*

*# return as time object*

*return f'{hh}:{mm}'*

Next, the hour value in the 'start' column is then converted into a numeric range of 0 - 24 hours.

*# convert all times in the df.loc[:, 'start'] column to the range 0-24 hours*

*df\_cleaned.loc[:, 'start'] = df\_cleaned.loc[:, 'start'].apply(convert\_to\_time)*

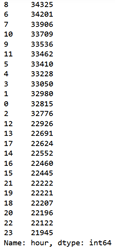
*# calculate hour value by splitting the string and taking the first part*

*df\_cleaned.loc[:, 'hour'] = df\_cleaned.loc[:, 'start'].apply(lambda x: int(x.split(':')[0]))*

#### We must ensure that the variables converted in the "start" column are distributed between 0 and 24 hours. Following that, the variable with the 'Most Frequent' occurrence is identified.

*# to confirm the most common hour value in 24-hour clock. From the value count, 0800hrs is the common value.*

*df\_cleaned['hour'].value\_counts()*

****

*Figure 24. Frequency count for each hourly variable in the ‘start’ column*

After confirming that the "Most Frequent" variable count is 8 (or 8 a.m.), we can run the following code.

*# print the most common hour value in 12-hour clock i.e. am or pm)*

*print(f"{df\_cleaned.loc[:, 'hour'].value\_counts().idxmax()} am")*



*Figure 25. Time (12-hour clock) whereby the highest number of commuters start their journey*

**Qn1e**

With Other is considered as Transexual identity, datas and analysis on Genders are worked on 3 genders.

First insight, with the following codes, we are able to generate and plot the comparison of the Subscribers distribution by Gender.

*Gender\_order = ['Female','Male','Other']*

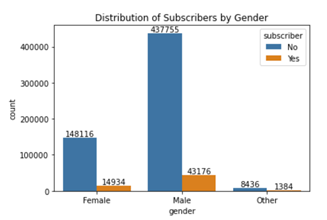
*ax = sns.countplot(x='gender',data=df\_cleaned2, hue='subscriber', order=Gender\_order)*

*for container in ax.containers:*

*ax.bar\_label(container)*

*p = ax*

*p.set\_title("Distribution of Subscribers by Gender")*



*Figure 26. A clustered column chart depicting the subscriber distribution by gender*

#### 

From Figure 26, with the use of column chart we are able to visualize and identify the ratio of the gender distribution of commuters who are subscribers and non-subscribers.

Female commuters with 148,116 non-subscribers and 14,934 subscribers, in a percentage of 90% and 10% respectively.

Male commuters with 437,755 non-subscribers and 43,176 subscribers, in a percentage of 90% and 10% respectively.

And other at 8,436 non-subscribers and 1,384 subscribers, in percentage of 86% and 14% respectively.

With the data above, we are able to assume that gender is not one of the key factors that will affect the conversion rate of subscribers. As for the different gender categories, the current subscriber rates are similar at approximately 10% - 15% of its gender commuters contribution.

We are also able to assume that the marketing effort to convert the non-subscriber to subscribers is not gender biased. As that is no difference between the subscribers distributions by genders.

Second insight, with the following codes, we are able to generate and plot the comparison of Distribution of Type by Gender.

*Gender\_order = ['Female','Male','Other']*

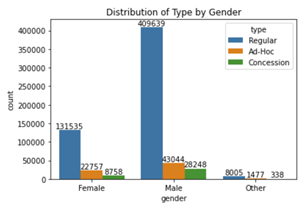
*ax = sns.countplot(x='gender',data=df\_cleaned2, hue='type', order=Gender\_order)*

*for container in ax.containers:*

*ax.bar\_label(container)*

*p = ax*

*p.set\_title("Distribution of Type by Gender")*



*Figure 27. A clustered column chart depicting type distribution by gender*

#### In Figure 27, we are able to see the column chart of the commuters type by gender.

#### 131,535 regular at 81%, 22,757 ad-hoc at 14% and 8,758 concession at 5% of Female commuters distribution.

#### 409,639 regular at 85%, 43,044 ad-hoc at 9% and 28,248 concession at 6% of Male commuters distribution.

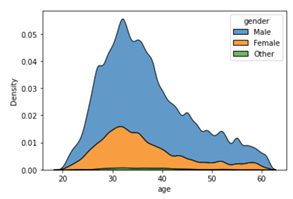
#### 8,005 regular at 82%, 1,417 ad-hoc at 15% and 338 concession at 3% of other commuters distribution.

From the data above, we are able to say that most of the commuters are under regular commuter type with more than 80% and a minimum are under concession type with less than 7% for all genders.

Looking at the type distribution by gender, as there are a higher ad-hoc commuters for female than male commuters, we are able to assume that the higher percentage can be due to non-fixed traveling by the homemakers as they do not have a fixed working hours which contributes to the regular commuters data.

Third insight, with the following codes, we are able to generate and plot the comparison of Density Distribution of Age by Gender.

*sns.kdeplot(data=df\_cleaned2, x='age', hue = 'gender', multiple='stack')*



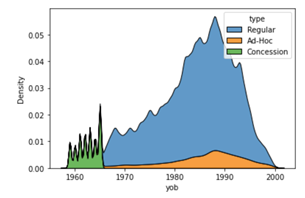
*Figure 28. Kernel density estimation chart depicting age distribution by gender*

#### In Figure 28, we are able to see that the peak of the distribution density for the 3 genders are similar at about 32 - 35 years old.

#### From this data, we can assume that the high volume of commuters for all genders in the early 30s is a result from accessing to and fro from their offices. And the density of the distributions dropped gradually can be due to the lifestyle of the people. For example, cars may be more affordable to people in their late 30s. Therefore, there’s a drop in commuters from 35 and beyond.

#### After generating the density distribution by gender, we can further generate a chart to visualize the density by yob.

*sns.kdeplot(data=df\_cleaned2, x='yob', hue = 'type', multiple='stack')*



*Figure 29. Kernel density estimation chart depicting yob distribution by type*

In Figure 29, we are able to see the density distribution of the commuters type by their year of birth. In the charts, concession type commuters can be found from the yob group of 1955 to 1965. And a high density peaking at yob 1987 - 1993 for both regular and ad-hoc commuters types.

From the density distribution of concession type, we are able to assume that the concessions are given to elderly, who are born before 1965. This can be an initiative from the government to promote healthy lifestyle and to keep the elderly active by providing discounted rates for access to transportation.

From the density distribution of regular type, we can assume that they are the working adults in their early 30s commuting to and fro from their offices. And ad-hoc peaking at also yob 1987, can be the homemakers without any fixed schedules.

With the above insights, we are able to conclude that most commuters are in their 30s with a higher ratio of male compared to female and others and most of the commuters are not subscribers. Assuming that we are tasked to increase the subscription volumes in the commuters, we can target the male commuters who are in their late 20s to early 30s. Despite gender not being the factor on the conversation rate, to increase the numbers of subscriptions in volume, we can first target to convert the male commuters as they have a higher distribution count in the total number of commuters.