**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

Program: BTI Computer Engineering

**Course: Data Mining**

**Experiment No.03**

PART A

(PART A: TO BE REFFERED BY STUDENTS)

**A.1 Aim:**

Apply Data Cleaning techniques for categorical, numerical attributes and observe the effect on data dispersion

**A.2 Prerequisite:**

**Python packages – matplotlib, pandas, numpy etc.**

**A.3 Outcome:**

**After successful completion of this experiment students will be able to:**

**Answer following questions:**

1. Different types of attribute in dataset
2. Count of Missing Values in Numerical and Categorical attribute
3. Handling the Missing Values
4. Handling Invalid values
5. Give the Count of Unique value for each attribute
6. Converting categorical data to numeric data
7. Visualizing the data after data cleaning using boxplot
8. Comment on the skewness of the data and remove outlier if any

**Task**

Select the dataset of your choice, clean the dataset and answer the above questions

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Black board access available)***

|  |  |
| --- | --- |
| **Roll No. C043** | **Name: Om Kamath** |
| **Class : B** | **Batch : B2** |
| **Date of Experiment: 1/8/23** | **Date of Submission: 8/8/23** |
| **Grade :** |  |

**B.1 Answers of Task to be written by student:**

***(Paste your answers completed during the 2 hours of practical in the lab here)***

### Importing Libraries

import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns

### Reading the file

df = pd.read\_csv('hotel\_bookings.csv')  
df.describe(include='all')

hotel is\_canceled lead\_time arrival\_date\_year \  
count 119390 119390.000000 119390.000000 119390.000000   
unique 2 NaN NaN NaN   
top City Hotel NaN NaN NaN   
freq 79330 NaN NaN NaN   
mean NaN 0.370416 104.011416 2016.156554   
std NaN 0.482918 106.863097 0.707476   
min NaN 0.000000 0.000000 2015.000000   
25% NaN 0.000000 18.000000 2016.000000   
50% NaN 0.000000 69.000000 2016.000000   
75% NaN 1.000000 160.000000 2017.000000   
max NaN 1.000000 737.000000 2017.000000   
  
 arrival\_date\_month arrival\_date\_week\_number \  
count 119390 119390.000000   
unique 12 NaN   
top August NaN   
freq 13877 NaN   
mean NaN 27.165173   
std NaN 13.605138   
min NaN 1.000000   
25% NaN 16.000000   
50% NaN 28.000000   
75% NaN 38.000000   
max NaN 53.000000   
  
 arrival\_date\_day\_of\_month stays\_in\_weekend\_nights \  
count 119390.000000 119390.000000   
unique NaN NaN   
top NaN NaN   
freq NaN NaN   
mean 15.798241 0.927599   
std 8.780829 0.998613   
min 1.000000 0.000000   
25% 8.000000 0.000000   
50% 16.000000 1.000000   
75% 23.000000 2.000000   
max 31.000000 19.000000   
  
 stays\_in\_week\_nights adults ... deposit\_type agent \  
count 119390.000000 119390.000000 ... 119390 103050.000000   
unique NaN NaN ... 3 NaN   
top NaN NaN ... No Deposit NaN   
freq NaN NaN ... 104641 NaN   
mean 2.500302 1.856403 ... NaN 86.693382   
std 1.908286 0.579261 ... NaN 110.774548   
min 0.000000 0.000000 ... NaN 1.000000   
25% 1.000000 2.000000 ... NaN 9.000000   
50% 2.000000 2.000000 ... NaN 14.000000   
75% 3.000000 2.000000 ... NaN 229.000000   
max 50.000000 55.000000 ... NaN 535.000000   
  
 company days\_in\_waiting\_list customer\_type adr \  
count 6797.000000 119390.000000 119390 119390.000000   
unique NaN NaN 4 NaN   
top NaN NaN Transient NaN   
freq NaN NaN 89613 NaN   
mean 189.266735 2.321149 NaN 101.831122   
std 131.655015 17.594721 NaN 50.535790   
min 6.000000 0.000000 NaN -6.380000   
25% 62.000000 0.000000 NaN 69.290000   
50% 179.000000 0.000000 NaN 94.575000   
75% 270.000000 0.000000 NaN 126.000000   
max 543.000000 391.000000 NaN 5400.000000   
  
 required\_car\_parking\_spaces total\_of\_special\_requests \  
count 119390.000000 119390.000000   
unique NaN NaN   
top NaN NaN   
freq NaN NaN   
mean 0.062518 0.571363   
std 0.245291 0.792798   
min 0.000000 0.000000   
25% 0.000000 0.000000   
50% 0.000000 0.000000   
75% 0.000000 1.000000   
max 8.000000 5.000000   
  
 reservation\_status reservation\_status\_date   
count 119390 119390   
unique 3 926   
top Check-Out 21-10-2015   
freq 75166 1461   
mean NaN NaN   
std NaN NaN   
min NaN NaN   
25% NaN NaN   
50% NaN NaN   
75% NaN NaN   
max NaN NaN   
  
[11 rows x 32 columns]

### Q1. Finding Different Attributes

attributes = df.columns.tolist()  
attributes

['hotel',  
 'is\_canceled',  
 'lead\_time',  
 'arrival\_date\_year',  
 'arrival\_date\_month',  
 'arrival\_date\_week\_number',  
 'arrival\_date\_day\_of\_month',  
 'stays\_in\_weekend\_nights',  
 'stays\_in\_week\_nights',  
 'adults',  
 'children',  
 'babies',  
 'meal',  
 'country',  
 'market\_segment',  
 'distribution\_channel',  
 'is\_repeated\_guest',  
 'previous\_cancellations',  
 'previous\_bookings\_not\_canceled',  
 'reserved\_room\_type',  
 'assigned\_room\_type',  
 'booking\_changes',  
 'deposit\_type',  
 'agent',  
 'company',  
 'days\_in\_waiting\_list',  
 'customer\_type',  
 'adr',  
 'required\_car\_parking\_spaces',  
 'total\_of\_special\_requests',  
 'reservation\_status',  
 'reservation\_status\_date']

### Q2. Finding Missing Numerical and Categorical Attributes

numerical\_columns = df.select\_dtypes(include=['int64', 'float64']).columns  
categorical\_columns = df.select\_dtypes(include=['object']).columns  
  
# Counting missing values in numerical and categorical columns  
missing\_values\_numerical = df[numerical\_columns].isnull().sum()  
missing\_values\_categorical = df[categorical\_columns].isnull().sum()  
  
print(missing\_values\_numerical)  
print(missing\_values\_categorical)

is\_canceled 0  
lead\_time 0  
arrival\_date\_year 0  
arrival\_date\_week\_number 0  
arrival\_date\_day\_of\_month 0  
stays\_in\_weekend\_nights 0  
stays\_in\_week\_nights 0  
adults 0  
children 4  
babies 0  
is\_repeated\_guest 0  
previous\_cancellations 0  
previous\_bookings\_not\_canceled 0  
booking\_changes 0  
agent 16340  
company 112593  
days\_in\_waiting\_list 0  
adr 0  
required\_car\_parking\_spaces 0  
total\_of\_special\_requests 0  
dtype: int64  
hotel 0  
arrival\_date\_month 0  
meal 0  
country 488  
market\_segment 0  
distribution\_channel 0  
reserved\_room\_type 0  
assigned\_room\_type 0  
deposit\_type 0  
customer\_type 0  
reservation\_status 0  
reservation\_status\_date 0  
dtype: int64

### Q3. Handling Missing Values

* children has 4 missing values -> Replace with Mode.
* country has 488 missing values -> Replace with 'Unknown'.
* agent has 16,340 missing values -> Replace with '0'
* company has 112,593 missing values -> Can be dropped

children\_mode = df['children'].mode()[0]  
  
df['children'].fillna(children\_mode, inplace=True)  
  
# Filling "country" with "Unknown"  
df['country'].fillna('Unknown', inplace=True)  
  
# Filling "agent" with "No Agent"  
df['agent'].fillna(0, inplace=True)  
  
# Dropping the "company" column  
df.drop(columns=['company'], inplace=True)

### Q4. Handling invalid values

Will check for negative values and replace that with mode.

# Finding rows with negative values in the specified columns  
for col in df.select\_dtypes(include=['float64', 'int64']).columns:  
 # print(df[col])  
 if (df[col]< 0).any():  
 mode\_value = df[col].mode()[0]  
 df.loc[df[col] < 0, col] = mode\_value

### Q5. Count of unique values

unique\_values = df.nunique()  
unique\_values

hotel 2  
is\_canceled 2  
lead\_time 479  
arrival\_date\_year 3  
arrival\_date\_month 12  
arrival\_date\_week\_number 53  
arrival\_date\_day\_of\_month 31  
stays\_in\_weekend\_nights 17  
stays\_in\_week\_nights 35  
adults 14  
children 5  
babies 5  
meal 5  
country 178  
market\_segment 8  
distribution\_channel 5  
is\_repeated\_guest 2  
previous\_cancellations 15  
previous\_bookings\_not\_canceled 73  
reserved\_room\_type 10  
assigned\_room\_type 12  
booking\_changes 21  
deposit\_type 3  
agent 334  
days\_in\_waiting\_list 128  
customer\_type 4  
adr 8878  
required\_car\_parking\_spaces 5  
total\_of\_special\_requests 6  
reservation\_status 3  
reservation\_status\_date 926  
dtype: int64

### Q6. Converting categorical data to numerical data

Using one-hot encoding since there is no hierarchical data.

data\_encoded = pd.get\_dummies(df)  
data\_encoded.head()

is\_canceled lead\_time arrival\_date\_year arrival\_date\_week\_number \  
0 0 342 2015 27   
1 0 737 2015 27   
2 0 7 2015 27   
3 0 13 2015 27   
4 0 14 2015 27   
  
 arrival\_date\_day\_of\_month stays\_in\_weekend\_nights stays\_in\_week\_nights \  
0 1 0 0   
1 1 0 0   
2 1 0 1   
3 1 0 1   
4 1 0 2   
  
 adults children babies ... reservation\_status\_date\_31-07-2015 \  
0 2 0.0 0 ... False   
1 2 0.0 0 ... False   
2 1 0.0 0 ... False   
3 1 0.0 0 ... False   
4 2 0.0 0 ... False   
  
 reservation\_status\_date\_31-07-2016 reservation\_status\_date\_31-07-2017 \  
0 False False   
1 False False   
2 False False   
3 False False   
4 False False   
  
 reservation\_status\_date\_31-08-2015 reservation\_status\_date\_31-08-2016 \  
0 False False   
1 False False   
2 False False   
3 False False   
4 False False   
  
 reservation\_status\_date\_31-08-2017 reservation\_status\_date\_31-10-2015 \  
0 False False   
1 False False   
2 False False   
3 False False   
4 False False   
  
 reservation\_status\_date\_31-10-2016 reservation\_status\_date\_31-12-2015 \  
0 False False   
1 False False   
2 False False   
3 False False   
4 False False   
  
 reservation\_status\_date\_31-12-2016   
0 False   
1 False   
2 False   
3 False   
4 False   
  
[5 rows x 1187 columns]

### Q6. Visualization

Using box-plot (should be performed on original dataframe since encoded one will definitely consist outliers)

columns\_to\_plot = ['lead\_time', 'stays\_in\_weekend\_nights', 'stays\_in\_week\_nights', 'adr']  
  
# Create a boxplot for each column  
for column in columns\_to\_plot:  
 plt.figure(figsize=(5, 5))  
 sns.boxplot(y=df[column])  
 plt.title('Boxplot of ' + column)  
 plt.show()

A graph of a diagram

Description automatically generated

A graph with a line and a line

Description automatically generated with medium confidence

A graph of a graph

Description automatically generated

A graph with a line and a line

Description automatically generated with medium confidence

### Q7. Remove Outliers

def remove\_outliers(df, column):  
 Q1 = df[column].quantile(0.25)  
 Q3 = df[column].quantile(0.75)  
 IQR = Q3 - Q1  
  
 # Define the range for outliers  
 lower\_bound = Q1 - 1.5 \* IQR  
 upper\_bound = Q3 + 1.5 \* IQR  
  
 # Remove the outliers  
 df\_out = df[(df[column] >= lower\_bound) & (df[column] <= upper\_bound)]  
   
 return df\_out

for col in columns\_to\_plot:  
 df\_without\_outliers = remove\_outliers(df, col)  
 plt.figure(figsize=(5, 5))  
 sns.boxplot(y=df\_without\_outliers[col])  
 plt.title('Boxplot of ' + col)  
 plt.show()

A diagram of a box with a line

Description automatically generated with medium confidence

A graph of a blue rectangular object

Description automatically generated

A graph with blue rectangles and black lines

Description automatically generated

A diagram of a box plot

Description automatically generated

**Except for adr, all other visualizations are positively skewed. Adr is fairly normally distributed.**

**B.2 Observations and learning:**

***(Students are expected to comment on the output obtained with clear observations and learning for each task/ sub part assigned)***

**Understood how to perform operations and analyse datasets using Exploratory Data Analysis.**

**B.3 Conclusion:**

*(****Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation noted in section B.3)***

**Understood how to perform operations and analyse datasets using Exploratory Data Analysis. The dataset consisted of few empty and invalid values which have now been handled. A visualization helped in finding out the outliers.**