Batch: B2 **Experiment Number:1**

Roll Number: 16010421073 **Name: Keyur Patel**

Aim of the Experiment: Data pre-processing by applying data normalization and data discretization

---

Output/Result:

Program/ Steps: 1) Importing modules

```
import pandas as pd

import numpy as np
```

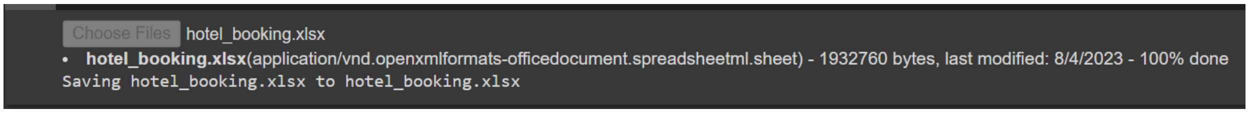
2) Importing the excel file

```
import io

import pandas as pd

from google.colab import files

uploaded=files.upload()
```



hotel_booking.xlsx

- **hotel_booking.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 1932760 bytes, last modified: 8/4/2023 - 100% done

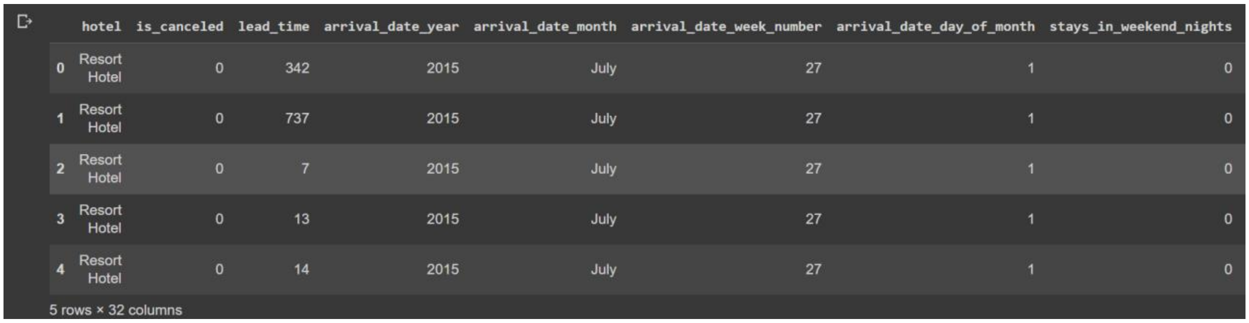
```
Saving hotel_booking.xlsx to hotel_booking.xlsx
```

3) Storing the dataset in a variable df

```
df=pd.read_excel(io.BytesIO(uploaded['hotel_booking.xlsx']))
```

4) First 5 rows of the dataset

```
df.head()
```



	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights
0	Resort Hotel	0	342	2015	July	27	1	0
1	Resort Hotel	0	737	2015	July	27	1	0
2	Resort Hotel	0	7	2015	July	27	1	0
3	Resort Hotel	0	13	2015	July	27	1	0
4	Resort Hotel	0	14	2015	July	27	1	0

5 rows × 32 columns

5) Choosing an attribute for normalization

Attribute chosen – Lead time

Reason for choosing this attribute for normalization :

Lead time is likely to have a wide range of values and can vary significantly. Normalizing this attribute will scale it to a common range, typically between 0 and 1, making it easier for machine learning algorithms to handle. Normalization ensures that each data point is relative to the entire range of values in the "lead_time" attribute.

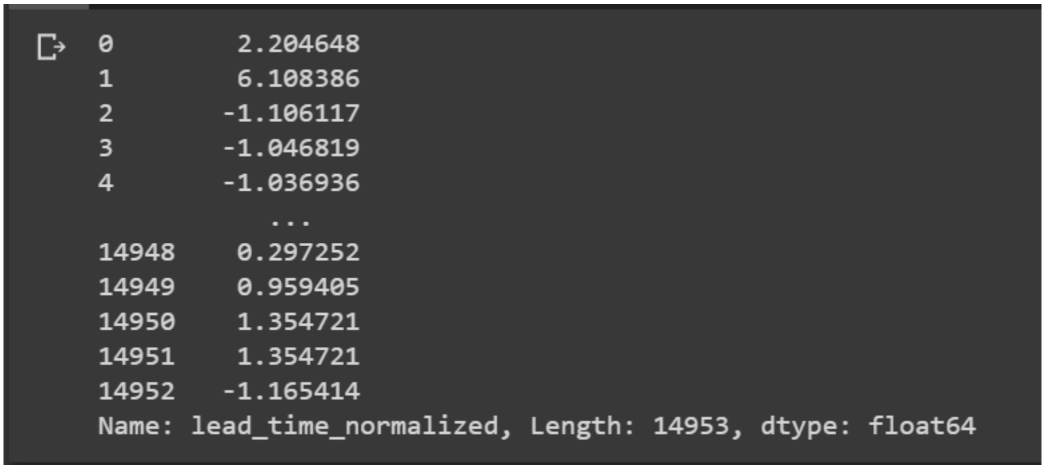
```
lead_time = df['lead_time']

mean_lead_time = lead_time.mean()

std_lead_time = lead_time.std()

df['lead_time_normalized'] = (lead_time - mean_lead_time) / std_lead_time
```

```
print(df['lead_time_normalized'])
```



```
0        2.204648
1        6.108386
2       -1.106117
3       -1.046819
4       -1.036936
           ...
14948    0.297252
14949    0.959405
14950    1.354721
14951    1.354721
14952   -1.165414
Name: lead_time_normalized, Length: 14953, dtype: float64
```

6) Choosing an attribute for discretization

Attribute chosen – adr (average daily rate)

Reason for choosing this attribute for discretization :

Discretizing the average daily rate can be beneficial if you want to group the values into specific categories or bins. For example, you can create bins for different price ranges like "low," "medium," and "high" to represent different levels of hotel room rates. Discretization can help reduce the impact of outliers and make the data more manageable and interpretable for certain types of machine learning algorithms.

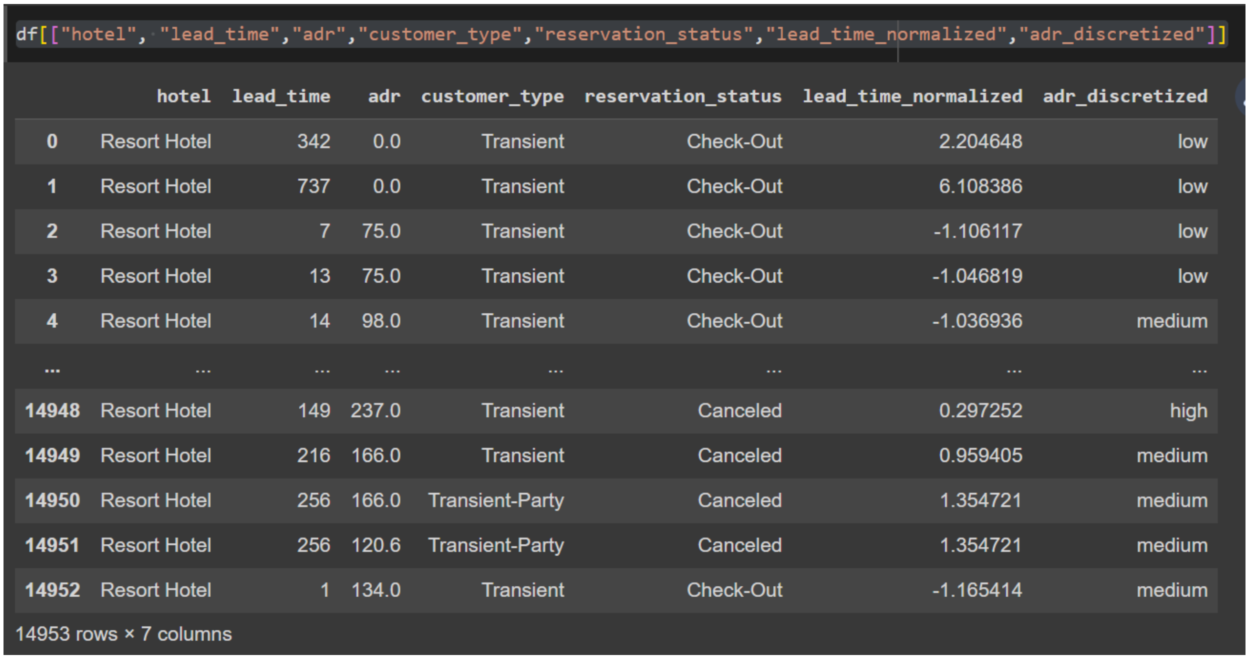
Creating number of bins: `num_bins = 5`

```
labels = ['low', 'medium', 'high', 'very high', 'extremely high']

df['adr_discretized'] = pd.cut(df['adr'], bins=num_bins, labels=labels)
```

```
df[["hotel","lead_time","adr","customer_type","reservation_status","lead_time_normalized","adr_discretized"]]
```

Both lead_time_normalized and adr_discretized columns were added to the dataset



```
df[["hotel", "lead_time","adr","customer_type","reservation_status","lead_time_normalized","adr_discretized"]]
```

	hotel	lead_time	adr	customer_type	reservation_status	lead_time_normalized	adr_discretized
0	Resort Hotel	342	0.0	Transient	Check-Out	2.204648	low
1	Resort Hotel	737	0.0	Transient	Check-Out	6.108386	low
2	Resort Hotel	7	75.0	Transient	Check-Out	-1.106117	low
3	Resort Hotel	13	75.0	Transient	Check-Out	-1.046819	low
4	Resort Hotel	14	98.0	Transient	Check-Out	-1.036936	medium
...	...	...	...	...	...	...	...
14948	Resort Hotel	149	237.0	Transient	Canceled	0.297252	high
14949	Resort Hotel	216	166.0	Transient	Canceled	0.959405	medium
14950	Resort Hotel	256	166.0	Transient-Party	Canceled	1.354721	medium
14951	Resort Hotel	256	120.6	Transient-Party	Canceled	1.354721	medium
14952	Resort Hotel	1	134.0	Transient	Check-Out	-1.165414	medium

14953 rows × 7 columns

Questions:

1. Explain with example Min-Max normalization technique.

Ans: Min-Max normalization, also known as feature scaling, is a data normalization technique used to transform numerical data into a specific range, typically between 0 and 1. It linearly scales the original values to fit within this range based on the minimum and maximum values of the attribute. This normalization is given by the formula:

$$\boxed{v' = \frac{v - \min(A)}{\max(A) - \min(A)}(\text{new\_max}(A) - \text{new\_min}(A)) + \text{new\_min}(A)}$$

To understand the formula, here is an example. Suppose a company wants to decide on a promotion based on the years of work experience of its employees. So, it needs to analyze a database that looks like this:

Employee Name	Years of Experience
ABC	8
XYZ	20
PQR	10
MNO	15

- The minimum value is 8
- The maximum value is 20

As this formula scales the data between 0 and 1,

- The new min is 0
- The new max is 1

Here, V stands for the respective value of the attribute, i.e., 8, 10, 15, 20

After applying the min-max normalization formula, the following are the **V'** values for the attributes:

- For 8 years of experience: **v'= 0**
- For 10 years of experience: **v' = 0.16**
- For 15 years of experience: **v' = 0.58**
- For 20 years of experience: **v' = 1**

So, the min-max normalization can reduce big numbers to much smaller values. This makes it extremely easy to read the difference between the ranging numbers.

Outcomes: CO1: Comprehend basics of machine learning

Conclusion (based on the Results and outcomes achieved):

Through this experiment we learned the concepts of data preprocessing by by applying data normalization and data discretization.

References:

Books/ Journals/ Websites:

1. Han, Kamber, "Data Mining Concepts and Techniques", Morgan Kaufmann 3nd Edition