

**Experiment No.1 Title:** Data Pre-processing

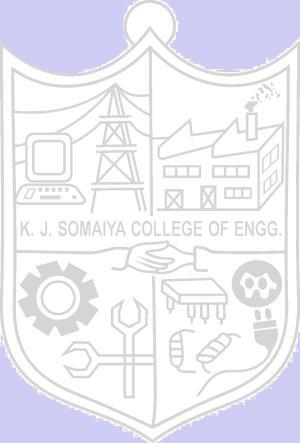
# Batch: ML 2 Roll No.: 16010421103 Experiment No.:1

**Aim**:

**Resources needed:** Any RDBMS, Java

# Theory:

Data processing techniques, when applied before mining, can substantially improve the overall quality of the patterns mined and/or the time required for the actual mining.

Different kinds of pre-processing tasks are performed on the data before applying mining techniques. Data reduction can reduce data size by, for instance, aggregating, eliminating redundant features, or clustering. *Data transformations* (e.g., normalization) may be applied, where data are scaled to fall within a smaller range like 0.0 to 1.0. This can improve the accuracy and efficiency of mining algorithms involving distance measurements. These techniques are not mutually exclusive; they may work together. Normalization, data discretization, and concept hierarchy generation are forms of data transformation.

# Data Reduction:

Data reduction techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. That is, mining on the reduced data set should be more efficient yet produce the same (or almost the same) analytical results. Data reduction strategies include dimensionality reduction, numerosity reduction, and data compression.

Dimensionality reduction is the process of reducing the number of random variables or attributes under consideration. Dimensionality reduction methods include wavelet transforms and principal components analysis, which transform or project the original data onto a smaller space. Attribute subset selection is a method of dimensionality reduction in which irrelevant, weakly relevant, or redundant attributes or dimensions are detected and removed.

Numerosity reduction techniques replace the original data volume by alternative, smaller forms of data representation. These techniques may be parametric or nonparametric. For parametric methods, a model is used to estimate the data, so that typically only the data parameters need to be stored, instead of the actual data. (Outliers may also be stored.) Regression and log-linear models are examples. Nonparametric methods for storing reduced representations of the data include histograms, clustering, sampling, and data cube aggregation.

# Data Normalization:

The measurement unit used can affect the data analysis. For example, changing measurement units from meters to inches for *height*, or from kilograms to pounds for *weight*, may lead to very different results. In general, expressing an attribute in smaller units will lead to a larger range for that attribute, and thus tend to give such an attribute greater effect or “weight.” To

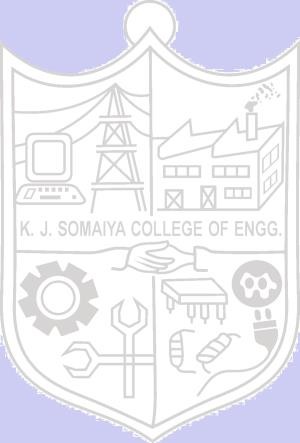
help avoid dependence on the choice of measurement units, the data should be *normalized* or *standardized*. This involves transforming the data to fall within a smaller or common range such as [-1, 1] or [0.0, 1.0].

In ***z*-score normalization** (or *zero-mean normalization*), the values for an attribute, *A*, are normalized based on the mean (i.e., average) and standard deviation of *A*. A value, *vi* , of *A* is normalized to *v*’i by computing ,



where σ*A* and 𝐴̅ are the mean and standard deviation, respectively, of attribute *A*. This method of normalization is useful when the actual minimum and maximum of attribute *A* are unknown, or when there are outliers that dominate the min-max normalization. **z-score normalization.** Suppose that the mean and standard deviation of the values for the attribute *income* are $54,000 and $16,000, respectively. With z-score normalization, a value of

$73,600 for *income* is transformed to,



# Data Discretization:

In data descretization, the raw values of a numeric attribute (e.g., *age*) are replaced by interval labels (e.g., 0–10, 11–20, etc.) or conceptual labels (e.g., *youth, adult*, *senior*). The labels, in turn, can be recursively organized into higher-level concepts, resulting in a *concept hierarchy* for the numeric attribute.

# Binning:

Binning is a top-down splitting technique based on a specified number of bins. These methods are also used as discretization methods for data reduction and concept hierarchy generation. For example, attribute values can be discretized by applying equal-width or equal-frequency binning, and then replacing each bin value by the bin mean or median, as in *smoothing by bin means* or *smoothing by bin medians*, respectively. These techniques can be applied recursively to the resulting partitions to generate concept hierarchies. Binning does not use class information and is therefore an unsupervised discretization technique. It is sensitive to the user-specified number of bins, as well as the presence of outliers. Example is shown in figure 1.

# Results: (Program printout with output / Document printout as per the format)

# Code:

# 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()

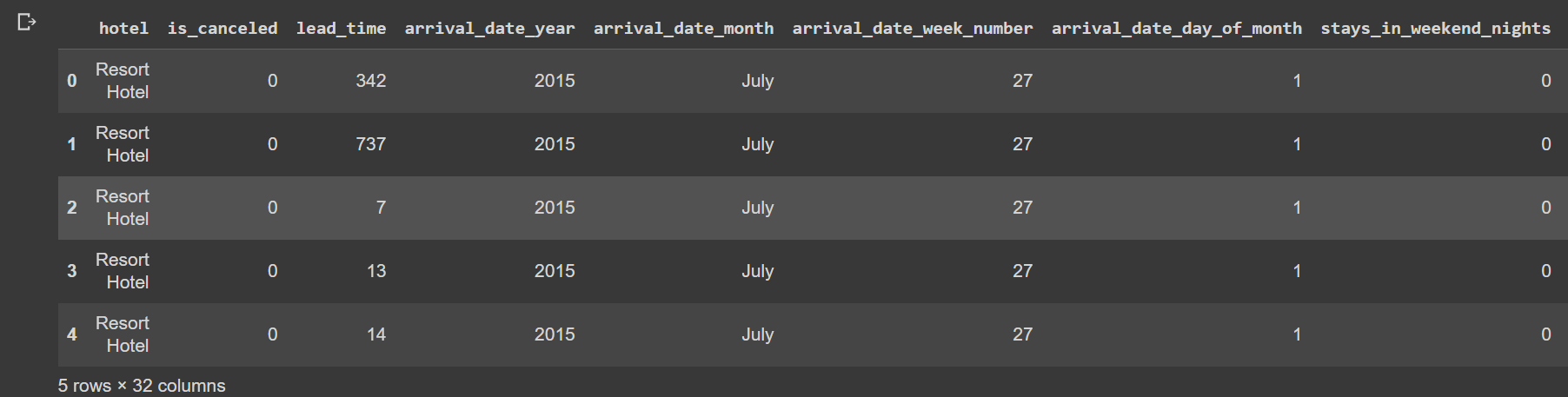
# 

**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()

****

**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.

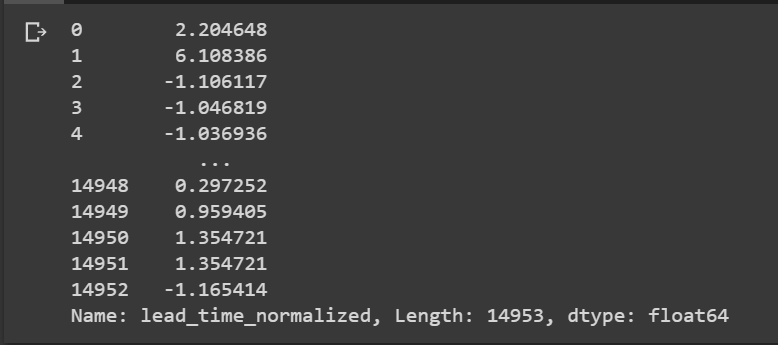
lead\_time = df['lead\_time']

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'])



**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.

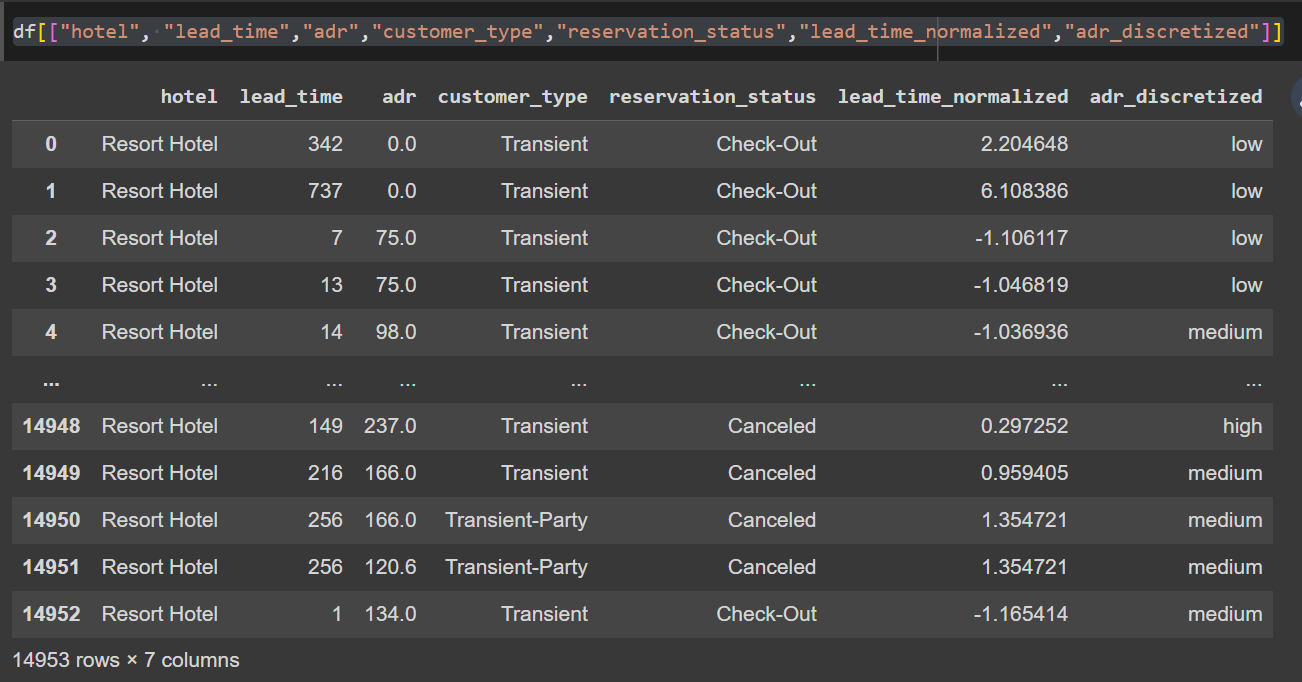
Creating number of bins: num\_bins = 5

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**

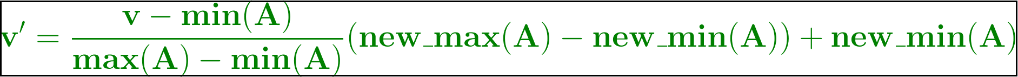




**Questions:**

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:



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: (Conclusion to be based on the objectives 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