



Experiment No.1

Title: Data Pre-processing



Batch: A1 **Roll No.: 16010422012**
Experiment No.:1

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

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:

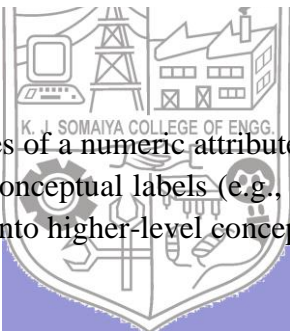
The measurement unit used can affect the data analysis. For example, changing measurement units from meters to inches for *height*, or from kilograms to pounds for *weight*, may lead to very different results. In general, expressing an attribute in smaller units will lead to a larger range for that attribute, 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, v_i , of A is normalized to v'_i by computing ,

$$v'_i = \frac{v_i - \bar{A}}{\sigma_A},$$

where \bar{A} and σ_A 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,

$$\frac{73,600 - 54,000}{16,000} = 1.225.$$


Data Discretization:

In data discretization, 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.

Sorted data for *price* (in dollars): 4, 8, 15, 21, 21, 24, 25, 28, 34

Partition into (equal-frequency) bins:	
Bin 1:	4, 8, 15
Bin 2:	21, 21, 24
Bin 3:	25, 28, 34
Smoothing by bin means:	
Bin 1:	9, 9, 9
Bin 2:	22, 22, 22
Bin 3:	29, 29, 29
Smoothing by bin boundaries:	
Bin 1:	4, 4, 15
Bin 2:	21, 21, 24
Bin 3:	25, 25, 34

Fig 1. Example of binning

Other methods of Discretization are,

- Histogram Analysis
- Discretization by Cluster, Decision Tree, Correlation Analyses

Procedure / Approach /Algorithm / Activity Diagram:

1. Identify attribute suitable for normalization and discretization
 2. Apply Z- score normalization on your dataset.
 3. Apply discretization using Binning technique
-

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

```
import random
import numpy as np
import matplotlib.pyplot as plt

data = [random.randint(0, 1000) for _ in range(100)]

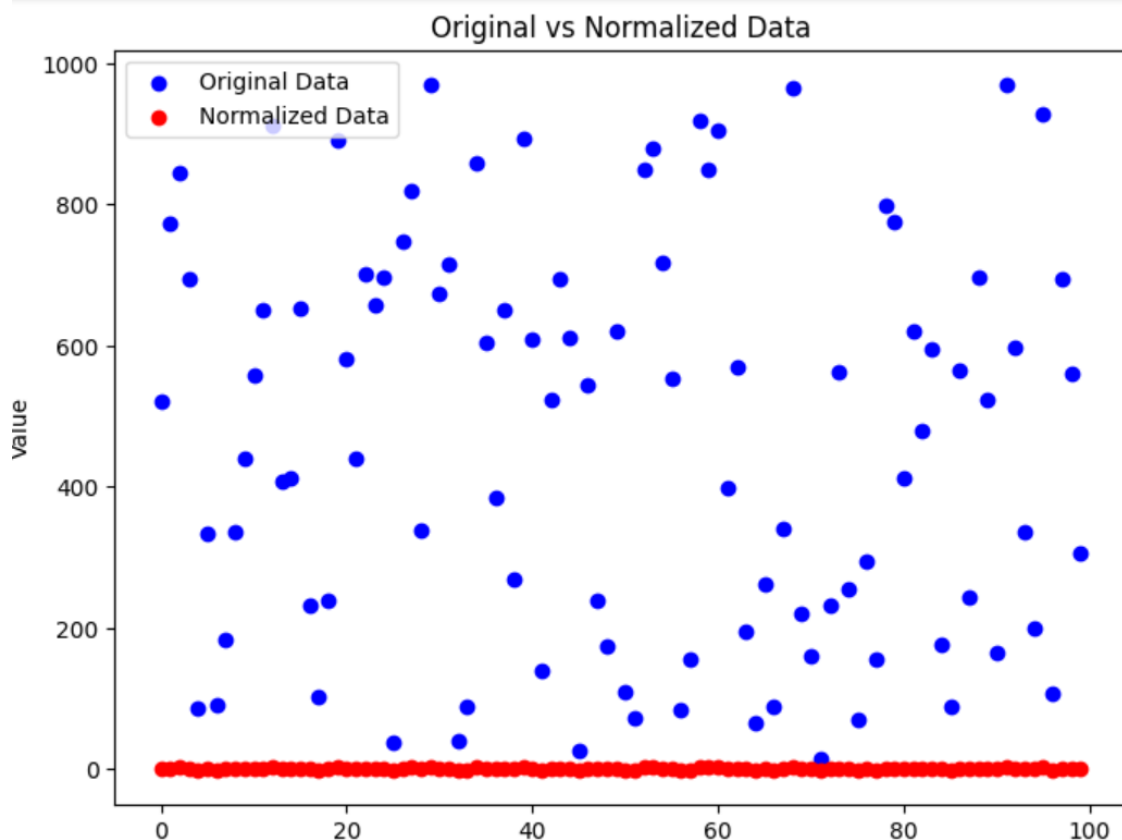
mean = np.mean(data)
std = np.std(data)

normalized_data = [(x - mean) / std for x in data]

print("Original Value | Normalized Value")
print("-----|-----")
for original, normalized in zip(data, normalized_data):
```

```
print(f'{{original:<15}} | {{normalized:.2f}}')
```

```
plt.figure(figsize=(8, 6))
plt.scatter(range(len(data)), data, color='blue', label='Original Data')
plt.scatter(range(len(normalized_data)), normalized_data, color='red',
label='Normalized Data')
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Original vs Normalized Data')
plt.legend()
plt.show()
```



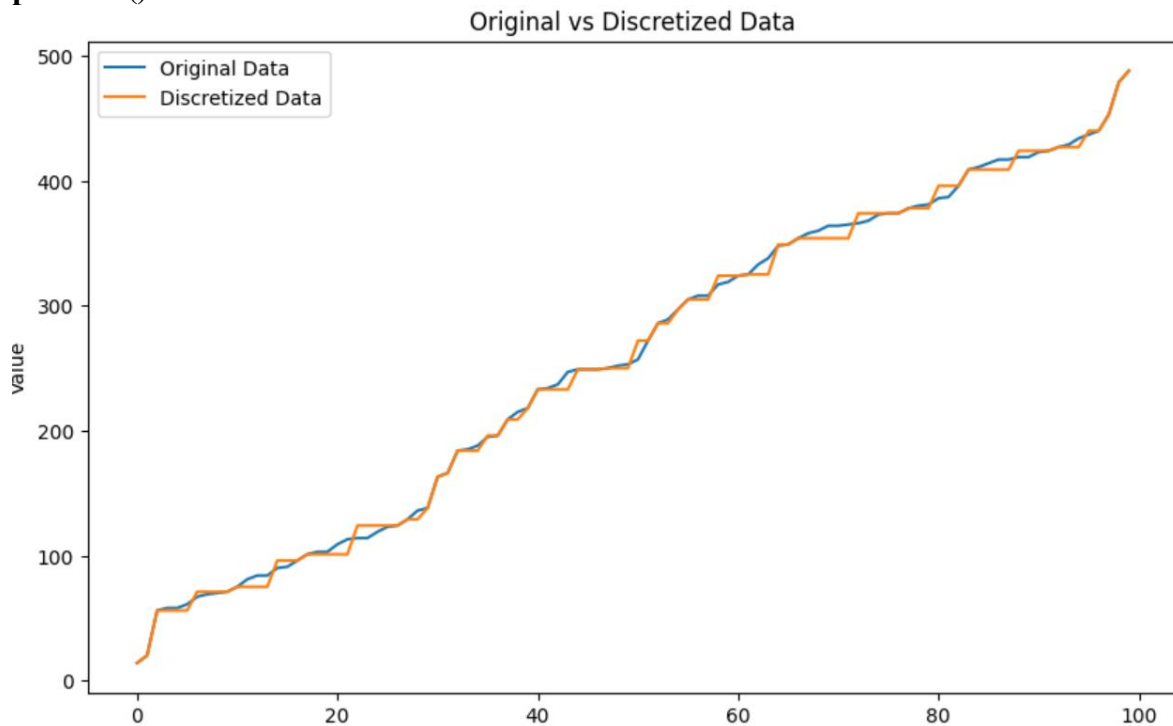
```
import random
import numpy as np
```

```
data = [random.randint(0, 500) for _ in range(100)]
data.sort()
```

```
discretized_data = []
for i in range(0, 501, 25):
    group = [x for x in data if i <= x < i + 25]
    if group:
        low = min(group)
        high = max(group)
        percentile_50 = np.percentile(group, 50)
        for x in group:
            discretized_data.append(low if x <= percentile_50 else high)
```

```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(10, 6))
plt.plot(data, label='Original Data')
plt.plot(discretized_data, label='Discretized Data')
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Original vs Discretized Data')
plt.legend()
plt.show()
```

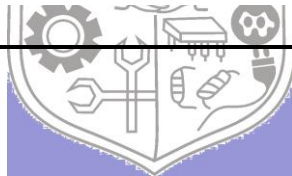
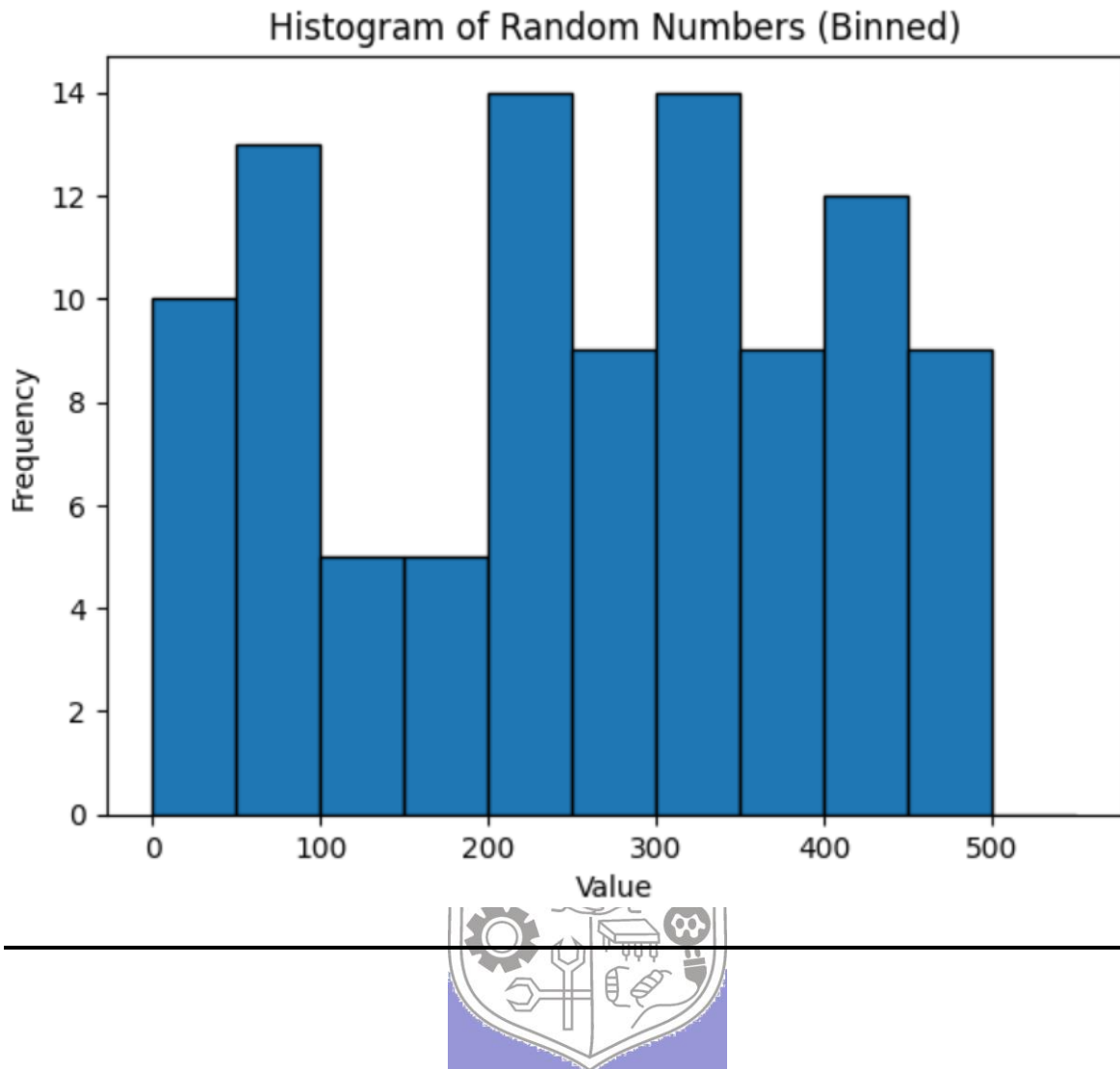


```
import random
import matplotlib.pyplot as plt
```

```
data = [random.randint(0, 500) for _ in range(100)]
```

```
bin_edges = range(0, 551, 50) # Bins: 0-50, 51-100, ..., 451-500
```

```
plt.hist(data, bins=bin_edges, edgecolor='black')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram of Random Numbers (Binned)')
plt.show()
```



Questions:

1. Explain with example Min-Max normalization technique.

Min-Max normalization is a technique used to scale data to a specific range, typically between 0 and 1. The formula used is $X' = \frac{X - \min(X)}{\max(X) - \min(X)}$, where X' is the normalized value, X is the original value, and $\min(X)$ and $\max(X)$ are the minimum and maximum values in the dataset, respectively. For example, if you have a dataset of ages ranging from 18 to 60 and you want to normalize the value 30, you'd compute $\frac{30-18}{60-18} \approx 0.29$. This method ensures that all values are rescaled within the desired range, making it useful for algorithms sensitive to the scale of data, such as neural networks.

Outcomes: Comprehend basics of ML.

Conclusion: (Conclusion to be based on the objectives and outcomes achieved)

Machine learning often requires preprocessing techniques like normalization to improve algorithm performance. Min-Max normalization scales data to a specific range, such as [0, 1], which can be essential when different attributes vary widely in their ranges. This process ensures that no single attribute dominates the others due to its scale. Additionally, other techniques like data discretization and reduction help to simplify data, making algorithms more efficient without compromising the integrity of results. Overall, these preprocessing steps are crucial for accurate and efficient data analysis in machine learning.

Grade: AA / AB / BB / BC / CC / CD / DD

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

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

