**Batch: HO-ML 1 Experiment Number: 01**

**Roll Number: 16010422234 Name: Chandana Galgali**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

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

**Program/ Steps:**

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

**Output/Result:**

**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()**

**Original Value | Normalized Value**

**---------------|-----------------**

**569 | 0.22**

**4 | -1.74**

**426 | -0.28**

**994 | 1.69**

**960 | 1.57**

**979 | 1.64**

**101 | -1.40**

**968 | 1.60**

**36 | -1.62**

**601 | 0.33**

**733 | 0.78**

**614 | 0.37**

**211 | -1.02**

**529 | 0.08**

**357 | -0.52**

**317 | -0.65**

**578 | 0.25**

**736 | 0.80**

**308 | -0.68**

**691 | 0.64**

**817 | 1.08**

**644 | 0.48**

**531 | 0.09**

**44 | -1.60**

**986 | 1.66**

**29 | -1.65**

**113 | -1.36**

**45 | -1.59**

**612 | 0.37**

**649 | 0.49**

**300 | -0.71**

**677 | 0.59**

**486 | -0.07**

**221 | -0.99**

**651 | 0.50**

**684 | 0.62**

**290 | -0.75**

**362 | -0.50**

**844 | 1.17**

**537 | 0.11**

**893 | 1.34**

**356 | -0.52**

**479 | -0.09**

**826 | 1.11**

**344 | -0.56**

**382 | -0.43**

**501 | -0.02**

**300 | -0.71**

**172 | -1.15**

**29 | -1.65**

**245 | -0.90**

**46 | -1.59**

**440 | -0.23**

**647 | 0.49**

**540 | 0.12**

**89 | -1.44**

**877 | 1.28**

**205 | -1.04**

**851 | 1.19**

**907 | 1.39**

**35 | -1.63**

**819 | 1.08**

**843 | 1.17**

**914 | 1.41**

**283 | -0.77**

**61 | -1.54**

**385 | -0.42**

**505 | -0.00**

**822 | 1.09**

**851 | 1.19**

**577 | 0.25**

**445 | -0.21**

**602 | 0.33**

**196 | -1.07**

**894 | 1.34**

**400 | -0.37**

**575 | 0.24**

**451 | -0.19**

**894 | 1.34**

**618 | 0.39**

**200 | -1.06**

**481 | -0.09**

**277 | -0.79**

**982 | 1.65**

**767 | 0.90**

**643 | 0.47**

**778 | 0.94**

**623 | 0.40**

**978 | 1.63**

**42 | -1.60**

**894 | 1.34**

**567 | 0.21**

**519 | 0.04**

**189 | -1.10**

**132 | -1.29**

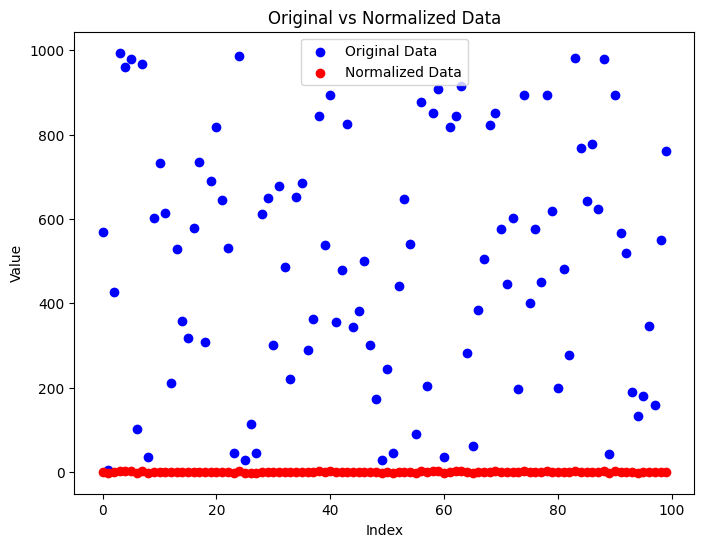
**180 | -1.13**

**347 | -0.55**

**158 | -1.20**

**551 | 0.16**

**760 | 0.88**

****

**import random**

**import numpy as np**

**import matplotlib.pyplot as plt**

**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)**

**plt.figure(figsize=(10, 6))**

**plt.plot(data, label='Original Data')**

**plt.plot(discretized\_data, label='Discretized Data')**

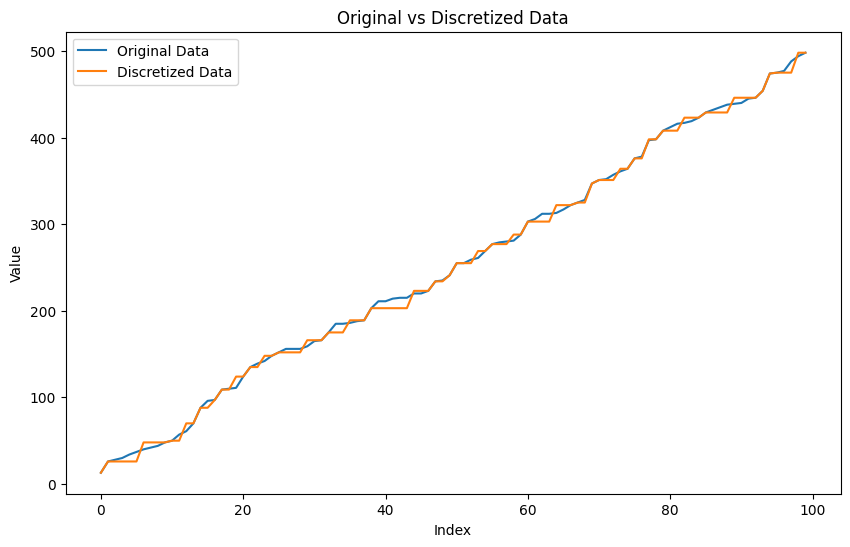
**plt.xlabel('Index')**

**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**

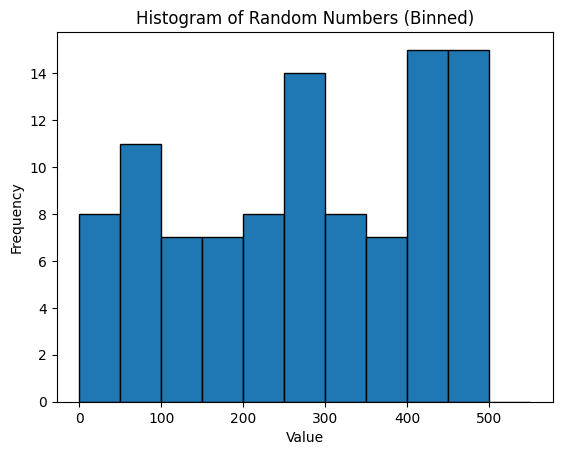
**plt.hist(data, bins=bin\_edges, edgecolor='black')**

**plt.xlabel('Value')**

**plt.ylabel('Frequency')**

**plt.title('Histogram of Random Numbers (Binned)')**

**plt.show()**

****

**Post Lab Question-Answers:**

**Explain with example Min-Max normalization technique.**

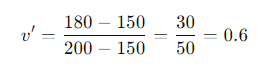
**Ans:** Min-Max Normalization is a data normalization technique used to scale data values to a fixed range, typically [0, 1]. This technique ensures that all attributes contribute equally to the analysis by transforming the data values into a uniform scale.



where:

* v is the original value.
* min(A) is the minimum value of attribute AAA.
* max(A) is the maximum value of attribute AAA.
* v′ is the normalized value.

Example: Suppose we have an attribute "Height" with values ranging from 150 cm to 200 cm. To normalize a value of 180 cm:

1. Determine the Min and Max Values:
   * Min = 150 cm
   * Max = 200 cm
2. Apply the Min-Max Formula:  
   

Thus, the normalized value of 180 cm is 0.6, which falls within the [0, 1] range.

This normalization method is straightforward and effective for scaling data when the minimum and maximum values are known and when it is crucial to maintain the data within a specific range for uniformity in analysis.

**Outcomes: Comprehend basics of machine learning**

**Conclusion (based on the Results and outcomes achieved):**

In this experiment, applying z-score normalization and binning for discretization effectively improved data quality and mining efficiency. Normalization ensured uniformity across attributes, while discretization simplified continuous data into manageable intervals. These pre-processing techniques enhanced the accuracy and computational efficiency of subsequent data mining tasks.

**References:**

Books/ Journals/ Websites:

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