Name :- Anugrah Kulkarni
Div :- TE09-28
Batch :- B
Subject :- DWM

EXPERIMENT NO 7:-Implementation of data discretization and visualization

Aim: The aim of this task is to implement **Data Discretization** using the **Equal Width Discretization** method and visualize the results. Data discretization is the process of converting continuous data into discrete categories or bins. The goal is to reduce the complexity of data while preserving its important characteristics.

Algorithm:

1. **Input the continuous data**.
2. **Choose the number of bins** or intervals for discretization.
3. **Calculate the width of each bin** using the formula:

   Bin Width=Max Value−Min ValueNumber of Bins\text{Bin Width} = \frac{\text{Max Value} - \text{Min Value}}{\text{Number of Bins}}Bin Width=Number of BinsMax Value−Min Value

4. **Divide the data into equal-width intervals** (bins) based on the calculated bin width.
5. **Assign each data point** to its respective bin.
6. **Plot the histogram** of the original continuous data and the discretized data for visualization.

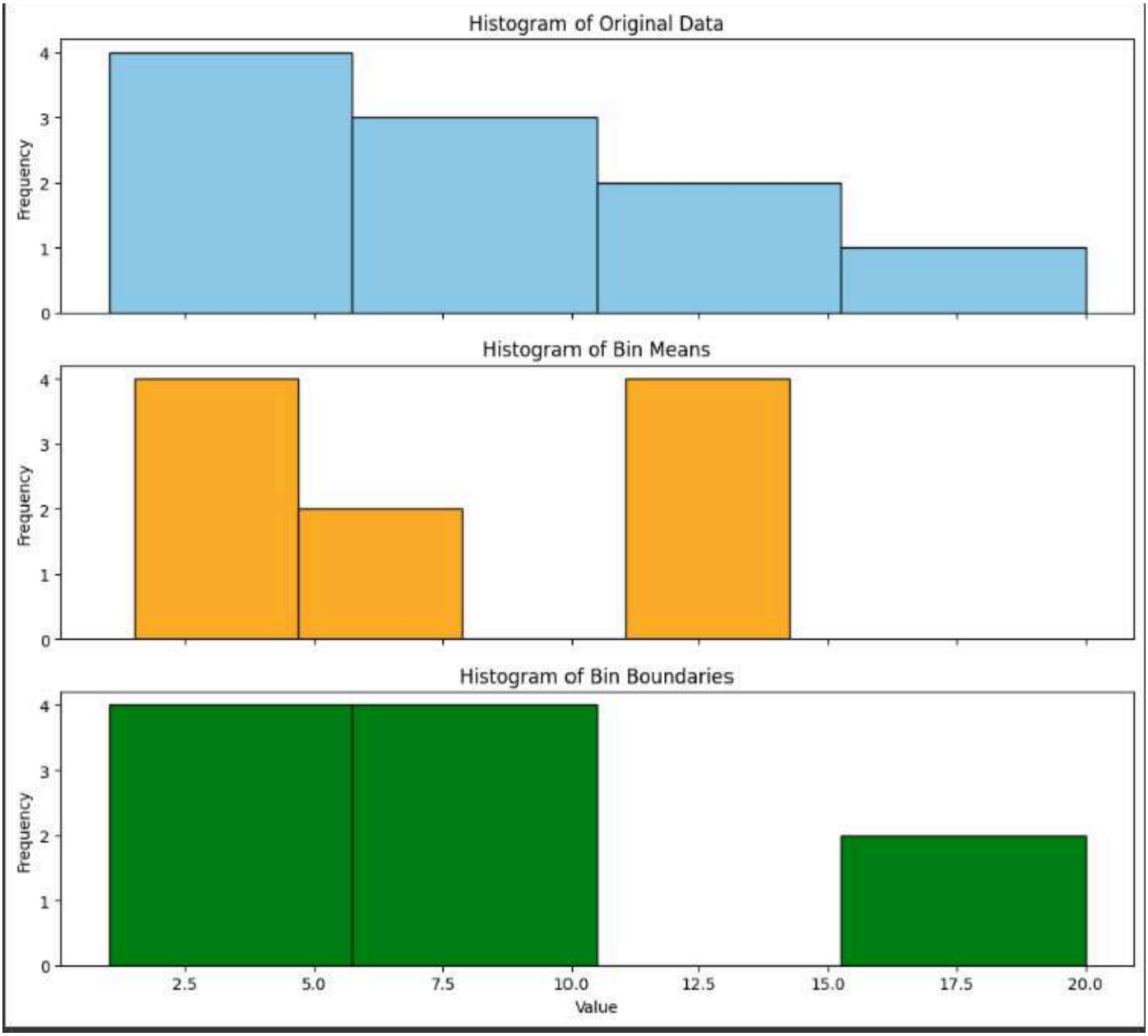
Program:

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

# Function to perform equi-depth binning
def equi_depth_binning(data, num_bins):
    sorted_data = sorted(data)
    bin_size = len(data) // num_bins
    bins = [sorted_data[i * bin_size: (i + 1) * bin_size] for i in range(num_bins - 1)]
    bins.append(sorted_data[(num_bins - 1) * bin_size:])
    return bins

# Replace each value in the bin with the bin mean
def bin_means(bins):
    return [[np.mean(bin)] * len(bin) for bin in bins]

# Replace each value in the bin with the closest boundary
def bin_boundaries(bins):
    new_bins = []
    for bin in bins:
        min_val, max_val = min(bin), max(bin)
        new_bins.append([min_val if x < (min_val + max_val) / 2 else max_val for x in bin])
    return new_bins

# Function to plot histograms
def plot_separate_histograms(data, bins, mean_bins, boundary_bins):
    # Flatten the bins for mean and boundary bins
    flattened_means = [val for bin in mean_bins for val in bin]
    flattened_boundaries = [val for bin in boundary_bins for val in bin]

    # Create subplots for three histograms with reduced height
    fig, axs = plt.subplots(3, 1, figsize=(10, 9), sharex=True)

    # Histogram for original data
```

```
    axs[0].hist(data, bins=len(bins), color='skyblue', edgecolor='black')
    axs[0].set_title("Histogram of Original Data")
    axs[0].set_ylabel("Frequency")

    # Histogram for bin means
    axs[1].hist(flattened_means, bins=len(bins), color='orange', edgecolor='black')
    axs[1].set_title("Histogram of Bin Means")
    axs[1].set_ylabel("Frequency")

    # Histogram for bin boundaries
    axs[2].hist(flattened_boundaries, bins=len(bins), color='green', edgecolor='black')
    axs[2].set_title("Histogram of Bin Boundaries")
    axs[2].set_xlabel("Value")
    axs[2].set_ylabel("Frequency")

    plt.tight_layout()

    # Bold visualization section title
    print("\n\033[1mVisualization Graph:\033[0m")
    plt.show()
# ====== Main Program ======

# User input
print("\033[1mProgram Input:\033[0m")
data = list(map(float, input("Enter the data values separated by spaces: ").split()))
num_bins = int(input("Enter the number of bins: "))

# Equi-depth bins
bins = equi_depth_binning(data, num_bins)

# Transformed bins
mean_bins = bin_means(bins)
boundary_bins = bin_boundaries(bins)

# Bold program output title
print("\n\033[1mProgram Output:\033[0m")
print("Original Data:", data)
print("Equi-depth Bins:", bins)
print("Bins with Means:", mean_bins)
print("Bins with Boundaries:", boundary_bins)

# Plot separate histograms
plot_separate_histograms(data, bins, mean_bins, boundary_bins)
```

```
Program Input:
Enter the data values separated by spaces: 1 2 4 6 8 10 12 15 20 4
Enter the number of bins: 4

Program Output:
Original Data: [1.0, 2.0, 4.0, 6.0, 8.0, 10.0, 12.0, 15.0, 20.0, 4.0]
Equi-depth Bins: [[1.0, 2.0], [4.0, 4.0], [6.0, 8.0], [10.0, 12.0, 15.0, 20.0]]
Bins with Means: [[np.float64(1.5), np.float64(1.5)], [np.float64(4.0), np.float64(4.0)], [np.float64(7.0), np.float64(7.0)], [np.float64(14.25), np.float64(14.25), np.float64(14.25), np.float64(14.25)]]
Bins with Boundaries: [[1.0, 2.0], [4.0, 4.0], [6.0, 8.0], [10.0, 10.0, 20.0, 20.0]]
```

Visualization Graph:



Review Questions:

Q1: How does the process of Equal Width Discretization transform continuous data into discrete categories, and what role does the choice of the number of bins play in the outcome?

Equal Width Discretization divides the range of the continuous data into equal intervals (bins), with each bin having the same width. The process transforms the continuous values into categorical values based on which bin they fall into. The **choice of the number of bins** plays a critical role in the outcome:

- A **small number of bins** may oversimplify the data, losing important details.
- A **large number of bins** may overfit the data, resulting in too many categories that could be difficult to interpret. The appropriate number of bins should balance between representing the data's complexity and simplifying it for analysis.

Q2: What insights can be derived from the histogram of the original continuous data, and how does it help in understanding the distribution of the data after discretization?

The **histogram of the original continuous data** provides insights into the distribution of the data, showing:

- **Central tendency**: Where the data tends to cluster (e.g., mean, median).
- **Spread**: How wide or narrow the data distribution is.
- **Skewness**: Whether the data is skewed to the left or right.
- **Outliers**: Extreme values that fall outside the typical range of the data. By comparing the histogram of the original data with the histogram of the discretized data, we can assess how well the discretization preserves the original distribution and which features may have been lost or simplified.

Q3: What are the potential advantages and limitations of using Equal Width Discretization for continuous data, and how can the choice of bin width affect the analysis of the data?

Advantages:

- **Simplicity**: Easy to understand and implement.
- **Efficiency**: Suitable for large datasets as it does not require sophisticated algorithms.
- **Consistency**: Equal-width discretization applies the same bin width across the entire data range, making it straightforward to interpret.

Limitations:

- **Insensitive to data distribution**: It does not account for how data is distributed. If data is not uniformly distributed, some bins may have too many or too few data points, leading to poor representation of the data.
- **Potential loss of information**: It might fail to capture important patterns, especially when the data has significant variations in different parts of the range.

The **choice of bin width** can heavily affect the results:

- **Narrower bins** might lead to more categories and a finer representation, but they could introduce noise.
- **Wider bins** might oversimplify the data and miss important nuances.

Conclusion:-

Equal Width Discretization is a simple method for converting continuous data into discrete categories by dividing the data range into equal intervals. It helps simplify complex data but may lose important details if the data is not uniformly distributed. The choice of bin width is crucial: too few bins may oversimplify the data, while too many can lead to overfitting. By comparing histograms of the original and discretized data, we can evaluate how well the process preserves key data features. This technique is useful for simplifying data but may require adjustments depending on the distribution and analysis goals.

GITHUB LINK : https://github.com/Anugrah0619/DWM.git