Name: Pratham Yadav
PRN: 2020BTEIT00006
Sub: Data Mining Lab

Walchand College of Engineering, Sangli.

Experiment no: 3

Title: Perform Binning of data

Theory: Data binning, also referred to as data bucketing is a preprocessing technique employed to mitigate the influence of minor observational errors in data. The process entails segmenting original data values into compact intervals called bins, and subsequently, they are substituted with a representative value computed for the specific bin. This action imparts a smoothing effect on the input data potentially curbing the risk of overfitting, especially when dealing with smaller datasets.

The categorization of data into bins follows two primary methods.

Equal Frequency Binning:

The bins are designed to contain an equivalent number of observations. This ensures that the data is distributed uniformly across the bins.

Equal width Binning

The bins are established to have uniform width. Each bin's range is defined as [min + w], [min + 2w] and so forth, where 'w' stands for (max - min) divided by the number of bins.

Formula:

For equal frequency binning:

bin_size = no_data / no_bins

For equal width binning

w = (max_ele − min_ele) / no_bins

size of each bin

[min + w], [min + 2w], ---- [min + 2w]

where

bin_size : size of the bin
no_data : number of data
no_bins : number of bins to make
w : width / range of data each bin can have
max_ele : maximum element in data
min_ele : minimum element in data

Algorithm:

1. Start
2. Import necessary libraries
3. Define function for Equal Frequency Binning and Equal width Binning.
4. Define a function to read data from a csv file
5. Define a function to write data from a csv file
6. In the main function
   - 6.1 Read from "input.csv"
   - 6.2 Prompt user to choose a binning method and the number of bins.
   - 6.3 If Equal frequency is chosen:
     - Perform equifreq. binning and write output to "output_equifreq.csv".
   - 6.4 If Equal width is chosen:
     - Perform equiwidth binning and write output to "output_equiwidth.csv"
   - 6.5 Display an error message if an invalid method is chosen.

Example:

Consider an array
am = [5, 10, 11, 13, 15, 35, 50]

Now am will be binned using both method
a) Equal Frequency
b) Equal width.

A) Equal Frequency

am_size = 6 · n = 6/3 = 2
· bin = 3

Bin-1 : [5, 10]
Bin-2 : [11, 13]
Bin-3 : [15, 35]

B) Equal width

am_size = 6 bin = 3
max_ele = 35 min_ele = 5
w = (35-5)/3 = 30/3 = 10

Bin1 : 5, 10, 11, 13
Bin2 : 15
Bin3 : 35

Conclusion :
Binning is a useful technique for transforming continous data into discrete categories, making it easier to analyze and visualize.

Code:

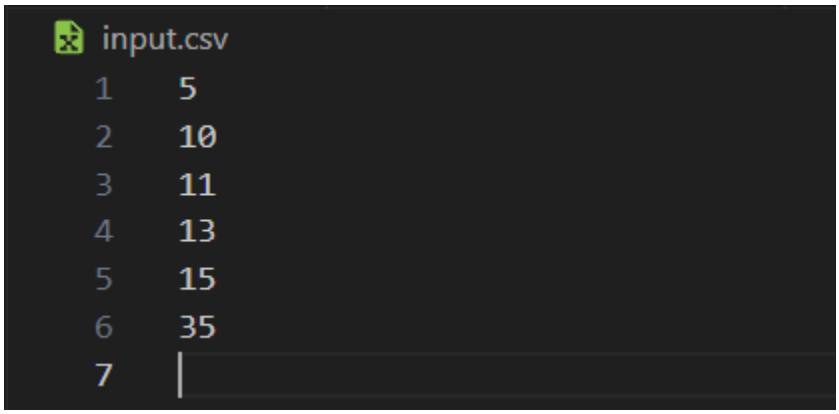
```
#include <iostream>
#include <fstream>
#include <vector>
#include <sstream>
#include <climits>

using namespace std;

// Equal Frequency Binning
vector<vector<int>> equifreq(vector<int> arr1, int m)
{
  int a = arr1.size();
  int n = a / m;
  vector<vector<int>> bins;
  for (int i = 0; i < m; i++)
  {
    vector<int> bin;
    for (int j = i * n; j < (i + 1) * n; j++)
    {
      if (j >= a)
      {
        break;
      }
      bin.push_back(arr1[j]);
    }
    bins.push_back(bin);
  }
  return bins;
}

// Equal Width Binning
vector<vector<int>> equiwidth(vector<int> arr1, int m)
{
  int a = arr1.size();

  int max_ele = INT_MIN;
  int min_ele = INT_MAX;
```

```
  for (int i = 0; i < arr1.size(); i++)
  {
   max_ele = max(max_ele, arr1[i]);
   min_ele = min(min_ele, arr1[i]);
  }

  int w = (max_ele - min_ele) / m;
  int min1 = min_ele;

  vector<int> arr;
  for (int i = 0; i < m + 1; i++)
  {
   arr.push_back(min1 + w * i);
  }

  vector<vector<int>> arri;
  for (int i = 0; i < m; i++)
  {
   vector<int> temp;
   for (int j : arr1)
   {
    if (j >= arr[i] && j <= arr[i + 1])
    {
     temp.push_back(j);
    }
   }
   arri.push_back(temp);
  }
  return arri;
}

// Read data from CSV
vector<int> readCSV(string filename)
{
  ifstream inputFile(filename);
  vector<int> data;
  string line, value;
  while (getline(inputFile, line))
  {
```

```
    stringstream ss(line);
    while (getline(ss, value, ','))
    {
      data.push_back(stoi(value));
    }
  }
  inputFile.close();
  return data;
}

// Write binning outputs to CSV
void writeCSV(string filename, vector<vector<int>> bins)
{
  ofstream outputFile(filename);
  for (int i = 0; i < bins.size(); i++)
  {
    outputFile << "Bin " << i + 1 << ",";
    for (int num : bins[i])
    {
      outputFile << num << ",";
    }
    outputFile << "\n";
  }
  outputFile.close();
}

int main()
{
  vector<int> data = readCSV("input.csv");
  int m;

  int method;
  cout << "Choose binning method: " << endl;
  cout << "1. Equal Frequency Binning" << endl;
  cout << "2. Equal Width Binning" << endl;
  cout << "\nEnter method number: ";
  cin >> method;
  cout << "\nEnter number of bins: ";
  cin >> m;
```

```
  if (method == 1)
  {
    vector<vector<int>> freqBins = equifreq(data, m);
    writeCSV("output_equifreq.csv", freqBins);
  }
  else if (method == 2)
  {
    vector<vector<int>> widthBins = equiwidth(data, m);
    writeCSV("output_equiwidth.csv", widthBins);
  }
  else
  {
    cout << "Invalid method choice." << endl;
  }

  return 0;
}
```

I/P



O/P

a) Equal frequency

output_equifreq.csv

```
1  Bin 1,5,10,
2  Bin 2,11,13,
3  Bin 3,15,35,
4
```

b) Equal Width

output_equiwidth.csv

```
1  Bin 1,5,10,11,13,
2  Bin 2,15,
3  Bin 3,35,
4
```