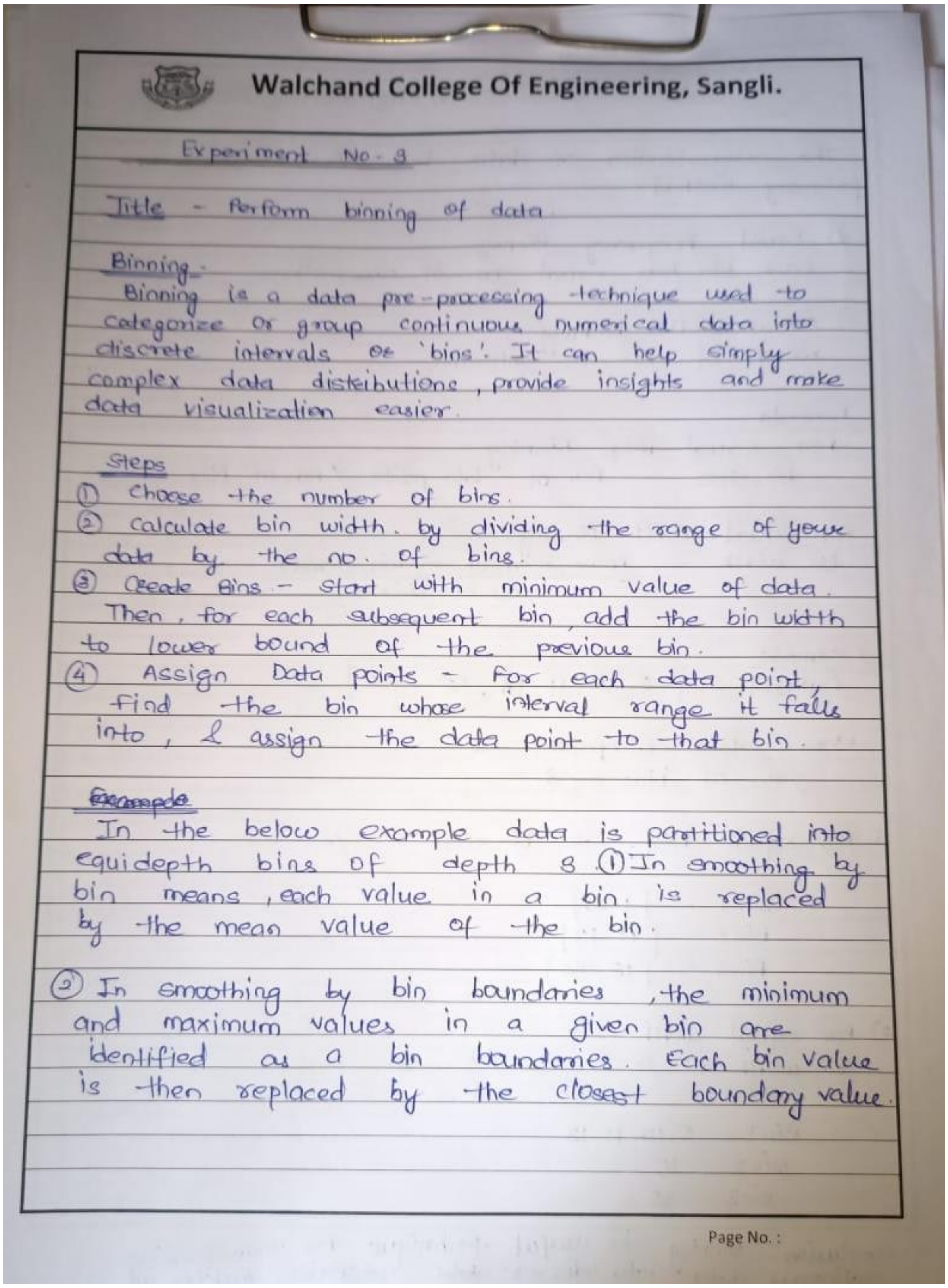
Assignment 3

Title- Perform Binning of data.



Walchand College Of Engineering, Sangli.

Experiment No - 3

Title - Perform binning of data

Binning -

Binning is a data pre-processing technique used to categorize or group continuous numerical data into discrete intervals or 'bins'. It can help simply complex data distributions, provide insights and make data visualization easier.

Steps

① Choose the number of bins.

② Calculate bin width by dividing the range of your data by the no. of bins.

③ Create Bins - Start with minimum value of data. Then, for each subsequent bin, add the bin width to lower bound of the previous bin.

④ Assign Data points - For each data point, find the bin whose interval range it falls into, & assign the data point to that bin.

Example

In the below example data is partitioned into equidepth bins of depth 3 ① In smoothing by bin means, each value in a bin is replaced by the mean value of the bin.

② In smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as a bin boundaries. Each bin value is then replaced by the closest boundary value.

Page No. :

The categorization of data into bins follows two primary methods

1) Equal frequency Binning
Each bin has equal no. of observations.

2) Equal width binining
width of bin is uniform.

Formula

For equal freq. binning
bin_size = no. of data points / no. of bins.

for equal width binning

$$\text{bin\_width} = \frac{\text{max\_element} - \text{min\_element}}{\text{no. of bins}}$$

Example

Dataset - [5, 10, 11, 13, 15, 35]

Here - total datapoints = 6
NO. of bins = 3

1) For equal freq. ⇒ bin.size = 6/3 = 2

∴ Bin 1 = [5,10]
Bin2 = [11, 13]
Bin3 = [15, 35]

2) for equal width

$$\text{bin-width} = \frac{35-5}{3} = \frac{30}{3} = 10.$$

∴ Bin 1 = 5, 10, 11, 13
Bin 2 = 15
Bin3 = 35

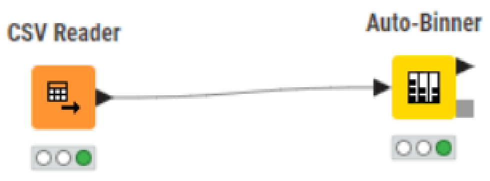
Conclusion - Binning is useful technique for transforming contineous data into discrete ~~data~~ categories, making it easier to analyze & visualize.

Code

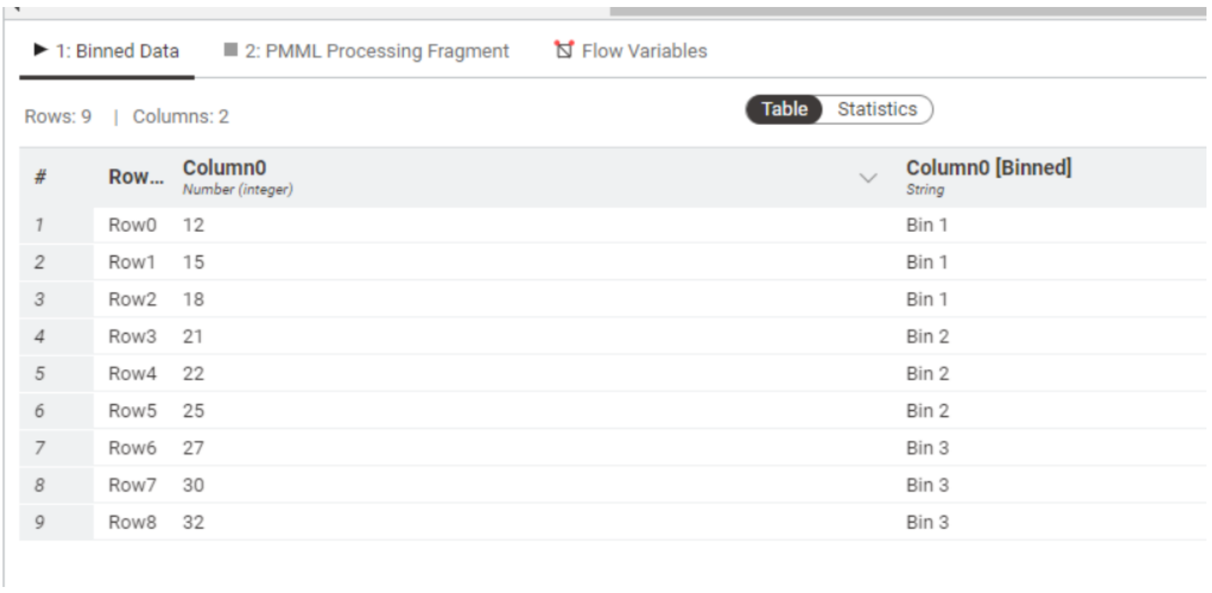
```cpp
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <climits>

using namespace std;

// Equal Frequency Binning
vector<vector<int>> equifreq(vector<int> data, int m)
{
  int a = data.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(data[j]);
    }
    bins.push_back(bin);
  }
  return bins;
}

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

  int max_ele = INT_MIN;
  int min_ele = INT_MAX;

  for (int i = 0; i < data.size(); i++)
  {
    max_ele = max(max_ele, data[i]);
    min_ele = min(min_ele, data[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 : data)
    {
      if (j >= arr[i] && j <= arr[i + 1])
      {
        temp.push_back(j);
      }
    }
    arri.push_back(temp);
  }
  return arri;
}

// Reading 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])
```

```cpp
    {
      outputFile << num << ",";
    }
    outputFile << "\n";
  }
  outputFile.close();
}

int main()
{
  vector<int> data = readCSV("data.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;
}
```

Knime



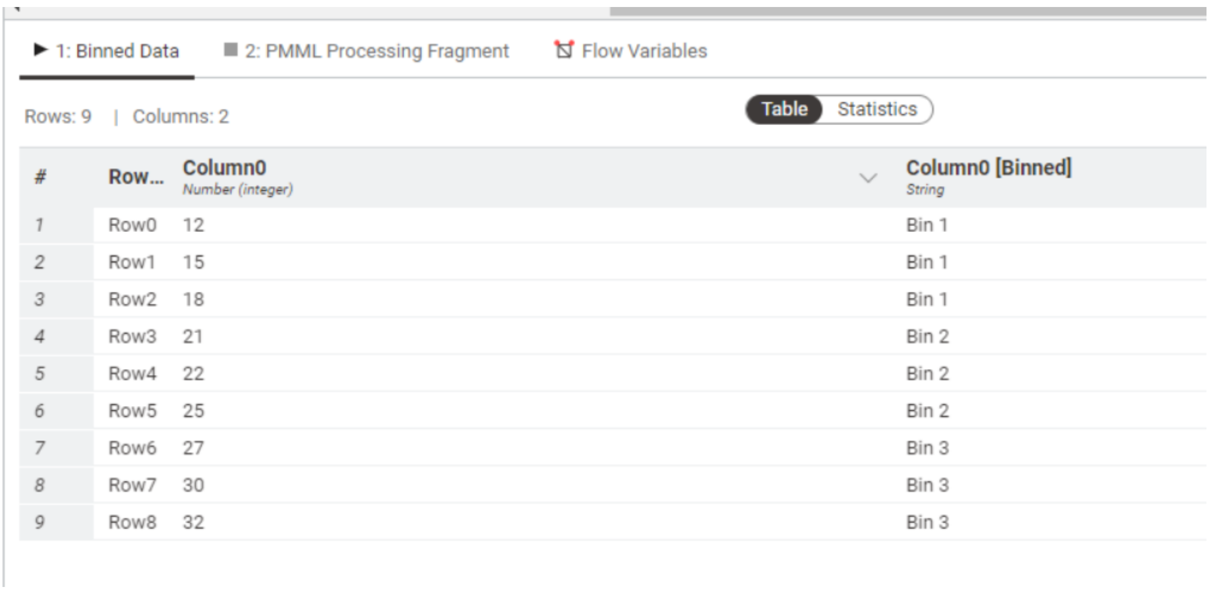
Equi-width binning



Rows: 9 | Columns: 2

#	Row...	Column0 Number (integer)	Column0 [Binned] String
1	Row0	12	Bin 1
2	Row1	15	Bin 1
3	Row2	18	Bin 1
4	Row3	21	Bin 2
5	Row4	22	Bin 2
6	Row5	25	Bin 2
7	Row6	27	Bin 3
8	Row7	30	Bin 3
9	Row8	32	Bin 3

Equi-frequency binning



Rows: 9 | Columns: 2

#	Row...	Column0 Number (integer)	Column0 [Binned] String
1	Row0	12	Bin 1
2	Row1	15	Bin 1
3	Row2	18	Bin 1
4	Row3	21	Bin 2
5	Row4	22	Bin 2
6	Row5	25	Bin 2
7	Row6	27	Bin 3
8	Row7	30	Bin 3
9	Row8	32	Bin 3