Data preprocessing is an important issue for data mining. What are the tasks in data preprocessing?

資料清理 (Data cleaning)

- Missing, Noisy, Inconsistent, Intentional, outliers, repetition.

資料整合 (Data integration)

- Schema integration, Entity identification problem, Different

representations, different scales, Remove redundancies, Detect

inconsistencies (chi squared, covariance, correlation)

資料轉換 (Data transformation and data discretization)

- Normalization

- Discretization

資料精簡 (Data reduction)

- Dimensionality reduction

- Numerosity reduction

- Data compression

Please use “Equal Width” and “Equal Depth” Binning Method to smooth the Dataset {17,42,3,29,12,21,6,29,22} into 3 boxes. Then, please use “Means”,” Median”, and “Boundaries” methods to eliminate the noisy data in final results, respectively.

Sort {3, 6, 12, 17, 21, 22, 29, 29, 42}

Equal Width:

(42 - 3) / 3 = 13

{3, 6, 12}

{17, 21, 22, 29, 29}

{42}

Means:

{7, 7, 7}

{24, 24, 24, 24,24}

{42}

Median:

{6, 6, 6}

{22, 22, 22, 22, 22}

{42}

Boundaries:

{3, 3, 12}

{17, 17, 17, 29, 29}

{42}

Equal Depth:

{3, 6, 12}

{17, 21, 22}

{29, 29, 42}

Means:

{7, 7, 7}

{20, 20, 20}

{33, 33, 33}

Median:

{6, 6, 6}

{21, 21, 21}

{29, 29, 29}

Boundaries:

{3, 3, 12}

{17, 22, 22}

{29, 29, 42}

a=[17,42,3,29,12,21,6,29,22]

c=sorted(a)

print(c)

b=[]

d=[]

def find\_bin(value, bins):

    for i in range(0, len(bins)):

        if bins[i][0] <= value <= bins[i][1]:

            return i

    return -1

def create\_bins(lower\_bound, width, quantity):

    bins = []

    for low in range(lower\_bound, lower\_bound + quantity\*width + 1, width):

        bins.append((low, low+width))

    return bins

from collections import Counter

bins = create\_bins(lower\_bound=3,

width=13,

quantity=2)

print("bins\_boarder:",bins)

def boarder(bins,b,c,d):

  i=0

  j=0

  print('hi',b)

  print(len(c))

  for i in range(len(c)):

    if b[i]==0:

      if (c[i]-bins[j][0]) <= (bins[j][1]-c[i]):

        d.append(bins[j][0])

      else:

        d.append(bins[j][1])

    elif b[i]==1:

      j=1

      if (c[i]-bins[j][0]) <= (bins[j][1]-c[i]):

        d.append(bins[j][0])

      else:

        d.append(bins[j][1])

    elif b[i]==2:

      j=2

      if (c[i]-bins[j][0]) <= (bins[j][1]-c[i]):

        d.append(bins[j][0])

      else:

        d.append(bins[j][1])

  print(d)

def get\_width(c,b):

  for value in c:

    bin\_index = find\_bin(value, bins)

    b.append(bin\_index)

get\_width(c,b)

boarder(bins,b,c,d)