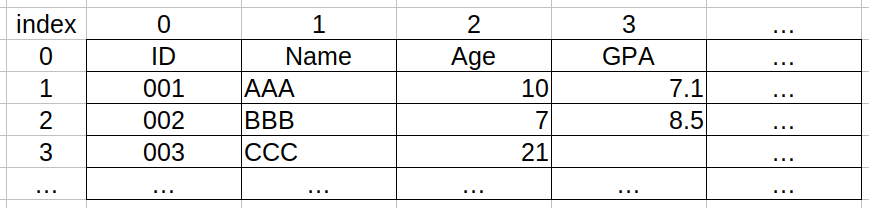
Đây là báo cáo các thuật toán đã sử dụng cho chương trình tiền xử lý dữ liệu đơn giản.

I. Tổ chức dữ liệu:

Tập dữ liệu được biểu diễn dưới dạng ma trận hai chiều. Trong đó, mỗi dòng là một mẫu dữ liệu, còn mỗi là cột là một thuộc tính. Dòng đầu tiên của ma trận là tên của các thuộc tính này.



II. Các thuật toán cho từng thao tác:

a. Min - max normalizing (chuẩn hóa dữ liệu theo phương pháp min – max):

- Phát biểu: chuẩn hóa thuộc tính A của tập dữ liệu M mẫu theo phương pháp min – max

- Input: tập dữ liệu M, thuộc tính A, giá trị newMin và newMax

- Output: tập dữ liệu có thuộc tính A đã được chuẩn hóa

- Thuật toán: mỗi phần tử giá trị x sẽ có giá trị mới là ((x - minA) / (maxA - minA)) \* (newMax - newMin) + newMin

- Mã giả:

minA = min(data[i][A]) with 1 <= i <= len(data) – 1

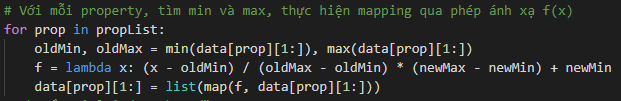
maxA = max(data[i][A]) with 1 <= i <= len(data) - 1

f(x)=((x - minA)/(maxA - minA))\*(newMax - newMin)+newMin

for instance in data:

instance[A] = f(instance[A])

- Cài đặt bằng Python:



b. Z-score normalizing (chuẩn hóa dữ liệu theo phương pháp z-score):

- Phát biểu: chuẩn hóa thuộc tính A của tập dữ liệu M mẫu theo phương pháp z-score

- Input: tập dữ liệu M, thuộc tính A

- Output: tập dữ liệu đã có thuộc tính A đã được chuẩn hóa

- Thuật toán: mỗi phần tử giá trị x sẽ có giá trị mới là (x - meanA) / deva, với meanA là trung bình, devA là độ lệch chuẩn

- Mã giả:

meanA = average(data[i][A]) with 1 <= i <= len(data) – 1

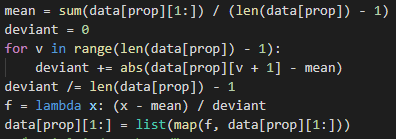
devA = sum(abs(data[i][A] - meanA)) / (len(data) – 1)

f(x) = (x - meanA) / devA

for instance in data:

instance[A] = f(instance[A])

- Cài đặt bằng Python:



c. Equal - width binning (rời rạc hóa dữ liệu bằng phương pháp chia giỏ theo độ rộng):

- Phát biểu: chia các mẫu vào N giỏ sao cho mỗi giỏ có độ rộng bằng nhau

- Input: tập dữ liệu M, thuộc tính A, số giỏ N

- Output: tập dữ liệu đã được thêm thuộc tính nhãn giỏ (mỗi mẫu sẽ được đánh nhãn giỏ)

- Thuật toán: ta có độ rộng mỗi giỏ là (maxA - minA)/N, mỗi phần tử có giá trị x sẽ nằm ở giỏ thứ (x - minA)//N (làm tròn xuống)

- Mã giả:

minA = min(data[i][A]) with 1 <= i <= len(data) – 1

maxA = max(data[i][A]) with 1 <= i <= len(data) – 1

width = (maxA - minA) / N

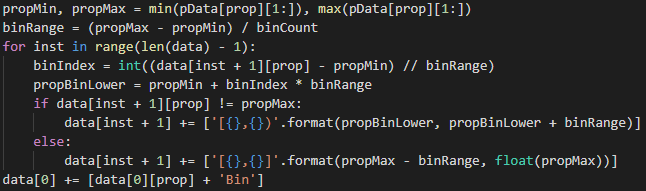
f(x) = x // width

for instance in data:

instance.add\_prop(f(instance[A])]

data[0].add\_prop(“binNumber”)

- Cài đặt bằng Python:



d. Equal - depth binning (rời rạc hóa dữ liệu bằng phương pháp chia giỏ theo độ sâu):

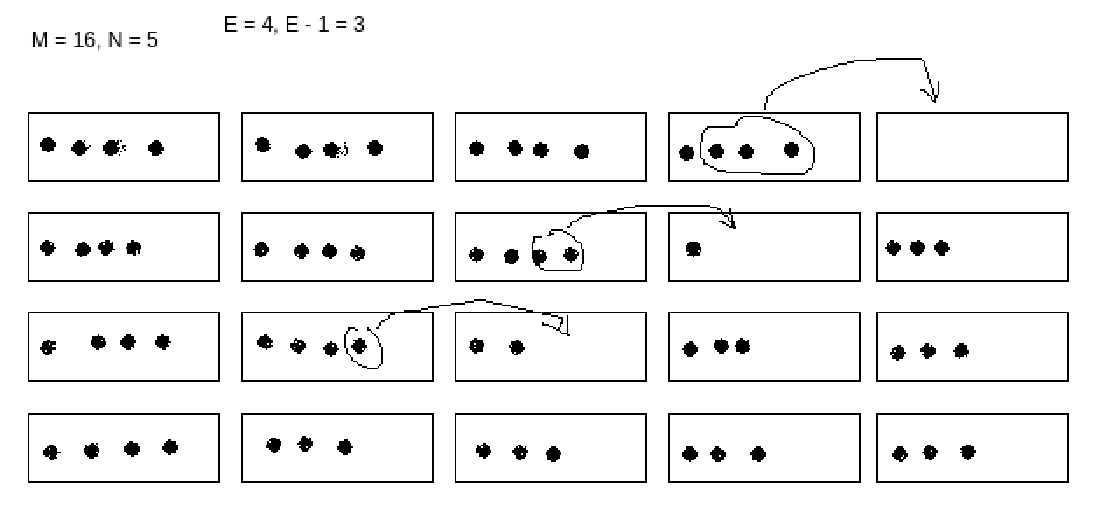
- Phát biểu: chia các mẫu vào N giỏ sao cho mỗi giỏ có số mẫu xấp xỉ ngang bằng nhau

- Input: tập dữ liệu M, thuộc tính A, số giỏ N

- Output: tập dữ liệu đã được thêm thuộc tính nhãn giỏ (mỗi mẫu sẽ được đánh nhãn giỏ)

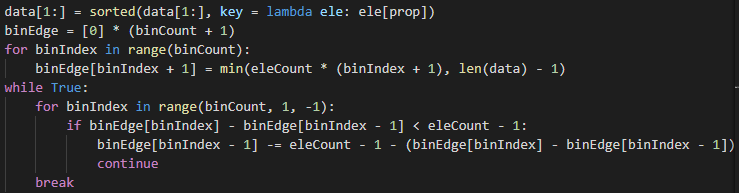
- Thuật toán: dễ thấy mỗi giỏ chỉ có thể có E hoặc E - 1 mẫu với E = M // N + 1, đầu tiên phân tối đa các mẫu vào những giỏ đầu tiên, sau đó duyệt từ giỏ cuối cùng ngược lên đầu, giỏ nào chưa đủ số mẫu, tiến hành lấy các mẫu trong giỏ gần nhất bên trái và bỏ vào đến khi thỏa điều kiện.

- Mô phỏng:



- Cải tiến: giữ nguyên vị trí các phần tử, xem như các vách ngăn của các giỏ có thể thay đổi được => tìm vị trí các vách ngăn

- Cài đặt:



e. Remove missing properties instances (xóa các mẫu bị thiếu thông tin):

- Phát biểu: xóa các mẫu bị thiếu thông tin thuộc tính A trong tập M mẫu

- Input: tập dữ liệu M, thuộc tính A

- Output: tập dữ liệu đã được xóa các mẫu bị thiếu thông tin

- Thuật toán: duyệt từng mẫu, nếu mẫu đó thiếu thông tin thuộc tính A, xóa mẫu đó khỏi tập mẫu (ở đây, do bản chất vòng lặp trong Python, ta duyệt từng mẫu, nếu mẫu nào không bị thiếu thông tin thuộc tính A, ta copy mẫu đó sang tập mẫu kết quả pData)

- Mã giả:

for instance in data:

if instance[A] == NULL:

data.remove(instance)

hoặc

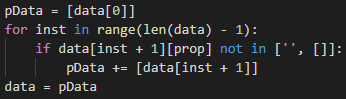
pData = []

for instance in data:

if instance[A] != NULL:

pData.add\_instance(instance)

- Cài đặt bằng Python:



f. Fill in missing properties (điền các thông tin bị thiếu):

- Phát biểu: điền thông tin thuộc tính A cho các mẫu bị thiếu thông tin này (điền mean nếu A là thuộc tính số, ngược lại, điền mode)

- Input: tập dữ liệu M, thuộc tính A

- Output: tập dữ liệu đã được điền đầy đủ thông tin thuộc tính A

- Thuật toán: tính mean hoặc mode của thuộc tính A (tùy theo kiểu dữ liệu, rồi duyệt qua từng mẫu, nếu mẫu này trống thông tin thuộc tính A thì điền vào

- Mã giả:

if isNumerical(A):

meanA = sum(data[i][A]) with 1 <= i <= len(data) – 1

for instance in data:

if instance[A] == NULL:

instance[A] = mean

else:

countTable[value] = count(data[i][A] == value) with with 1 <= i <= len(data) – 1

modeA = countable.getkeys(max(countTable))

for instance in data:

if instance[A] == NULL:

instance[A] = mean

- Cài đặt bằng Python:

