Bài tập 01

Tiền xử lý dữ liệu với Weka

Thông tin cá nhân

Họ và tên	Vũ Lê Thế Anh	Nguyễn Lê Hồng Hạnh
MSSV	1612838	1612849
Email	{1612838, 1612849}@student.hcmus.edu.vn	
SĐT	0961565087	0902719551

Yêu cầu bài tập

STT	Yêu cầu	Hoàn thành
1	Tích hợp dữ liệu	100%
2	Tóm tắt mô tả dữ liệu	100%
3	Chọn lọc dữ liệu	100%
4	Làm sạch dữ liệu	100%
5	Chuyển đổi dữ liệu	100%
6	Rút gọn dữ liệu	100%

Mục lục

1 Tích hợp dữ liệu (Integration)

a. Tích hợp dữ liệu (data integration) là là sự hợp nhất dữ liệu từ nhiều nguồn dữ liệu khác nhau để xử lý. Việc tích hợp dữ liệu chính xác sẽ giúp làm giảm những dữ liệu dư thừa, mâu thuẫn giữa những nguồn dữ liệu ban đầu.

b. Nhận diện thực thể (entity identification) là vấn đề khi có hai tên gọi khác nhau cho cùng một đối tượng vật thể (hoặc hai đối tượng vật thể tương tự nhau). Ví dụ: "sex" và "gender" cùng chỉ giới tính, hay "male" và "M" cùng chỉ giới tính nam,. . . Bộ dữ liệu đầu đề có tồn tại vấn đề này.

Cụ thể, ở tập dữ liệu của người Hungary (sau này sẽ gọi là dữ liệu `h`) thì dữ liệu liên quan đến "loại cơn đau ở ngực" (chest pain type) được đặt tên thuộc tính là `cp`, trong khi trong tập dữ liệu của người Cleveland (sau này gọi tắt là dữ liệu `c`) thì được đặt tên là `chest_pain`. Điều này gây ra sự không thống nhất giữa 2 tập dữ liệu, và tập kết quả sau khi tích hợp sẽ phát sinh thêm một thuộc tính dư thừa không mong muốn. Do đó, chúng ta sẽ giải quyết vấn đề này bằng cách đổi tên thuộc tính trên của một trong 2 tập dữ liệu. Ở đây, chúng tôi chọn cách đổi tên thuộc tính của dữ liệu `h` thành `cp` (cho giống với dữ liệu `c`).

Ngoài ra, ở cả hai tập dữ liệu, thuộc tính `num` theo đặc tả chỉ nhận 2 giá trị là `'<50'` và `'>50'` nhưng dữ liệu lại tồn tại 5 giá trị là `'<50'` và các giá trị `'>50_x'` (x = 1, 2, 3, 4). Có khả năng các giá trị `'>50_x'` đều cùng mang ý nghĩa là `'>50'`. Ở đây, chúng tôi đề xuất thay tất cả `'>50_x'` thành `'>50'`.

c. Dư thừa (data redundancy) là vấn đề khi các thuộc tính có liên hệ với nhau. Ví dụ, hai thuộc tính năm sinh và tuổi có liên hệ suy ra từ nhau (tuổi = năm sinh - năm hiện tại). Ngoài ra, dư thừa cũng có thể do các vấn đề về đặt tên, không nhất quán giá trị,. . . Về vấn đề này, ta có thể giải quyết bằng cách bỏ đi 1 trong 2 thuộc tính dư thừa.

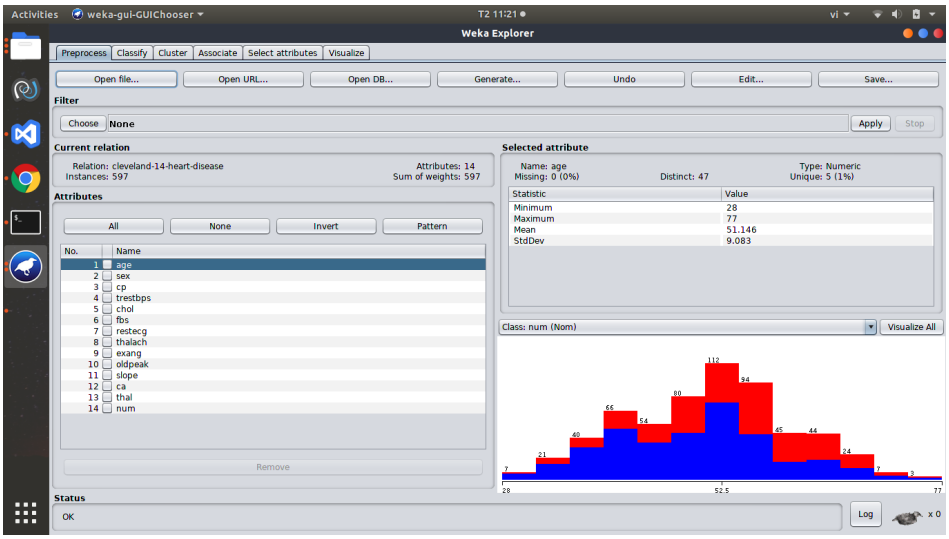
Trong bộ dữ liệu đầu đề, sau khi đã giải quyết vấn đề nhận diện thực thể như trên, chúng tôi không thấy có vấn đề dư thừa dữ liệu.

d. Mâu thuẫn giá trị dữ liệu (data value conflict) xảy ra khi hai thuộc tính có khác biệt về thứ nguyên hoặc cách thống kê khác nhau. Ví dụ, chiều cao ở một tập dữ liệu tính bằng m nhưng lại tính bằng cm ở một tập dữ liệu khác. Hay điểm của một sinh viên có thể theo thang 10 hay thang 4 tùy trường đại học. Vấn đề này đôi lúc sẽ rất khó để giải quyết, với ví dụ m và cm, ta có thể thực hiện đổi đơn vị, nhưng với thang 10 hay thang 4, có khả năng phép chuyển đổi này là không thể (điểm 9.5 và 10.0 trong thang 10 điểm có thể cùng tương ứng điểm 4.0 trong thang 4 điểm, tức không có hàm ngược).

Sau khi kiểm tra đặc tả dữ liệu, 2 bộ dữ liệu đầu đề không tồn tại vấn đề này.

e. Sau khi tích hợp dữ liệu, ta thu được một bộ dữ liệu gồm 597 mẫu và 14 thuộc tính. Dữ liệu này được lưu trong tập tin `heart-intergration.arff` đính kèm.

f. Hình 1 cho thấy màn hình cửa sổ Explorer khi đã nạp lên bộ dữ liệu đã tích hợp.



Hình 1: Màn hình Explorer sau khi tích hợp 2 tập dữ liệu.

2 Tóm tắt mô tả dữ liệu (Descriptive data summarization)

a. Thuộc tính age có các giá trị:

- Trung bình: 51.146
- Độ lệch chuẩn: 9.083
- Giá trị nhỏ nhất: 28
- Giá trị lớn nhất: 77

b. Five-number summary của thuộc tính age là:

- min: 28
- Q1: 44
- median: 52
- Q3: 58
- max: 77

Trong Weka không cung cấp thông tin thống kê về 5 con số này.

c.

- Thuộc tính số (numeric): `age`, `trestbps`, `chol`, `thalach`, `oldpeak`, `ca`.
- Thuộc tính rời rạc (nomial): `sex`, `cp`, `restecg`, `exang`, `slope`, `thal`, `fbs`, `num`.
- Thuộc tính có thứ tự là các thuộc tính tuy rời rạc nhưng các giá trị có ý nghĩa có thể sắp xếp được, ví dụ với `fbs` thì 1 có nghĩa là lớn hơn 120 còn 0 thì ngược lại, tức giá trị 1 đứng trước giá trị 0 trong quan hệ lớn hơn. Trong dữ liệu trên, các thuộc tính thứ tự là: `fbs`, `num`.

d. Đồ thị trong cửa sổ Explorer (xem Hình 2) cho biết sự phân bố giá trị của thuộc tính.

Đối với các thuộc tính dạng số (numeric), đồ thị này sẽ là histogram: chia miền giá trị $[min, max]$ thành nhiều miền con $[a_i, b_i]$ với kích thước xấp xỉ nhau. Ứng với mỗi miền con, ta đếm số lượng mẫu có giá trị thuộc tính nằm trong miền. Cuối cùng, ta biểu diễn dưới dạng đồ thị cột.

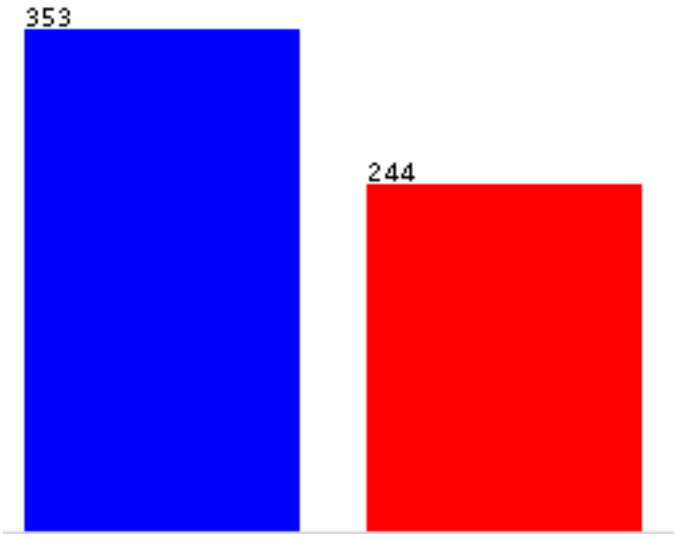
Đối với các thuộc tính rời rạc (nomial), ứng với mỗi giá trị của thuộc tính, ta đếm số lượng mẫu có giá trị đó. Tương tự trên, ta cũng biểu diễn thành đồ thị cột.

Ngoài ra, đồ thị còn biểu diễn tương đối số lượng mẫu ứng với từng nhãn. Thuộc tính làm nhãn là do người dùng lựa chọn. Mỗi cột sẽ gồm nhiều cột màu xếp chồng, mỗi màu tương ứng với một nhãn.

Như vậy, nhìn vào đồ thị, với mỗi giá trị của thuộc tính, ta biết được số lượng mẫu với có giá trị đó và biết tương đối trong đó có bao nhiêu mẫu có nhãn tương ứng.

Ví dụ, trong hình 2, cũng như trong các đồ thị khác của tập dữ liệu này, màu đỏ ứng với các mẫu có nhãn (`num`) là `'>50'`, màu xanh ứng với các mẫu có nhãn là `'<50'`.

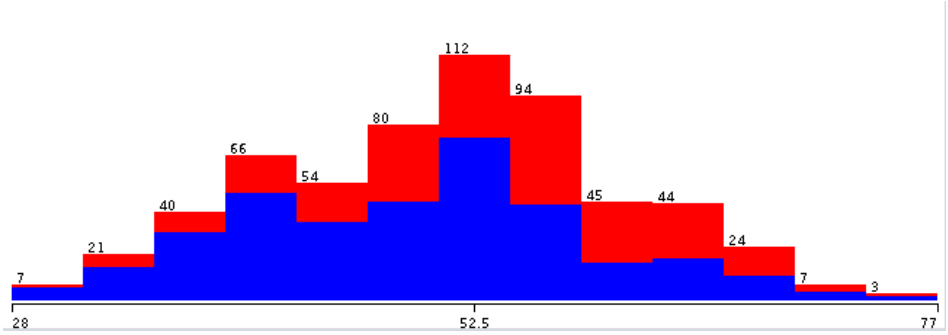
Dựa trên những phân tích trên, có thể đặt tên đồ thị là: *Đồ thị phân bố giá trị của thuộc tính theo nhãn*.



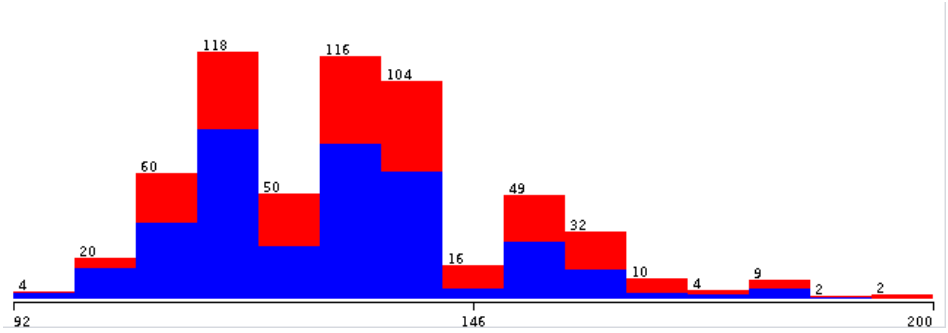
Hình 2: Đồ thị phân bố giá trị của thuộc tính `num`

e. Xem xét các thuộc tính khác của tập dữ liệu dưới dạng đồ thị:

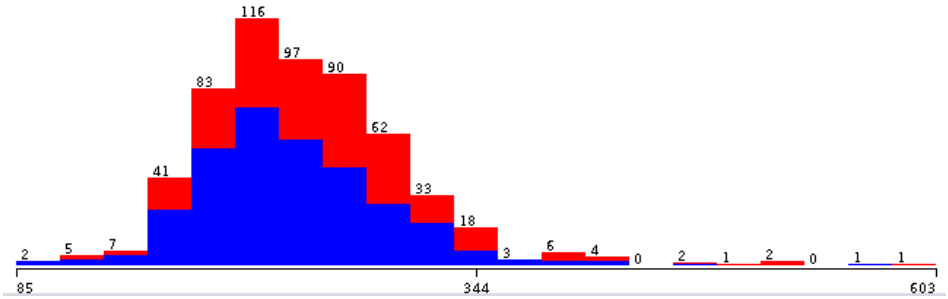
* Nhóm thuộc tính dạng số (numeric):
  - Thuộc tính `age`:
  - Thuộc tính `trestbps`:
  - Thuộc tính `chol`:
  - Thuộc tính `thalach`:



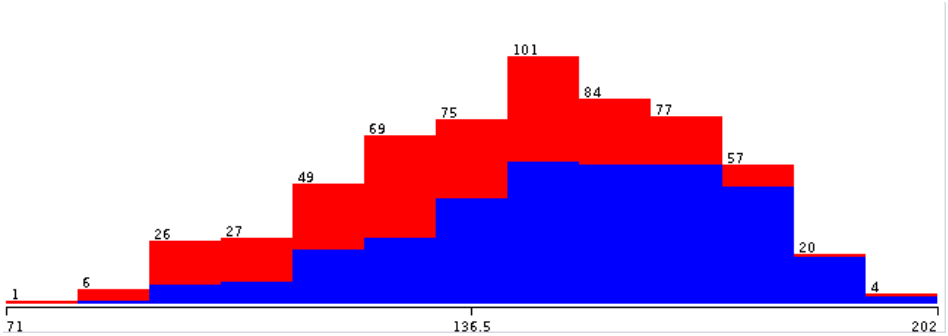
Hình 3: Đồ thị phân bố giá trị của thuộc tính `age`.



Hình 4: Đồ thị phân bố giá trị của thuộc tính `trestbps`.



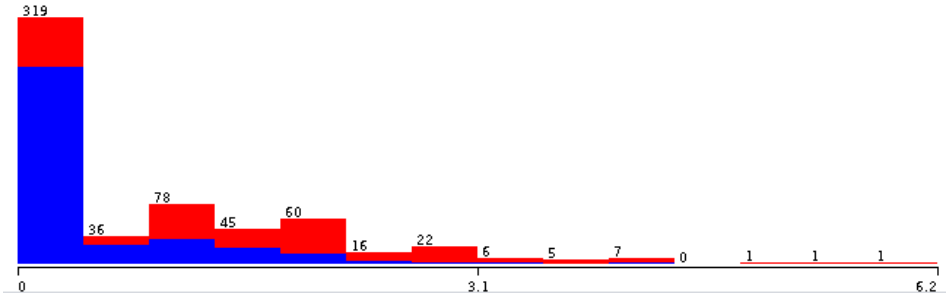
Hình 5: Đồ thị phân bố giá trị của thuộc tính `chol`.



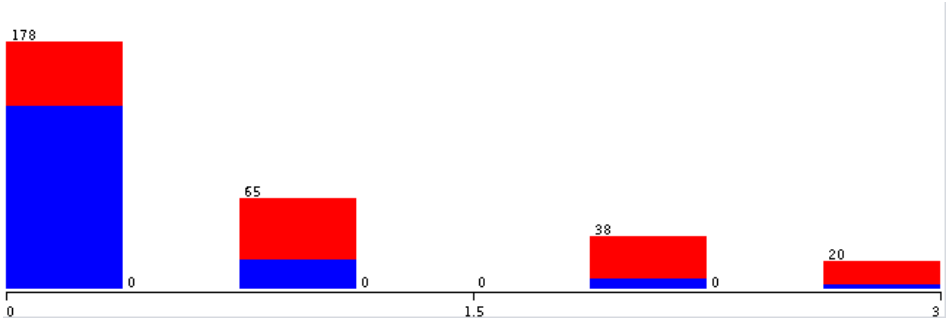
Hình 6: Đồ thị phân bố giá trị của thuộc tính `thalach`.

– Thuộc tính `oldpeak`:

– Thuộc tính `ca`:

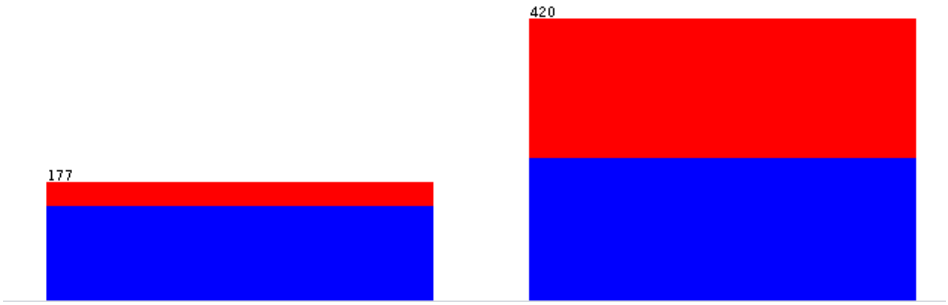


Hình 7: Đồ thị phân bố giá trị của thuộc tính `oldpeak`.



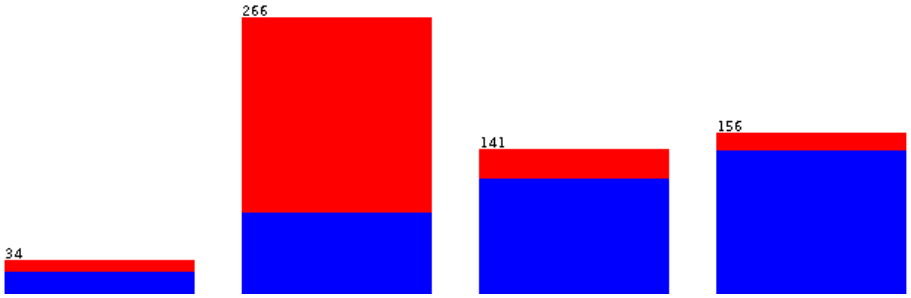
Hình 8: Đồ thị phân bố giá trị của thuộc tính `ca`.

* Nhóm thuộc tính dạng rời rạc (nomial):
  - Thuộc tính `sex`:



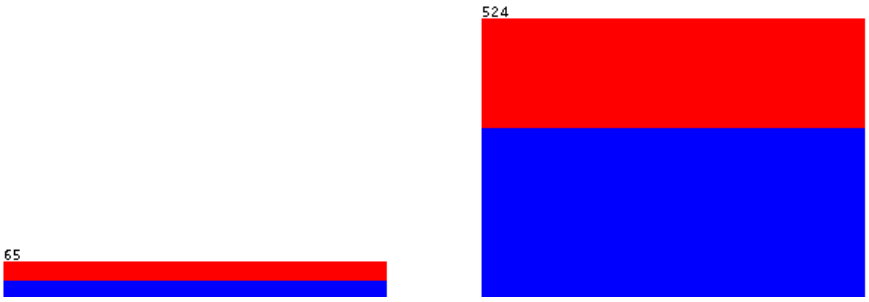
Hình 9: Đồ thị phân bố giá trị của thuộc tính `sex`.

– Thuộc tính cp:



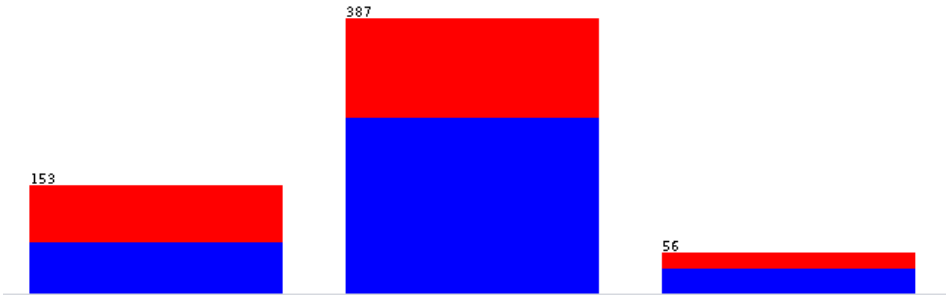
Hình 10: Đồ thị phân bố giá trị của thuộc tính cp.

– Thuộc tính fps:



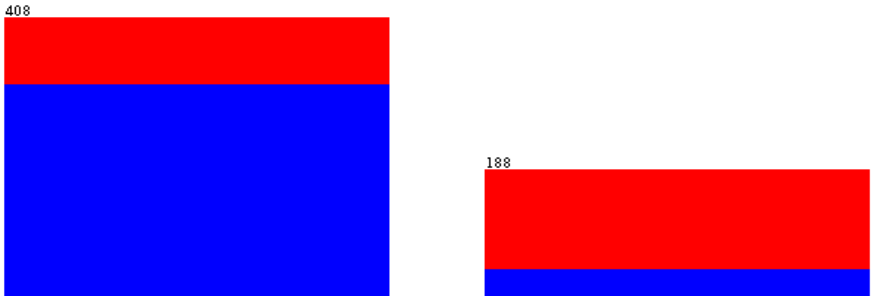
Hình 11: Đồ thị phân bố giá trị của thuộc tính fbs.

– Thuộc tính restecq:



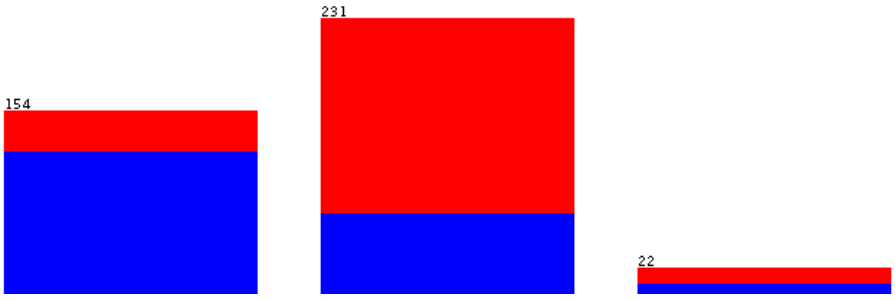
Hình 12: Đồ thị phân bố giá trị của thuộc tính restecq.

– Thuộc tính `exang`:



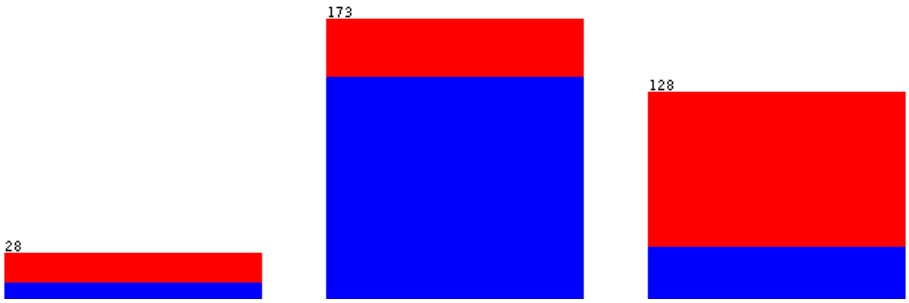
Hình 13: Đồ thị phân bố giá trị của thuộc tính `exang`.

– Thuộc tính `slope`:



Hình 14: Đồ thị phân bố giá trị của thuộc tính `slope`.

– Thuộc tính `thal`:



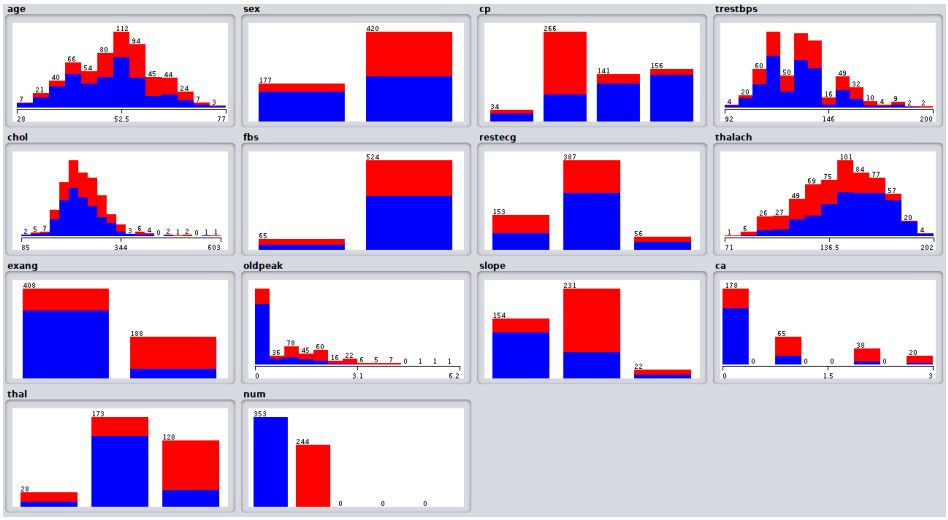
Hình 15: Đồ thị phân bố giá trị của thuộc tính `thal`.

f. Khi bấm nút Visualize All, ta thu được kết quả như Hình 16. Từ đó, ta có thể thấy rằng:

- Các thuộc tính `chol, oldpeak, và ca` có những giá trị 0 có vẻ vô lý, có thể những thuộc tính này có nhiều giá trị trống.
- Các thuộc tính `age, chol, thalach` dường như tuân theo phân phối chuẩn.

g. Các đồ thị này là biểu đồ phân tán (scatter plot).

Trong tab Visualize, các thuộc tính dường như ảnh hưởng đến bệnh tim:



Hình 16: Đồ thị phân bố giá trị của thuộc tính `all`.

- Thuộc tính `sex`: có 1 (trong 2) giá trị có số lượng mẫu màu xanh nhiều hơn đáng kể so với màu đỏ.
- Thuộc tính `cp`: có 1 (trong 4) giá trị có số lượng mẫu màu đỏ nhiều hơn đáng kể so với màu xanh, và có 2 (trong 4) giá trị có số lượng mẫu màu xanh nhiều hơn đáng kể so với màu đỏ. Giá trị còn lại khá ít để đưa ra kết luận.
- Thuộc tính `slope`, có 1 (trong 3) giá trị có số lượng mẫu màu xanh nhiều hơn đáng kể so với màu đỏ, và có 1 (trong 3) giá trị có số lượng mẫu màu đỏ nhiều hơn đáng kể so với màu xanh. Gía trị còn lại khá ít để đưa ra kết luận.
- Thuộc tính `thal`, có 1 (trong 3) giá trị có số lượng mẫu màu xanh nhiều hơn đáng kể so với màu đỏ. 2 giá trị còn lại, 1 khá ít để đưa ra kết luận, 1 có số lượng màu đỏ nhiều hơn nhưng không quá chênh lệch.
- Thuộc tính `exang`, có 1 (trong 2) giá trị có số lượng xanh nhiều hơn đáng kể so với đỏ, và có 1 (trong 2) giá trị có số lượng đỏ nhiều hơn đáng kể so với xanh.

Hình 17 là đồ thị của các thuộc tính trên.

Hình 17: Đồ thị của các thuộc tính trong Visualize (thứ tự: `sex, cp, exang, slope, thal`)

Với những nhận xét và đồ thị trên, chúng tôi cho rằng thuộc tính `exang` (hình giữa của Hình 17) có thể cho một hàm dự đoán bệnh tim tốt:

$$Y = (\text{exang} == \text{'yes'})$$

h. Sau khi nghiên cứu bộ dữ liệu, chúng tôi không tìm thấy tương quan giữa các thuộc tính với nhau. Cụ thể, các điểm trong scatter plot đều được phân tán khá rộng chứ không thể hiện xu hướng chung nào.

3 Chọn lọc dữ liệu (Selection)

a. Trước khi xử lý, tập dữ liệu ban đầu có 76 thuộc tính (đã bao gồm thuộc tính nhãn `num`).

b. Một số phương pháp Weka sử dụng để đánh giá thuộc tính để chọn ra các thuộc tính tiêu biểu:

- **CorrelationAttributeEval**: Đánh giá giá trị của một thuộc tính bằng cách đo lường mối tương quan giữa thuộc tính đó và nhãn.
- **ClassifierAttributeEval**: Đánh giá giá trị của một thuộc tính bằng cách sử dụng bộ phân loại do người dùng chỉ định.
- **GainRatioAttributeEval** Đánh giá giá trị của một thuộc tính bằng cách đo tỷ lệ gain tương ứng với mỗi nhãn.
- **InfoGainAttributeEval** Đánh giá giá trị của một thuộc tính bằng cách đo mức gain thông tin ứng với mỗi nhãn.
- **OneRAtrributeEval** Đánh giá giá trị của một thuộc tính bằng cách sử dụng bộ phân loại OneR.
- **SymmetricalUncertAttributeEval** Đánh giá giá trị của một thuộc tính bằng cách đo độ không đảm bảo đối xứng ứng với mỗi nhãn.
- **PrincipalComponents** Thực hiện phân tích thành phần chính (principal component analysis) và chuyển đổi dữ liệu.
- **LatentSemanticAnalysis** Thực hiện phân tích ngữ nghĩa tiềm ẩn và chuyển đổi dữ liệu.
- **CfsSubsetEval**: Đánh giá giá trị của tập con của các thuộc tính bằng cách xem xét khả năng dự đoán riêng của từng đặc trưng cùng với mức độ dư thừa (redundancy) giữa các đặc trưng với nhau.
- **ClassifierSubsetEval**: Đánh giá các tập con thuộc tính trên bộ dữ liệu huấn luyện hoặc kiểm thử. Sử dụng một bộ phân loại để ước tính giá trị của một tập con các thuộc tính.
- **WrapperSubsetEval** Đánh giá các tập thuộc tính bằng cách sử dụng một sơ đồ học tập.

Có thể thấy cái phương pháp trong Weka xoay quanh: đánh giá độ quan trọng của từng thuộc tính (AttributeEval), hoặc đánh giá độ quan trọng của từng tập con thuộc tính (SubsetEval), hoặc thực hiện phép biến đổi (PrincipalComponens, LatentSemanticAnalysis).

c. Trong textbook, các phương pháp được giới thiệu một cách khái quát và sơ lược, không đi sâu vào chi tiết. Nhìn chung, các phương pháp cũng thuộc về các dạng:

- Đánh giá mức độ tương quan giữa cặp thuộc tính: Nếu là các thuộc tính rời rạc, sử

dụng phép thử χ^2. Nếu là các thuộc tính số, sử dụng độ tương quan (correlation) hoặc hiệp phương sai (covariance) giữa hai thuộc tính. Trong Weka, việc lựa chọn thuộc tính là dựa vào mức độ tương quan giữa thuộc tính với thuộc tính làm nhãn.

- Biến đổi dữ liệu: nhằm tạo ra một bộ thuộc tính nhỏ hơn nhưng vẫn giữ được gần như đầy đủ thông tin. Dạng này tượng tự với PrincipalComponents của Weka.
- Chọn lọc tập thuộc tính con: sử dụng một phương pháp đánh giá (ví dụ như information gain) và đặt một ngưỡng để chọn ra những tập con của tập thuộc tính ban đầu vẫn đạt đủ ngưỡng yêu cầu. Có các cách tiếp cận như chọn lọc tiến (forward selection), loại bỏ lùi (backward elimination), cây quyết định (decision tree induction). Dạng này tương tự với các phương pháp tập con trong Weka.

4 Làm sạch dữ liệu (Cleaning)

a. Một số phương pháp làm sạch dữ liệu:

- **Bỏ qua dữ liệu bị thiếu**: Đây là một phương pháp không hiệu quả, vì chúng ta sẽ làm mất đi một số lượng mẫu đáng kể (nếu số lượng giá trị bị thiếu lớn). Thậm chí, nếu chẳng may dữ liệu này lại quan trọng thì sẽ ảnh hưởng đến kết quả.
- **Điền thông tin cho giá trị bị thiếu một cách thủ công**: Có thể dễ dàng nhận ra sự lãng phí thời gian trong phương pháp này, và thậm chí là không thể đối với một tập dữ liệu lớn.
- **Dán nhãn một giá trị khác cho dữ liệu bị thiếu**: Thay thế các giá trị bị thiếu bằng một nhãn như `Unknown` or `nan`. Mặc dù đơn giản, nhưng phương pháp lại không hoàn hảo. Chúng ta chưa hoàn toàn xử lý các dữ liệu bị thiếu, thay vào đó việc chúng ta làm chỉ là "dán" cho nó một cái nhãn mới. Khi huấn luyện trên tập dữ liệu này, máy tính có thể khai thác ra một số tri thức kỳ thú, ví dụ như những người "Unknown" có xu hướng bị bệnh tim chẳng hạn (rõ ràng là không hợp lý).
- **Sử dụng giá trị trung tâm như trung bình (median) hoặc trung vị (mean)**: Đối với dữ liệu đối xứng (phân phối gần chuẩn) thì dùng giá trị trung bình để điền, còn đối với dữ liệu lệch thì dùng trung vị. Ví dụ, độ tuổi trung bình là 45 thì ta điền 45 vào các vị trí tuổi còn thiếu. Phương pháp này có thể gây sai lệch vì làm giảm độ phân tán dữ liệu, gây xu hướng thiên vị (bias).
- **Sử dụng giá trị trung tâm của các mẫu cùng nhãn**: tương tự trên nhưng thay vì tính trung bình (hoặc trung vị) trên toàn dữ liệu, ta chỉ tính các thống kê này cho những dữ liệu có cùng nhãn. Ví dụ nếu ta phân loại bệnh tim thành có hoặc không, và tuổi trung bình của người có bệnh là 60, thì ta điền 60 cho những người có bệnh tim mà bị thiếu thông tin tuổi (thay vì 45 như trên). Phương pháp này cũng gặp vấn đề tương tự (tạo nên xu hướng luật là tuổi càng cao thì càng dễ bệnh tim, mặc dù thực tế có thể không phải vậy).
- **Dự đoán giá trị có thể xảy ra nhất cho giá trị khuyết**: Có thể dùng các mô hình như hồi quy, các mô hình dựa trên Bayes, hoặc cây quyết định để xác định. Các mô hình này có thể huấn luyện sử dụng các thuộc tính khác đã biết. Ví dụ ta xây dựng mô hình

dựa vào tuổi để đoán chiều cao và điền giá trị chiều cao tính toán được từ mô hình nếu bị thiếu.

Trong Weka, có hai phương pháp được cài đặt để sử lý dữ liệu thiếu là:

1. **ReplaceMissingValues** được dùng để thay để các giá trị thiếu bằng giá trị trung bình (đối với các thuộc tính số) và mode (đối với các thuộc tính rời rạc).
2. **ReplaceMissingWithUserConstant** được dùng để thay thế các giá trị bị thiếu bằng các giá trị cố định được điền bới người dùng.

Trong dữ liệu này, các thuộc tính bị thiếu bao gồm: `trestbps (1), chol (23), thalach (1), ca (296)` là các thuộc tính số và `fbs (8), restecg (1), exang (1), slope (190), thal (268)` là các thuộc tính rời rạc.

Do đó, sẽ là thích hợp khi sử dụng phương pháp `ReplaceMissingValues`.

b. Các phương pháp làm loại bỏ dữ liệu nhiễu:

- **Binning**: phương pháp này làm mịn một dữ liệu đã được sắp xếp bằng cách dựa vào các giá trị "hàng xóm" (giá trị xung quanh) của nó. Những giá trị này sẽ được phân chia vào các bin. Khi **làm mịn bằng trung bình** (hay trung vị), mỗi giá trị sẽ được thay thế bằng giá trị trung bình (hay trung vị) của bin. Còn **làm mịn bằng biên**, nghĩa là các giá trị lớn nhất và nhỏ nhất của bin được xem như biên của bin, và sau đó giá trị bin sẽ được thay thế bằng giá trị biên gần nhất.

  Ví dụ: Ta có một dãy dữ liệu đã được sắp xếp: 2, 7, 9, 12, 16, 23, 24, 25, 32

  Sắp xếp vào các bin, mới mỗi bin có độ lớn là 3:
  Bin 1: 2, 7, 9
  Bin 2: 12, 16, 23
  Bin 3: 24, 25, 32

  Làm mịn bằng mean:
  Bin 1: 6, 6, 6
  Bin 2: 17, 17, 17
  Bin 3: 27, 27, 27

  Làm mịn bằng biên:
  Bin 1: 2, 9, 9
  Bin 2: 12, 12, 23
  Bin 3: 24, 24, 32

- **Hồi quy (Regression)**: ta có thể làm mịn dữ liệu bằng cách xây dựng một mô hình dự đoán giá trị và ép các giá trị về mô hình đó. Ví dụ như sử dụng hồi quy tuyến tính tìm đường thẳng phù hợp và ép giá trị về nằm trên đường thẳng đó.
- **Xử lý điểm tạp**: Các điểm tạp có thể làm ảnh hưởng đến việc làm mịn, ví dụ như ảnh hưởng đến giá trị trung bình trong binning, hay khiến ta xây dựng mô hình hồi quy không hợp lý. Do đó, cần phát hiện và loại bỏ các điểm này.

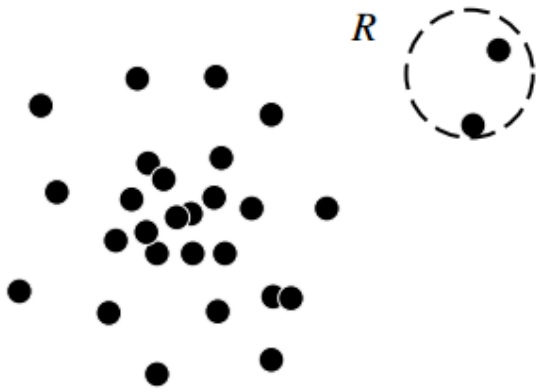
c. Các phương pháp dò tìm dữ liệu tạp (Outlier detection):

- **Phương pháp thống kê**: phương pháp này đặt ra giả thiết rằng các đối tượng bình

thường đi theo một mô hình thống kê xác định, và những dữ liệu không thỏa mô hình này là điểm tạp.

Ví dụ, ta có thể giả thiết tuổi của một người tuân theo phân phối chuẩn và lập nên một mô hình phân phối chuẩn (với trung bình và độ lệch chuẩn) phù hợp cho tập dữ liệu. Với mô hình này, những điểm với giá trị có xác suất thấp (nằm ngoài một mức độ lệch chuẩn nào đó) sẽ được cho là dữ liệu nhiễu.

Ví dụ trong hình 18, các điểm dữ liệu bên ngoài R tuân theo phân phối chuẩn.



Hình 18: Các điểm dữ liệu trong R là dữ liệu tạp. Source: textbook

Phương pháp này gồm 2 hướng tiếp cân. Một hướng là dựa trên tham số (parametric), có thể 1 hoặc nhiều tham số có trọng số, ví dụ như phân phối chuẩn ở trên. Một hướng khác không dựa trên tham số (non-parametric), ví dụ như sử dụng histogram.

- **Phương pháp dựa vào lân cận**: các dữ liệu bị cho là tạp khi mà có một khoảng cách lớn giữa nó và các "láng giềng" gần nhất. Phương pháp này có nhiều cách tiếp cận, ví dụ dựa trên khoảng cách hay dựa trên mật độ.

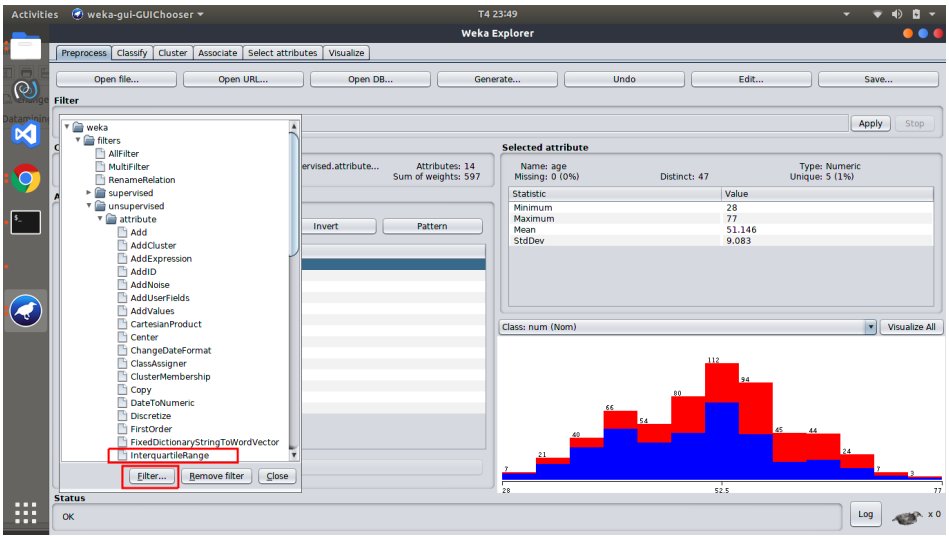
  Ví dụ, cũng dùng hình 18, ta có thể thấy các điểm trong R mặc dù có khoảng cách gần với điểm còn lại

- **Phương pháp dựa vào gom cụm**: phương pháp này đặt giả thiết là các đối tượng thông thường sẽ gom lại thành một cụm đặc khít, và do đó các dữ liệu thuộc về các cụm thưa hơn hoặc không thuộc về cụm nào sẽ bị xem là tạp. Cụ thể, 3 hướng tiếp cận cơ bản của phương pháp này đặt ra các câu hỏi: liệu đối tượng này có thuộc một cụm có sẵn? liệu có khoảng cách lớn giữa đối tượng và một cụm gần nhất với nó? liệu đối tượng có thuộc về một cụm nhỏ và thưa?

- **Phương pháp dựa trên phân lớp**: Một bộ phân lớp được xây dựng để phân biệt các dữ liệu thông thường với các dữ liệu tạp. Tuy nhiên, khó khăn của phương pháp này là không phải lúc nào cũng có nhãn dán cho các điểm dữ liệu, cũng như nếu có thì sẽ rất mất cân bằng (do số điểm tạp rất ít).

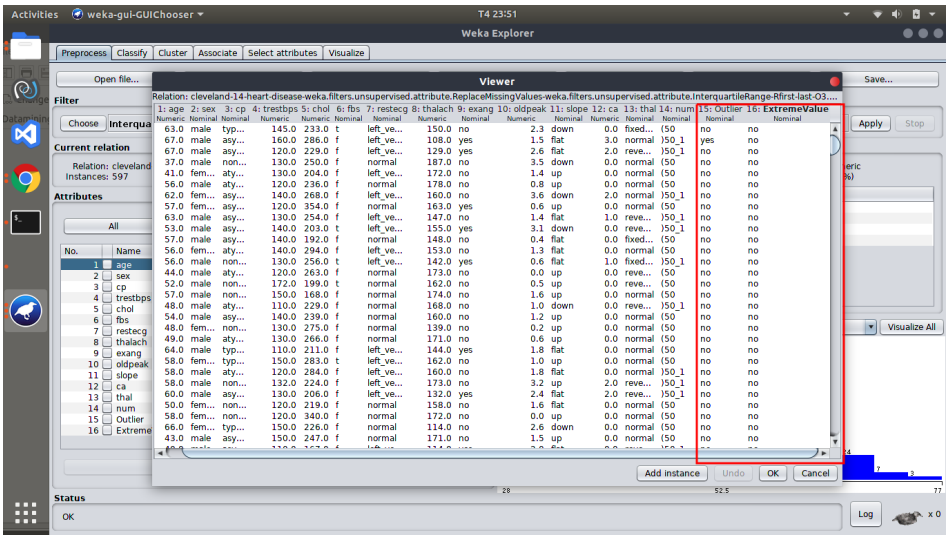
Để dò tìm dữ liệu tạp trong Weka, chúng tôi thực hiện các bước sau:

1. Trong mục Filter, chọn `unsupervised/attribute/InterquartileRange` (xem Hình 19).



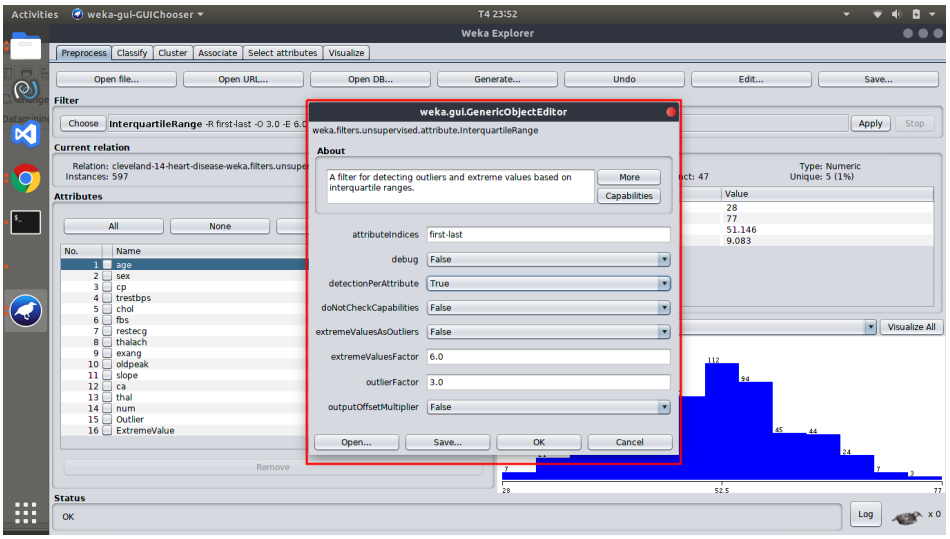
Hình 19: Chọn InterquartileRange

Sau khi nhấn **Apply**, ta nhấn **Edit** để xem kết quả như hình 20. Tập dữ liệu của chúng ta đã được tạo thêm 2 thuộc tính `Outlier` và `Extreme Value`, với giá trị `yes` nghĩa là có dữ liệu tạp, và `no` là ngược lại.

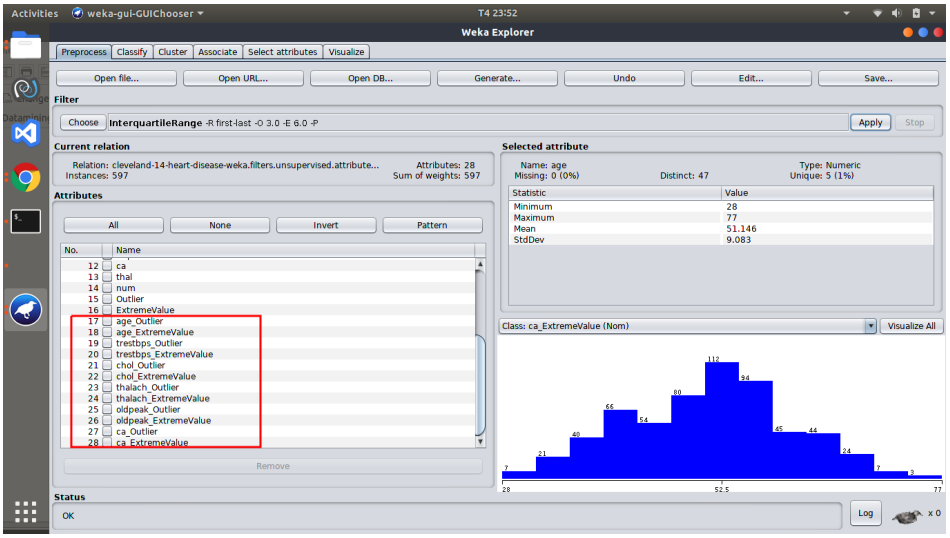


Hình 20: Kết quả sau khi chọn filter

2. Tiếp theo, thay đổi **detectionPerAttribute** của bộ lọc **InterquartileRange** từ `False` thành `True` (như hình 21). Kết quả là mỗi thuộc tính sẽ được tạo ra thêm các thuộc tính có `Outlier` và `ExtremeValue` (xem hình 22).



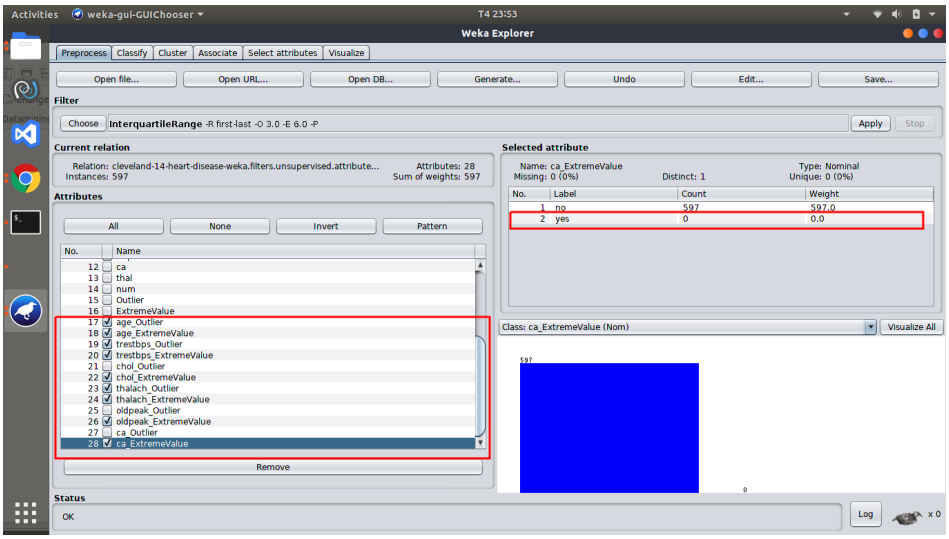
Hình 21: Thay đổi giá trị **detectionPerAttribute**



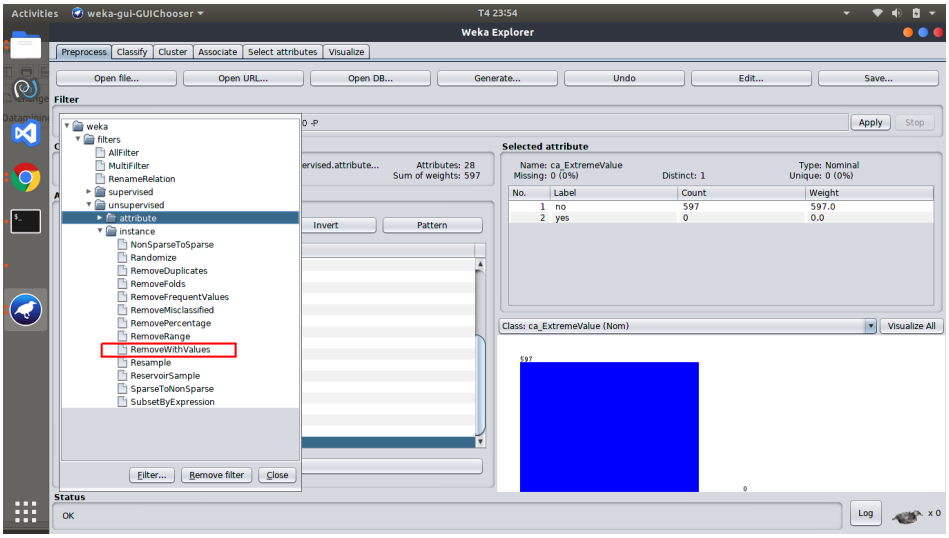
Hình 22: Kết quả sau khi thay đổi

3. Chọn các thuộc tính vừa mới tạo thêm nào có số lượng mẫu có giá trị `outlier, extreme` là 0 để remove, tức không cần chú ý đến các thuộc tính không có tạp (xem Hình 23).

4. Sau khi tìm được các thuộc tính chứa giá trị tạp, dùng filter **RemoveWithValues** (trong danh sách `unsupevised/instance`) để loại bỏ các giá trị đó ra khỏi tập dữ liệu (xem Hình 24). Sau đó, nhấn chuột trái vào ô vuông chứa tên filter, một hộp thoại chỉnh sửa thông số sẽ xuất hiện. Ở đây, có 2 giá trị cần được sửa: **attributeIndex** sẽ là thứ tự của thuộc tính `Outlier` trong danh sách các thuộc tính; **nomialIndices** sẽ thay đổi thành `last` như hình 25.



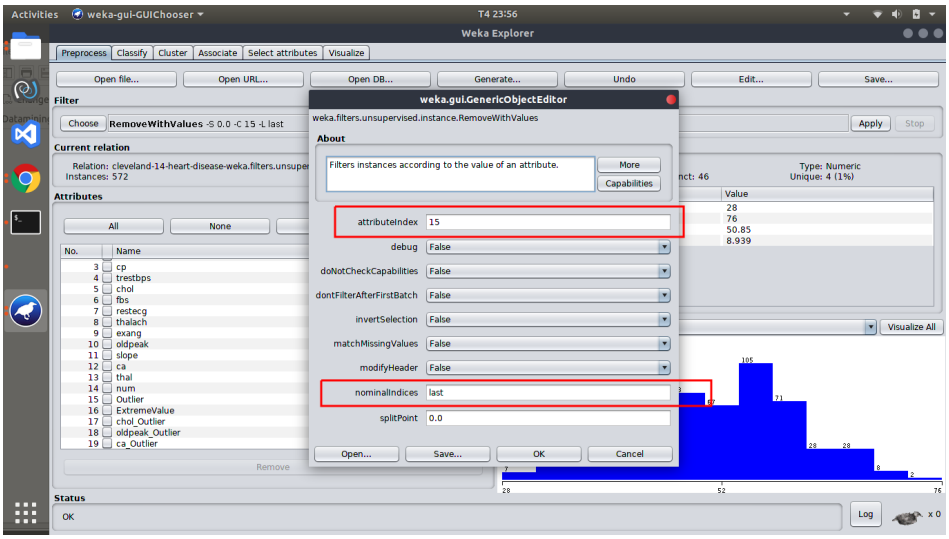
Hình 23: Xóa các thuộc tính có giá trị `outlier, extreme` là 0



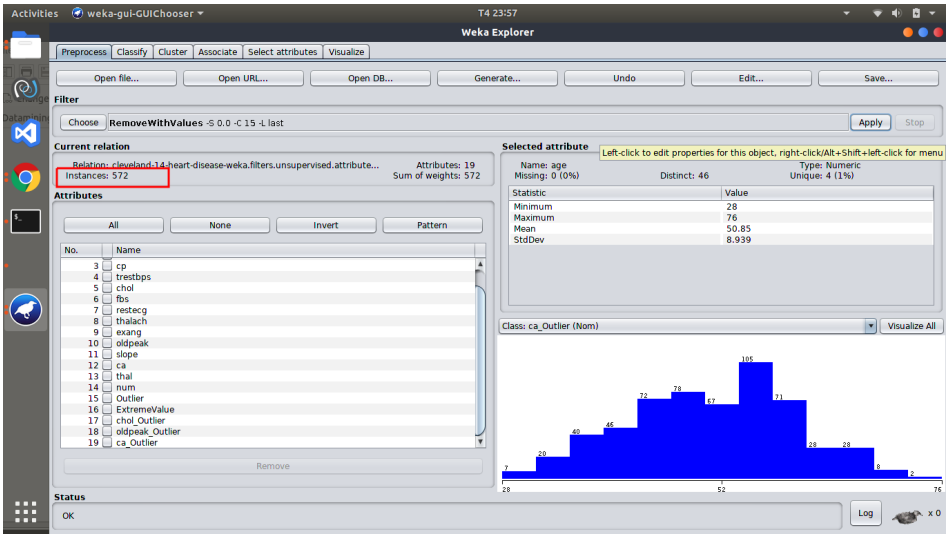
Hình 24: Dùng filter RemoveWithValues

5. Cuối cùng, sau khi thực hiện các bước, ta sẽ thu được tập dữ liệu đã được loại bỏ dữ liệu tạp. Trong dữ liệu này, số mẫu đã được giảm xuống còn 572 mẫu (hình 26).

Sau khi áp dụng filter **InterquatileRange** trong Weka, một số thuộc tính có chứa dữ liệu tạp là: `chol, oldpeak và ca`. Có thể thấy điều này qua Hình 27.

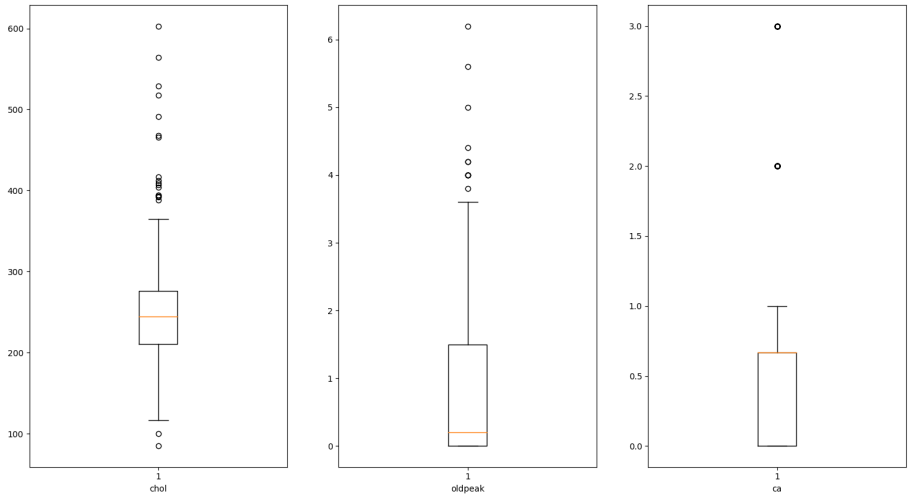


Hình 25: Thay đổi thông số của filter RemoveWithValues

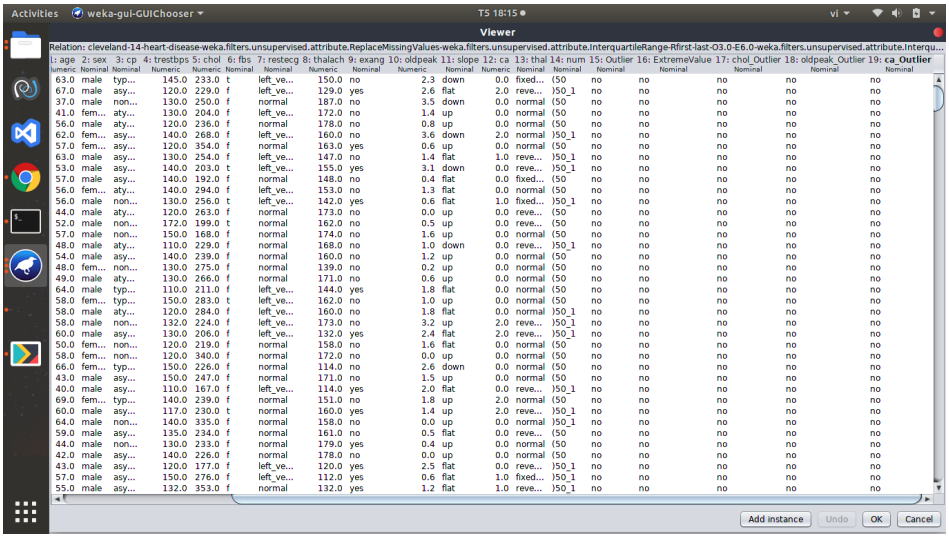


Hình 26: Tập dữ liệu sau khi đã loại bỏ dữ liệu tạp

d. Hình 28 cho thấy tập dữ liệu sau khi làm sạch.



Hình 27: Biểu đồ hộp cho thấy các phân vị và điểm tạp



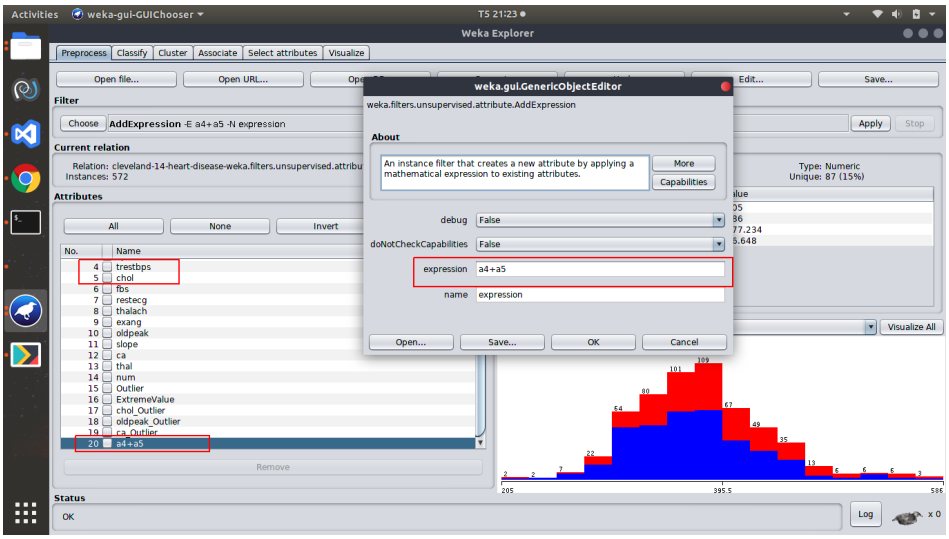
Hình 28: Kết quả tập dữ liệu sau khi được làm sạch

5 Chuyển đổi dữ liệu (Transformation)

a. Để tạo một thuộc tính mới dựa trên những thuộc tính đã có sẵn, chúng tôi dùng filter **AddExpression**. Đây là filter tạo ra một thuộc tính mới bằng cách áp dụng một biểu thức toán học vào các thuộc tính hiện tại. Trong trường hợp dữ liệu này, chúng tôi giả sử ghép 2 thuộc tính `trestbps` và `chol` với biểu thức toán là tổng từng giá trị của 2 thuộc tính. Cụ thể trong hình 29, thứ tự của 2 thuộc tính trên lần lượt là 4 và 5, nên ta có biểu thức tổng như sau: **a4+a5**. (xem Hình 29).

b. Weka cho phép chuẩn hóa (normalize) dữ liệu thông qua 2 bộ lọc: **Standardize** và **Normalize**.

Bộ lọc **Standardize** dùng để thay đổi giá trị của các thuộc tính số sao cho chúng có giá trị



Hình 29: Dùng AddExpression

trung bình là 0 và độ lệch chuẩn là 1. Cụ thể, với mỗi thuộc tính, ta tính giá trị trung bình μ và độ lệch chuẩn σ, sau đó mỗi giá trị x sẽ được thay bằng:

$$X = \frac{x - \mu}{\sigma}$$

Bộ lọc này giả định rằng dữ liệu có phân phối chuẩn.

Bộ lọc **Normalize** dùng để thay đổi giá trị của một hoặc nhiều thuộc tính số về phạm vi từ 0 đến 1 bằng cách: đối với mỗi thuộc tính, cho giá trị lớn nhất là 1 và giá trị nhỏ nhất là 0, thay các giá trị chính giữa bằng tỉ lệ chênh lệch của nó với hai giá trị biên này, cụ thể với mỗi giá trị x, ta thay bằng:

$$X = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Để sử dụng 2 bộ lọc này, lần lượt chọn `unsupervised/attribute/Standardize` (hoặc `Normalize`). (xem Hình 30).

Trong Weka không có sẵn bộ lọc để chuẩn hóa thập phân. Tuy nhiên, ta có thể làm được việc này bằng AddExpression ở trên.

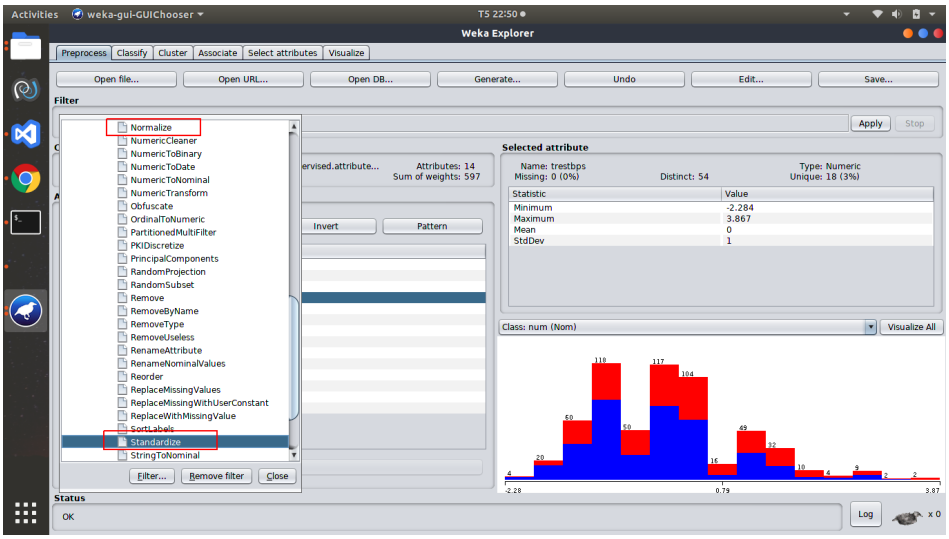
Chuẩn hóa thập phân là công việc đưa giá trị về phạm vi từ 0 đến 1 bằng cách: tìm số lượng chữ số của phần nguyên của số lớn nhất, gọi đó là i, khi đó, với mỗi giá trị x, ta thay bằng

$$X = \frac{x}{10^i}$$

Ví dụ thuộc tính có các giá trị là `1, 5, 12, 89, 192, 73` sẽ trở thành `0.001, 0.005, 0.012, 0.089, 0.192, 0.073`

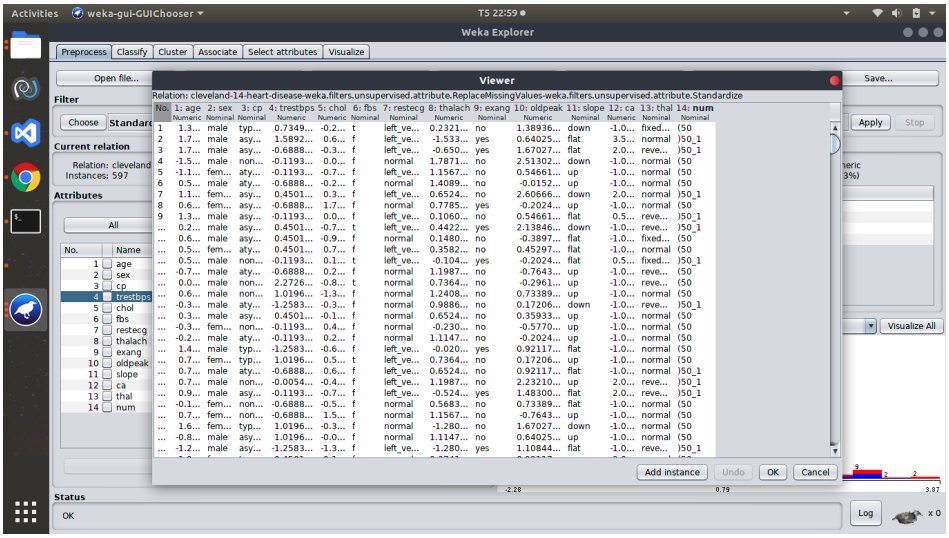
Như vậy, ta chỉ cần thêm bộ lọc AddExpression để chia cho số lượng chữ số giá trị max.

c. Chúng tôi cho rằng các giá trị của các thuộc tính đều tuân theo phân phối chuẩn. Do đó, phương pháp phù hợp là Standardize. Một điểm mạnh của lựa chọn này là nó sẽ không quá bị ảnh hưởng bởi giá trị nhiễu.



Hình 30: Hai bộ lọc Normalize và Standardize

d. Hình 31 cho thấy kết quả sau khi chuẩn hóa.



Hình 31: Kết quả sau khi chuẩn hóa

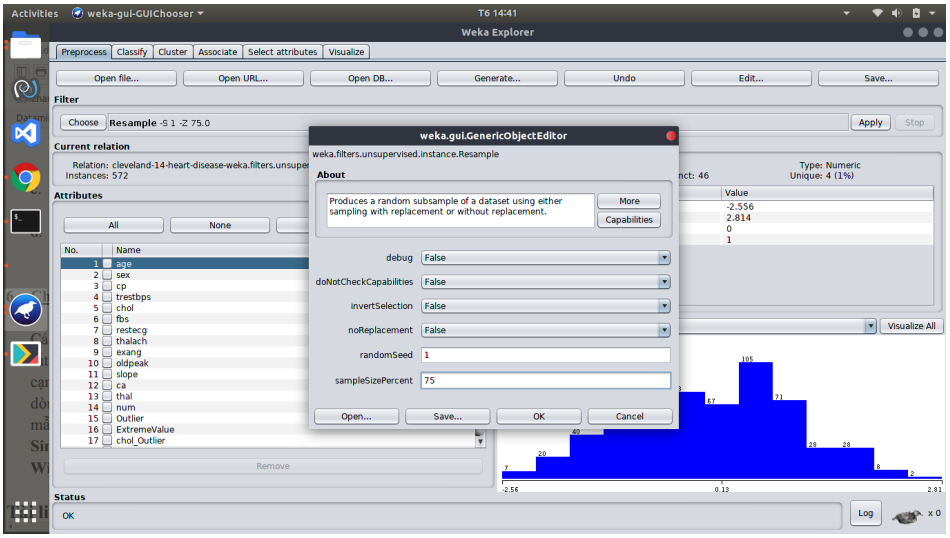
6 Rút gọn dữ liệu (Reduction)

Một phương pháp rút gọn dữ liệu là chọn lọc một tập dữ liệu con từ tập dữ liệu ban đầu với số lượng nhỏ hơn. Hai hướng tiếp cận chính của phương pháp này là chọn không thay thế (tức các phần tử được chọn ra đảm bảo là các phần tử khác nhau trong tập dữ liệu ban đầu) và chọn có thay thế (tức các phần tử được chọn ra có thể có trùng lặp).

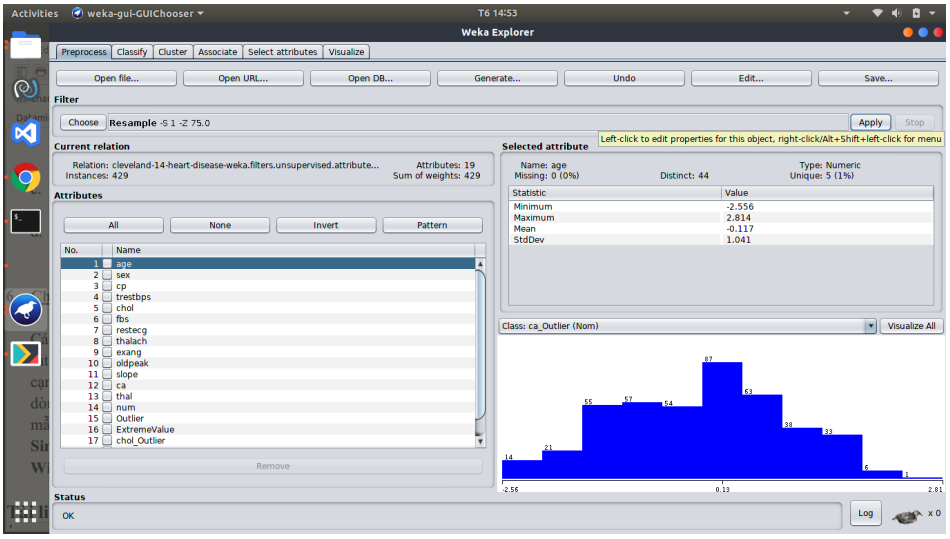
Ví dụ, từ tập dữ liệu ban đầu `1,2,3,4,5,6,7,8,9,10`, ta tạo tập con với kích thước giảm đi 40%, tức 6 phần tử. Với chọn không thay thế, ta có thể được `1,5,6,7,9,10` hoặc `2,3,5,7,8,9`. Còn với chọn có thay thế, ta có thể được `1,1,1,2,3,4` hay `2,2,3,4,4,5`

(như vậy, kể cả khi kích thước tập con bằng kích thước tập ban đầu, ta vẫn có thể sẽ mất một số dữ liệu).

Trong Weka, sử dụng bộ lọc **Resample** để rút gọn dữ liệu theo phương pháp nêu trên. Để sử dụng bộ lọc, chọn `unsupervised/instances/Resample`. Bộ lọc này có thể sử dụng với cả 2 hướng tiếp cận là chọn ngẫu nhiên có thay thế (with replacement) và không có thay thế (without replacement). Như hình 32, chúng tôi thay đổi thông số **sampleSizePercent**, nghĩa là giảm số lượng mẫu đi 30%. Đối với thông số **noReplacement**, nếu chọn `True` nghĩa là không có thay thế, ngược lại là có. Ở đây, chúng tôi chọn có thay thế và giảm dữ liệu 30%, và kết quả biểu thị như hình 33.



Hình 32: Chọn bộ lọc Resample



Hình 33: Kết quả sau khi rút gọn dữ liệu

Tài liệu

[1] Trang chủ của Weka: `http://www.cs.waikato.ac.nz/ml/weka/`

[2] Textbook: J. Han and M. Kamber: Data Mining, Concepts and Techniques, Second Edition