Discretization of continuous attributes

Transforming a continuous attribute into a discrete (ordinal) attribute

Ricco RAKOTOMALALA

Université Lumière Lyon 2

Outline

1. What is the discretization process? Why discretize?
2. Unsupervised approaches
3. Supervised approaches
4. Conclusion
5. References

Ideas underlying the discretization task

Why and how to discretize a continuous attribute?

Converting a continuous attribute into a discrete one (with a small set of values)

X	Classes
14	C1
15	C1
17	C1
19	C1
21	C1
23	C1
25	C1
26	C1
27	C1
28	C1
29	C1
31	C1
38	C2
39	C2
41	C2
44	C2
45	C2
49	C3
51	C3
53	C3

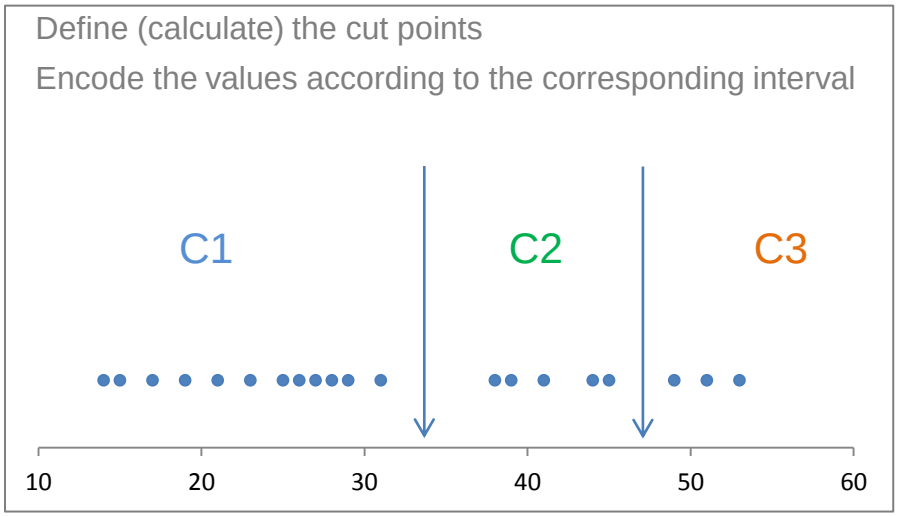
X is a quantitative (continuous) variable

It is converted into an ordinal (discrete) variable

2 steps:

Define (calculate) the cut points

Encode the values according to the corresponding interval



Main issues

(1) How to choose the number of intervals K

(2) How to define the cut points

… which are relevant according to the studied problem….

Why do we need to discretize?

- Some statistical method or machine learning algorithms can handle discrete attributes only.

  E.g. Rule induction methods (predictive rules, association rules,…)

- Harmonization of the type of variables in heterogeneous data sets.

  (NB: We can also harmonize the data set by turning categorical variables in a numerical ones e.g. dummy coding)

  E.g. Discretize the continuous attributes and perform a multiple correspondence analysis on the whole database (including the original nominal variables)

- Transforming the characteristics of the data set in order to make the subsequent statistical algorithm more efficient.

  E.g. To correct skewed distribution, to reduce the influence of outliers, to handle nonlinearities.

A domain expert may adapt the discretization to the context and the goal of the study.

He takes into account other information than those only provided by the available dataset.

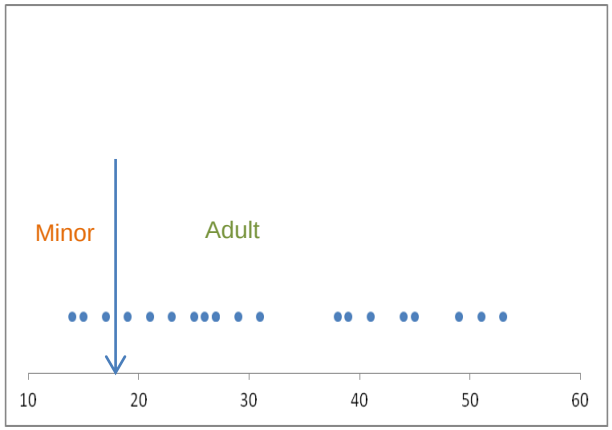
X = Age

The majority age is a possible cut point.

X < 18 : minor

X ≥ 18 : adult

Age	Majorité
14	Mineur
15	Mineur
17	Mineur
19	Majeur
21	Majeur
23	Majeur
25	Majeur
26	Majeur
27	Majeur
28	Majeur
29	Majeur
31	Majeur
38	Majeur
39	Majeur
41	Majeur
44	Majeur
45	Majeur
49	Majeur
51	Majeur
53	Majeur



☹ But this solution is not obvious on the most of cases, because the expertise is rare.

➔ **We need statistical approach based on the dataset characteristics.**

Unsupervised discretization

Methods that use only the characteristics of the variable to discretize

A few statistical indicators calculated from data

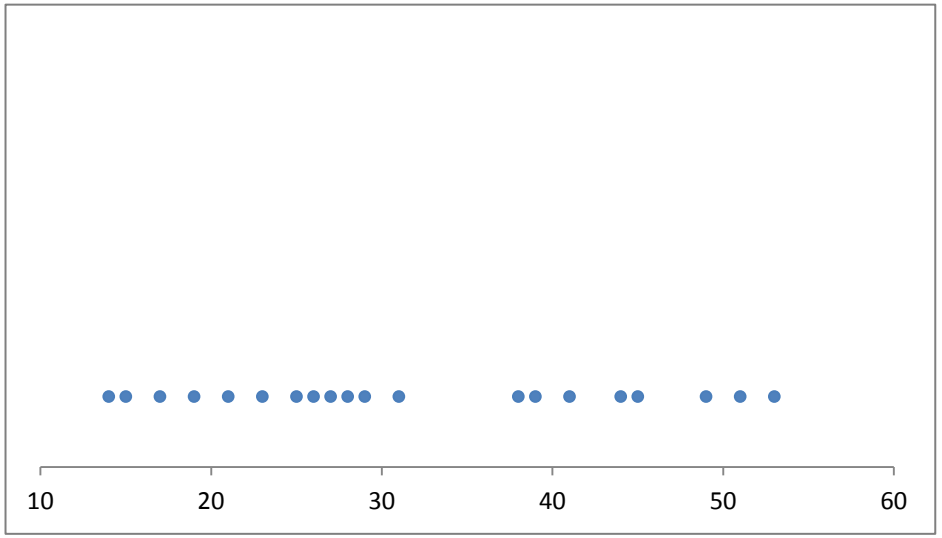
X
14
15
17
19
21
23
25
26
27
28
29
31
38
39
41
44
45
49
51
53

n	20

Min	14
Max	53

1st quartile	22.5
Median	28.5
3rd quartile	41.75

Mean	31.75
Standard deviation	12.07



How to use such information as guides to obtain a relevant discretization?

Some popular unsupervised approaches

Equal width intervals method (1)

K : the number of interval is a parameter

Divide the dataset into K ranges of equal size

$$a = \frac{\max - \min}{K}$$

We calculate from the data the width of the intervals (a)

We derive the (K-1) cut points (b_1, b_2, etc.)

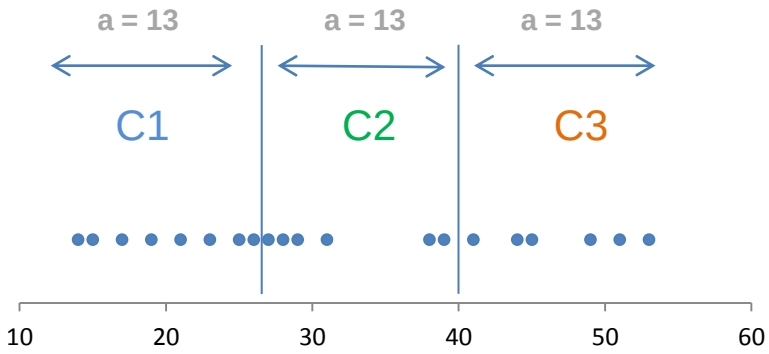
$$b_1 = \min + a$$

$$b_2 = b_1 + a = \min + 2 \times a$$

$$\ldots$$

X	Classes
14	C1
15	C1
17	C1
19	C1
21	C1
23	C1
25	C1
26	C1
27	C2
28	C2
29	C2
31	C2
38	C2
39	C2
41	C3
44	C3
45	C3
49	C3
51	C3
53	C3

K	3
max	53
min	14
a	**13**
b1	**27.0**
b2	**40.0**



C1 : x < 27

C2 : 27 ≤ x < 40

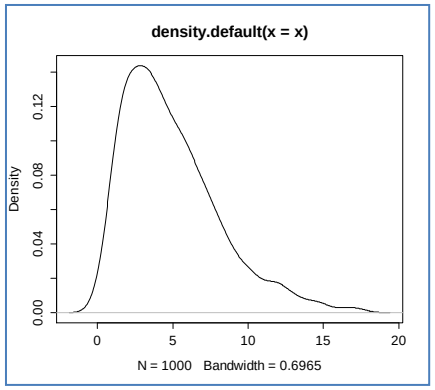
C3 : x ≥ 40

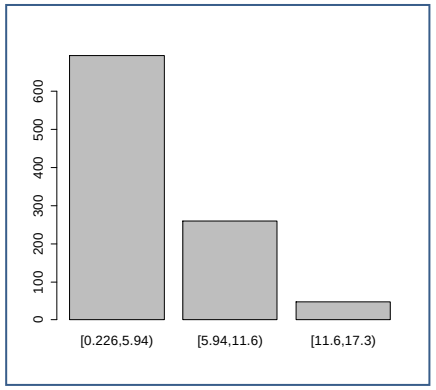
Strict or not inequalities are defined arbitrarily.

Equal width intervals method (2)

Simplicity and quickness

The shape of the distribution is not modified





The choice of K (the number of intervals) is not obvious

Sensitivity to the outliers

Possibility of obtaining intervals with a very few instances or even empty

The cut points do not take into account the proximities of the values

Equal frequency intervals (1)

K : the number of interval is a parameter

The same (roughly) number of instances in each interval

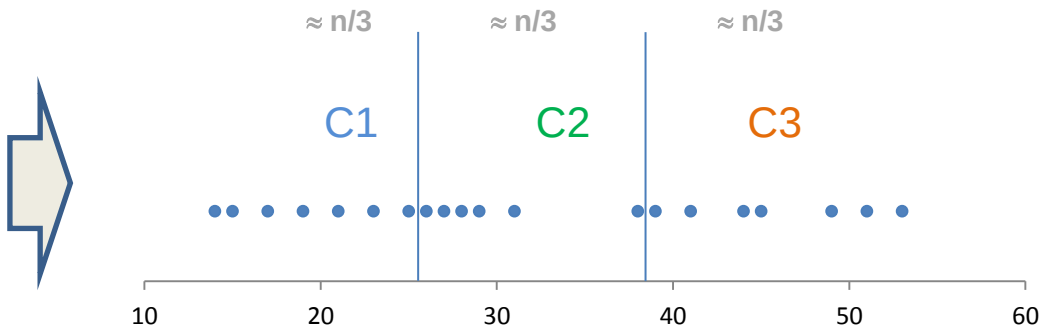
We calculate the quantiles from the data (q_1, q_2)

Quantiles = cut points (q_1, q_2, etc.)

Ex. quantile of order 0.25 = 1st quartile ; quantile of order 0.5 = median ; etc.

X	Classes
14	C1
15	C1
17	C1
19	C1
21	C1
23	C1
25	C1
26	C2
27	C2
28	C2
29	C2
31	C2
38	C2
39	C3
41	C3
44	C3
45	C3
49	C3
51	C3
53	C3

q(0.33)	25.33
q(0.66)	38.67



C1 : x < 25.33

C2 : 25.33 ≤ x < 38.67

C3 : x ≥ 38.67

Strict or not inequalities are defined arbitrarily.

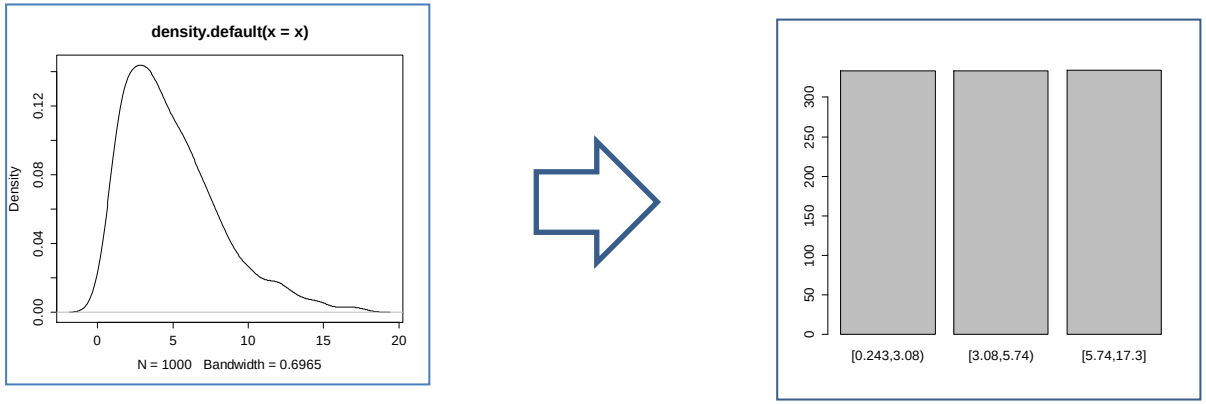
Equal frequency intervals (2)

Simplicity and quickness (we need to sort the instances)

Not sensitive to the outliers

The number of instances in each interval is defined a priori

Balancing the shape of the distribution



The choice of K (the number of intervals) is not obvious

The cut points do not take into account the proximities of the values

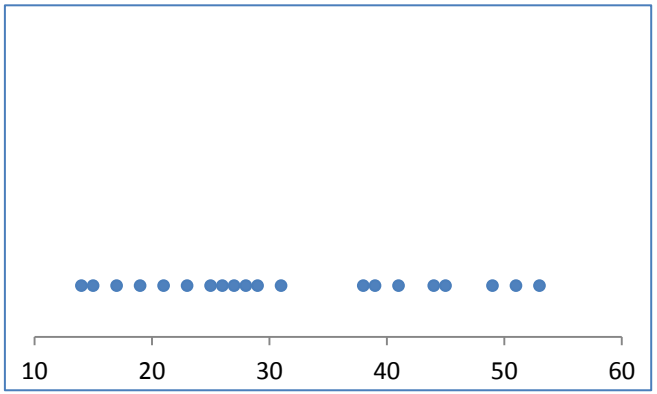
Some formulas to determine the "right" number of intervals K

(http://www.info.univ-angers.fr/~gh/wstat/discr.php)

Approach	Formula	K (without rounding)
Brooks-Carruthers	$5 \times \log_{10}(n)$	6.51
Huntsberger	$1 + 3.332 \times \log_{10}(n)$	5.34
Sturges	$\log2(n + 1)$	4.39
Scott	$(max - min)/(3.5 \times \sigma \times n^{-1/3})$	2.50
Freedman-Diaconis	$(max - min)/(2 \times IQ \times n^{-1/3})$	2.75

σ: standard deviation

IQ : interquartile range



The two last formulas use more information from the data (range, dispersion).

Nested means. Top-down algorithm. The first cut point is the mean computed from the whole sample. Then, we subdivide the sample with this cut point. We calculate the local average in each subsample. We subdivide again. Etc. The number of intervals is a power of 2.

Large relative difference. We sort the data in increasing order. We identify the large difference between 2 successive values. We define a cut point if the gap is upper than a threshold expressed in % of the standard deviation of the values (or in percentage of MAD - median absolute deviation - if you want to avoid the outliers problem).

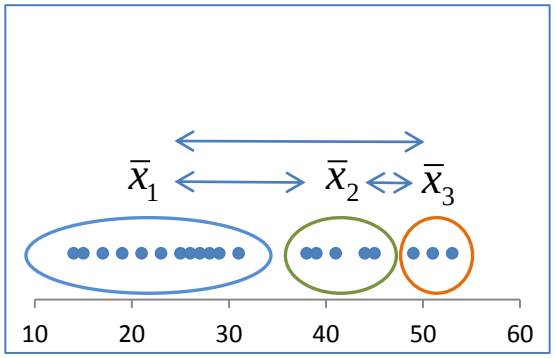
Difference to the average. K is a parameter. If K is even, on both side of the average, the first intervals are a range σ [or m x σ, m is a parameter], etc. until we have K intervals [last intervals have a different width]. If K is odd, the first interval is around the average.

Approaches that take into account the dispersion of values

Discretization based on clustering methods

Take into account the proximity between the values

Data can be organized into "clusters". We are interested in the characteristics of dispersion of data.



ANOVA equation
(analysis of variance)

$$T = B + W$$

$$\sum_{i=1}^{n}(x_i - \bar{x})^2 = \sum_{k=1}^{K} n_k(\bar{x}_k - \bar{x})^2 + \sum_{k=1}^{K}\sum_{i=1}^{n_k}(x_{ik} - \bar{x}_k)^2$$

Goal: Maximizing the differences between the conditional means

$$\eta^2 = \frac{B}{W}$$

η is the correlation ratio

An optimal approach exists (Fisher algorithm), but…

☹ The number of intervals K remains a parameter

☹ The algorithm requires quadratic time $O(n^2)$.

☹ It is not available into data mining tools.

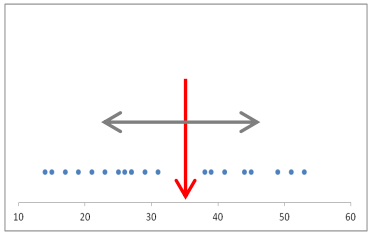
Top down strategy (1)

Find the best binary separation

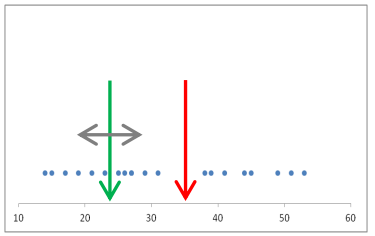
Continue recursively in each subgroup

Until stopping rules are met

(1)



(2)



Etc.

What stopping rules?

- A new partition does not imply a significant difference between conditional means (α significance level, pay attention about the multiple comparisons problem)
- Minimum number of instances before or after a splitting process
- Maximum number of intervals

We use a regression tree program where the target and the input attributes are the same.

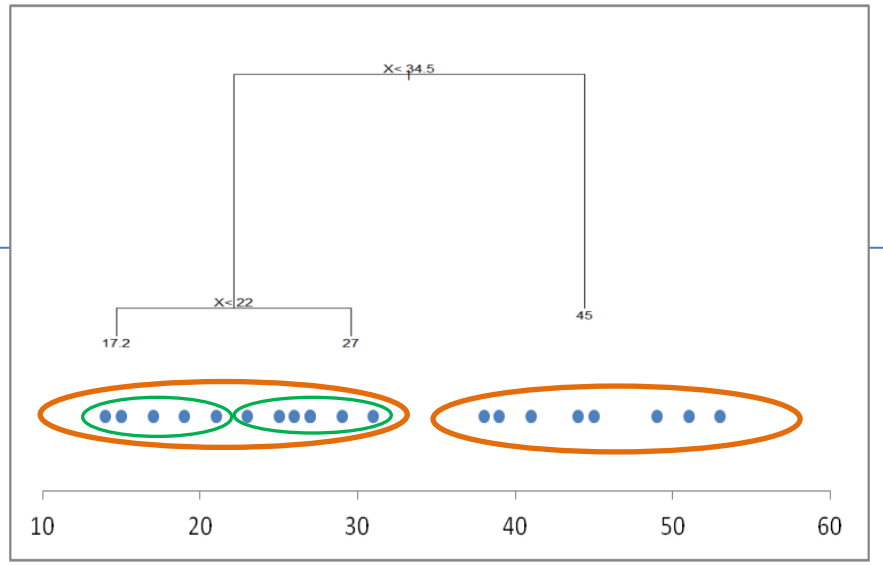
The program creates groups which maximize the between-group variance (e.g. rpart() of R)

```
> library(rpart)
> Y <- donnees$X
> arbre <- rpart(Y ~ X, data = donnees, method = "anova", control = rpart.control(minsplit=5,cp=0.07))
> print(arbre)
n= 20

node), split, n, deviance, yval
      * denotes terminal node

1) root 20 2913.7500 31.75000
  2) X< 34.5 12  354.9167 22.91667
    4) X< 22 5   32.8000 17.20000 *
    5) X>=22 7   42.0000 27.00000 *
  3) X>=34.5 8  218.0000 45.00000 *
> plot(arbre)
> text(arbre)
```

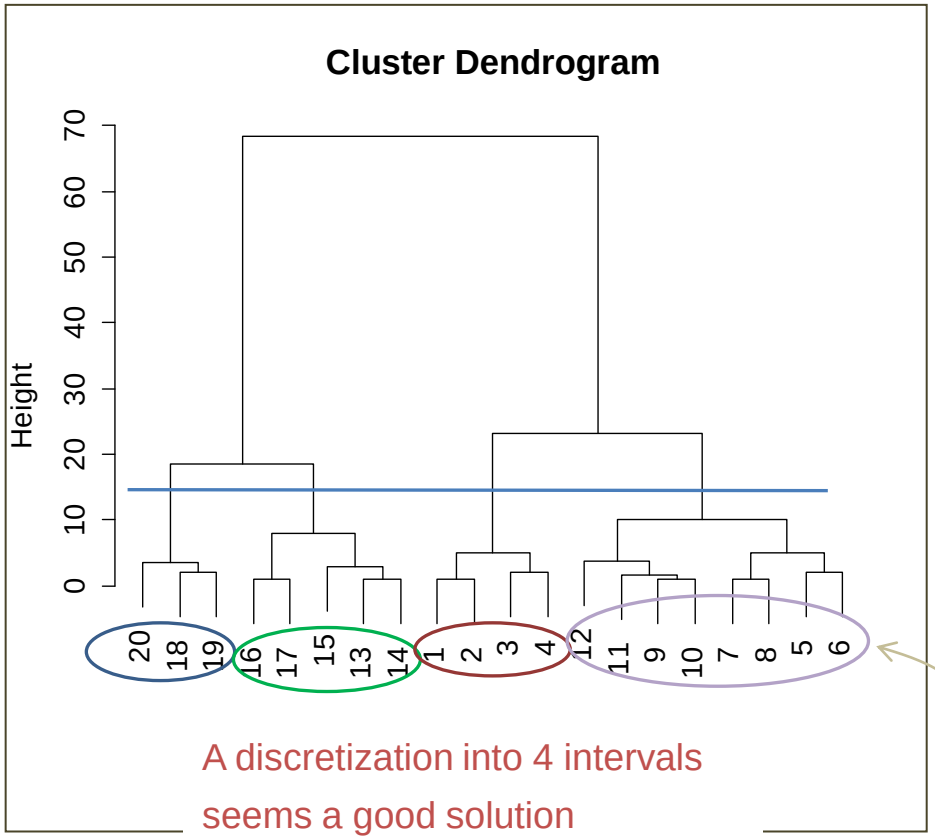
The result is very sensitive to the algorithm settings.



Note: plotcp() and prunecp() of rpart() can help us to detect the "right" number of groups

Bottom-up approach – Using the HAC algorithm.

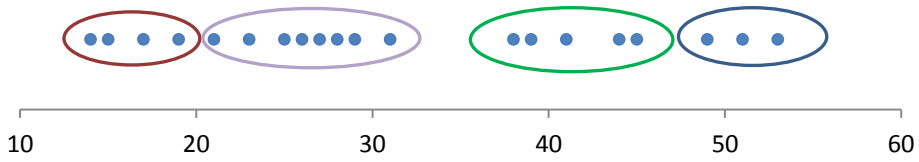
```
#distance matrix
d <- dist(donnees)
#HAC, WARD's method
dendro <- hclust(d,method="ward.D2")
#plot
plot(dendro)
```



A discretization into 4 intervals seems a good solution

N° of observation

The solution suggested by the HAC method



+ Use more information than the previous approaches
+ The resulting groups (intervals) are more compact
+ The quality of the discretization may be measured (correlation ratio)

- Some tools which suggest the right number of intervals are available

* The top-down algorithm is much more faster than the bottom-up approach
* The top-down algorithm can be applied on a large number of variables automatically (if we can defined a parameter which is appropriate on any variable..?)

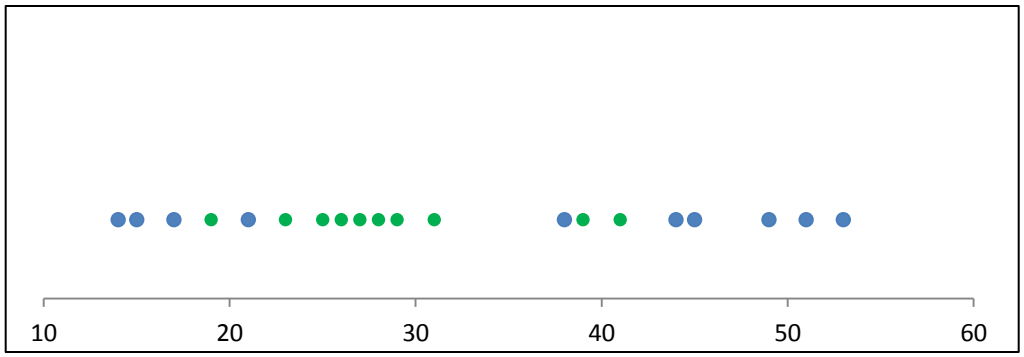
Supervised discretization

In a predictive analytics scheme, a target attribute Y may guide the discretization of the attribute X

Discrete (categorical) target attribute

The instances are labeled by a target variable Y
(they belong of predefined classes).

X	Y
14	Y1
15	Y1
17	Y1
21	Y1
38	Y1
44	Y1
45	Y1
49	Y1
51	Y1
53	Y1
19	Y2
23	Y2
25	Y2
26	Y2
27	Y2
28	Y2
29	Y2
31	Y2
39	Y2
41	Y2



How to cut X into intervals in order to gather together - as much as possible - the instances with the same label (color)?

➔ Number of intervals?

➔ Cut points?

Note: individuals with the same label are not necessarily adjacent

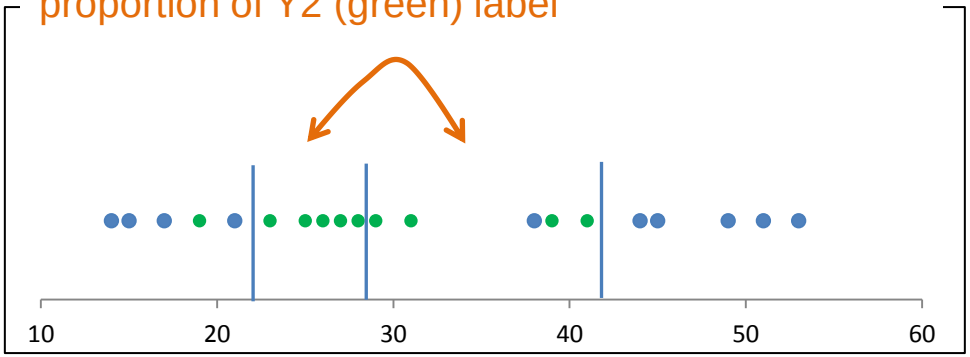
Intuitive bottom-up approach, widely used and yet ...

- Intervals are first defined using a simplistic approach (e.g. deciles, quartile, …)
- Merging in an iterative process the adjacent intervals with a similar classes distribution
- Stop when no "relevant" merge is possible

X	Y
14	Y1
15	Y1
17	Y1
21	Y1
38	Y1
44	Y1
45	Y1
49	Y1
51	Y1
53	Y1
19	Y2
23	Y2
25	Y2
26	Y2
27	Y2
28	Y2
29	Y2
31	Y2
39	Y2
41	Y2

1st quartile	22.50
Median	28.50
3rd quartile	41.75

Merge these intervals because we observe a high proportion of Y2 (green) label



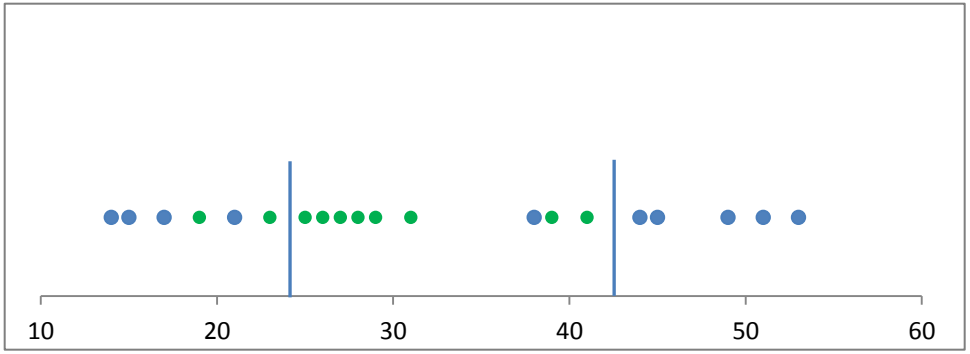
Drawbacks:

- The process cannot be automated for the processing of a dataset with a large number of variables
- The intervals defined in the first step may be inappropriate (unless we use a high order quantile)
- The approach relies heavily on the intuition of the user

Bottom-up approach – Chi-Merge (Kerber, 1992)

- Intervals are first defined for each value
- Iterative merging of adjacent intervals according to chi-square test of classes distribution equivalences (the most similar intervals are merged first)
- Stop when no "relevant" merge is possible
- The significance level α is a parameter of the algorithm

```
> print(summary(don.sup))
       X          Y
 Min.   :14.00   Y1:10
 1st Qu.:22.50   Y2:10
 Median :28.50
 Mean   :31.75
 3rd Qu.:41.75
 Max.   :53.00
>
> library(discretization)
> res.cm <- chiM(don.sup,alpha=0.05)
> print(res.cm$cutp)
[[1]]
[1] 22.0 42.5
```



Conclusion:

+ Scientifically founded. Available in many tools.

- Not really fast, especially on large datasets
- The choice of α is not easy. Multiple comparisons context.

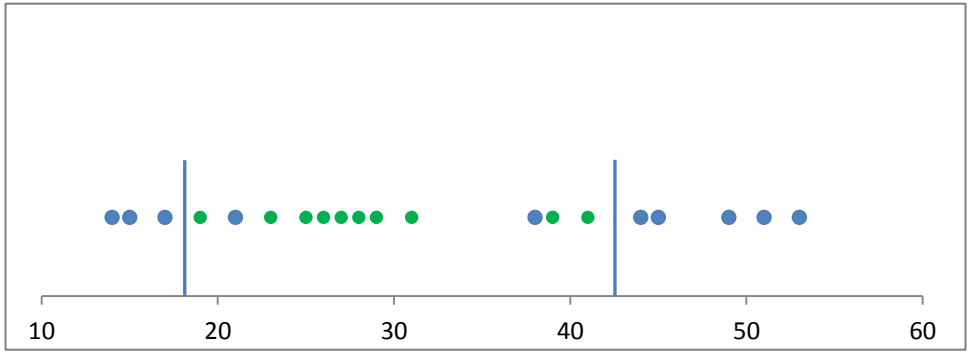
Top-down approach – Using a decision tree algorithm

- Using a standard decision tree algorithm
- Y is the target attribute, X is the only input attribute
- The results relies on the settings of the algorithm

```
> arbre.2 <- rpart(Y ~ X, data = don.sup, method = "class", control = rpart.control(minsplit=5))
> print(arbre.2)
n= 20

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 20 10 Y1 (0.5000000 0.5000000)
  2) X>=42.5 5  0 Y1 (1.0000000 0.0000000) *
  3) X< 42.5 15  5 Y2 (0.3333333 0.6666667)
    6) X< 18 3  0 Y1 (1.0000000 0.0000000) *
    7) X>=18 12  2 Y2 (0.1666667 0.8333333) *
```



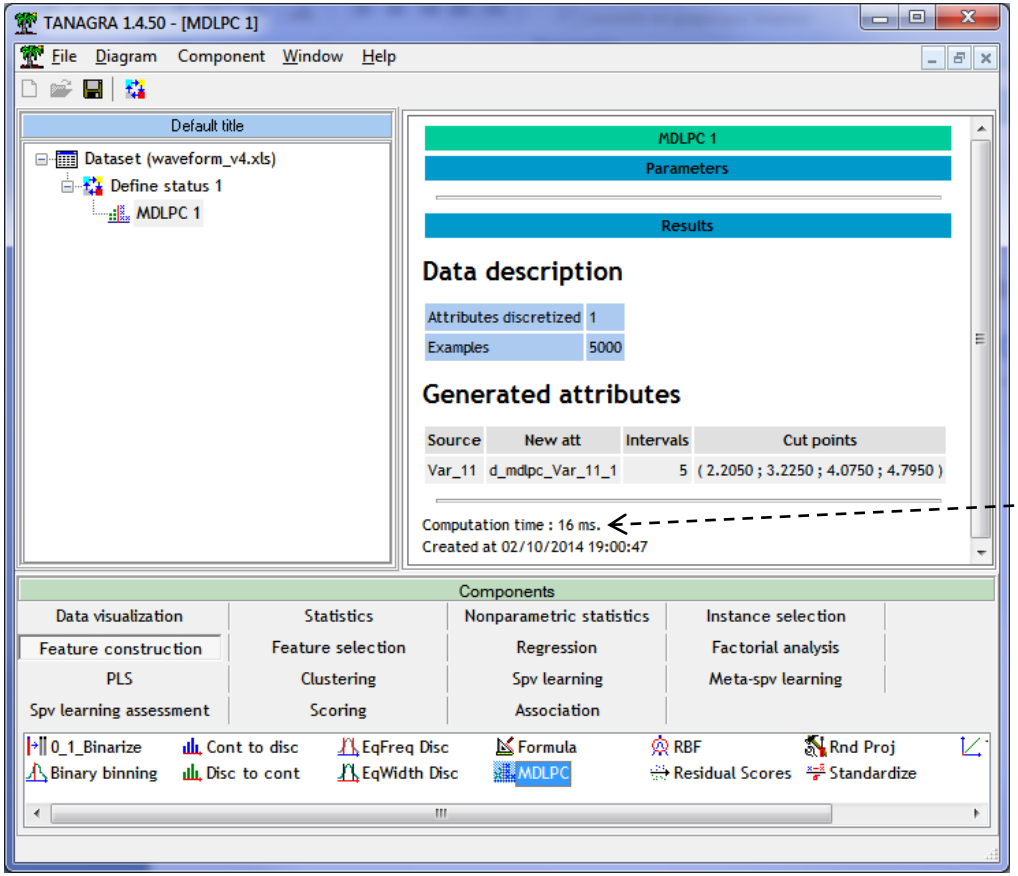
Conclusion :

+ Scientifically founded. Available in popular tools.

+ Quickness on large databases

- Determining the good parameters for the decision tree algorithm is not always obvious

Top-down approach – MDLPC (Fayyad & Irani, 1993)



- Top down process (based on decision tree algorithm)
- It uses a stopping rule especially intended for the discretization
- **A "state-of-the-art" method**

Really fast ! 16 ms. for the discretization in 5 intervals of a variable with 5000 instances

Conclusion :

+ Scientifically founded. Available in popular tools.
+ Quickness
+ No settings
- No settings i.e. we cannot adapt the algorithm according to the data characteristics

On our dataset (10 instances), MDLPC does not perform a subdivision... is it right?

Why a supervised discretization is better for a supervised task?

IRIS dataset

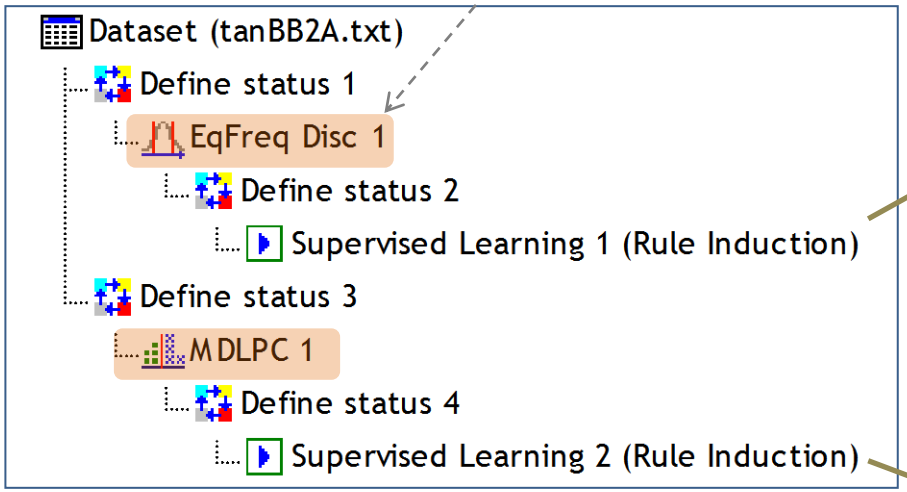
150 instances

4 continuous variables

3 classes

2 intervals according to the equal frequency approach

(it is not really suitable for a 3-classes problem)



Error rate = 21.33%

0.2133				
Confusion matrix				
	Iris-setosa	Iris-versicolor	Iris-virginica	Sum
Iris-setosa	48	2	0	50
Iris-versicolor	1	20	29	50
Iris-virginica	0	0	50	50
Sum	49	22	79	150

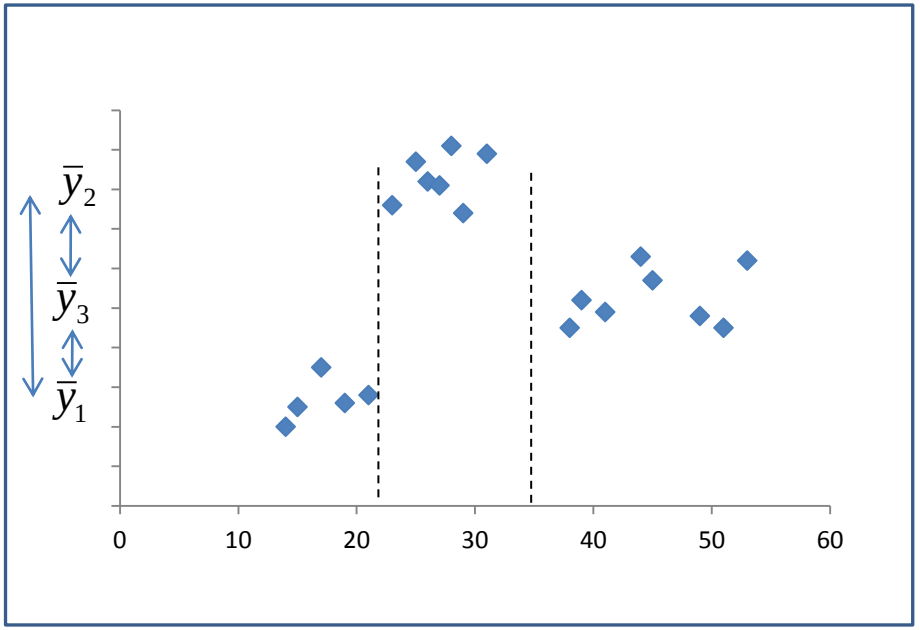
Error rate = 4.67%

0.0467				
Confusion matrix				
	Iris-setosa	Iris-versicolor	Iris-virginica	Sum
Iris-setosa	50	0	0	50
Iris-versicolor	0	44	6	50
Iris-virginica	0	1	49	50
Sum	50	45	55	150

Continuous target attribute

The instances are labeled with a quantitative variable Y

X	Y
14	**2**
15	**2.5**
17	**3.5**
19	**2.6**
21	**2.8**
23	**7.6**
25	**8.7**
26	**8.2**
27	**8.1**
28	**9.1**
29	**7.4**
31	**8.9**
38	**4.5**
39	**5.2**
41	**4.9**
44	**6.3**
45	**5.7**
49	**4.8**
51	**4.5**
53	**6.2**



This is an analysis of variance scheme, but according to the target attribute Y i.e. split X in intervals in order to groups of individuals as homogeneous as possible according Y in each group

Top-down approach – Regression tree

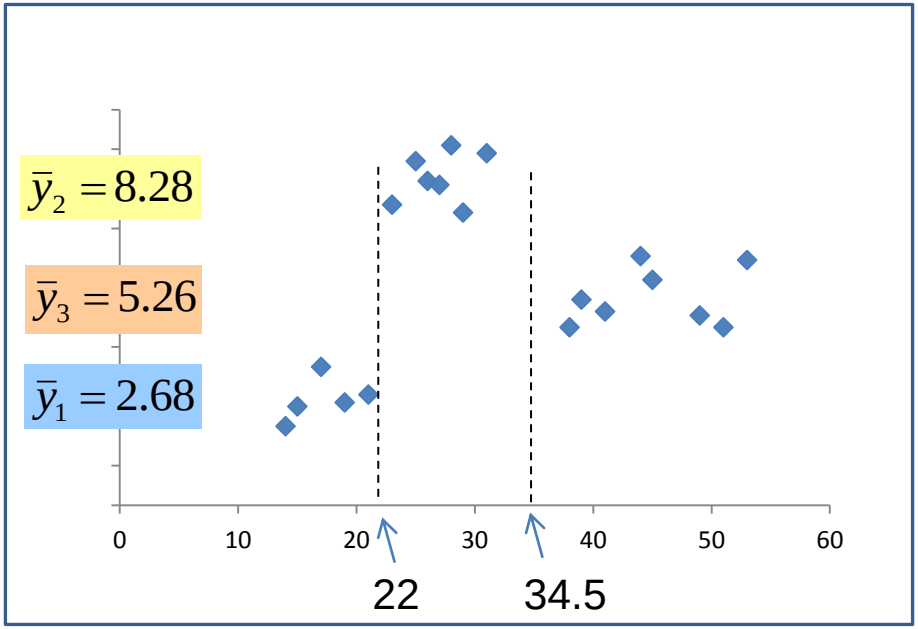
```
> don.reg <- read.xlsx(file="data_discretisation.xlsx",sheetIndex=3)
> print(summary(don.reg))
       X                Y
 Min.   :14.00   Min.   :2.000
 1st Qu.:22.50   1st Qu.:4.250
 Median :28.50   Median :5.450
 Mean   :31.75   Mean   :5.675
 3rd Qu.:41.75   3rd Qu.:7.725
 Max.   :53.00   Max.   :9.100
>
> arbre.reg <- rpart(Y ~ X, data = don.reg, method = "anova", control = rpart.control(minsplit=5,cp=0.07))
> print(arbre.reg)
n= 20

node), split, n, deviance, yval
      * denotes terminal node

1) root 20 101.277500 5.675000
  2) X< 22 5   1.188000 2.680000 *
  3) X>=22 15  40.289330 6.673333
    6) X>=34.5 8   3.658750 5.262500 *
    7) X< 34.5 7   2.508571 8.285714 *
```

Regression tree =

nonlinear regression



Conclusion

Conclusion

The Discretization consists to transform a continuous variable into a discrete attribute, by defining a set of intervals.

The two main issues are: how to determine the number of intervals; how to determine the cut points.

The best discretization is the one performed by a domain expert. Indeed, he takes into account other information than those only provided by the available dataset. But this knowledge is often unavailable.

The data-driven methods are distinguished by their context (supervised or unsupervised); the kind of information they handle; and the strategy used (top-down vs. bottom-up in the most of the cases).

The discretization is part of the learning process. It influences the performance of the subsequent statistical learning process.

References

Tanagra tutorial, « Discretization of continuous features », 2010 ;

http://data-mining-tutorials.blogspot.fr/2010/05/discretization-of-continuous-features.html

Tanagra tutorial, « Discretization and Naive Bayes Classifier », 2008 ;

http://data-mining-tutorials.blogspot.fr/2008/11/discretization-and-naive-bayes.html

F. Muhlenbach, R. Rakotomalala , « Discretization of continuous attributes » ;

Encyclopedia of Data Warehousing and Mining, John Wang Editor, pp. 397-402, 2005.