MUFFAKHAM JAH COLLEGE OF ENGINEERING AND TECHNOLOGY

Banjara Hills, Hyderabad, Telangana

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Data Mining Laboratory Manual

Academic Year 2016-2017

Table of Contents

I Contents

1. Vision of the Institution i
2. Mission of the Institution i
3. Department Vision ii
4. Department Mission ii
5. Programme Education Objectives iii
6. Programme Outcomes iv
7. Programme Specific Outcomes v
8. Introduction vi

II Programs

1. Perform data preprocessing using weka 1
2. Obtain decision tree for different data sets using WEKA 28
3. Classification algorithms using WEKA 37
4. Obtain The Association Rules For Given Dataset Using WEKA. 45
5. Perform data transformations using an ETL tool. 49
6. PROGRAM-6: Write a program to implement star schema. 59
7. PROGRAM-6A: write a program to implement K-Means algorithm. . . 64
8. PROGRAM-6B: K-Means algorithm for two dimension data items. . . 67
9. a program to implement k-mediod algorithm 70
10. CASE STUDY 1- K-NEAREST NEIGHBOR 74
11. CASE STUDY 2 - KDD PROCESS 77
12. DECISION TREE INDUCTION 81
13. Case Study On Apriori 85

Part I

Contents

1. Vision of the Institution

To be part of universal human quest for development and progress by contributing high calibre, ethical and socially responsible engineers who meet the global challenge of building modern society in harmony with nature.

2. Mission of the Institution

- To attain excellence in imparting technical education from undergraduate through doctorate levels by adopting coherent and judiciously coordinated curricular and co-curricular programs.
- To foster partnership with industry and government agencies through collaborative research and consultancy.
- To nurture and strengthen auxiliary soft skills for overall development and improved employability in a multi-cultural work space.
- To develop scientific temper and spirit of enquiry in order to harness the latent innovative talents.
- To develop constructive attitude in students towards the task of nation building and empower them to become future leaders
- To nourish the entrepreneurial instincts of the students and hone their business acumen.
- To involve the students and the faculty in solving local community problems through economical and sustainable solutions.

3. Department Vision

To contribute competent computer science professionals to the global talent pool to meet the constantly evolving societal needs.

4. Department Mission

Mentoring students towards a successful professional career in a global environment through quality education and soft skills in order to meet the evolving societal needs.

5. Programme Education Objectives

1. Graduates will demonstrate technical skills and leadership in their chosen fields of employment by solving real time problems using current techniques and tools.

2. Graduates will communicate effectively as individuals or team members and be successful in the local and global cross cultural working environment.

3. Graduates will demonstrate lifelong learning through continuing education and professional development.

4. Graduates will be successful in providing viable and sustainable solutions within societal, professional, environmental and ethical contexts

6. Programme Outcomes

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Lifelong learning:** Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

7. Programme Specific Outcomes

The graduates will be able to:

PSO1: Demonstrate understanding of the principles and working of the hardware and software aspects of computer systems.

PSO2: Use professional engineering practices, strategies and tactics for the development, operation and maintenance of software

PSO3: Provide effective and efficient real time solutions using acquired knowledge in various domains.

8. Introduction

Data Mining is defined as the procedure of extracting information from huge sets of data. In other words, we can say that data mining is mining knowledge from data. The tutorial starts off with a basic overview and the terminologies involved in data mining and then gradually moves on to cover topics such as knowledge discovery, query language, classification and prediction, decision tree induction, cluster analysis, and how to mine the Web.

Audience

This tutorial has been prepared for computer science graduates to help them understand the basic-to-advanced concepts related to data mining.

Prerequisites

Before proceeding with this tutorial, you should have an understanding of the basic database concepts such as schema, ER model, Structured Query language and a basic knowledge of Data Warehousing concepts.

Applications

Data mining is highly useful in the following domains

- Market Analysis and Management
- Corporate Analysis & Risk Management
- Fraud Detection
- Production Control
- Science Exploration

Tools

Audience
This tutorial has been prepared for computer science graduates to help them understand the basic-to-advanced concepts related to data mining.
Prerequisites
Before proceeding with this tutorial, you should have an understanding of the basic database concepts such as schema, ER model, Structured Query language and a basic knowledge of Data Warehousing concepts.
Applications
Data mining is highly useful in the following domains

- Market Analysis and Management
- Corporate Analysis & Risk Management

- Fraud Detection
- Production Control
- Science Exploration

Tools

Large variety of DM software is now available. Some more widely used software is:

- IBM - Intelligent Miner
- Weka(Waikato Environment of Knowledge Analysis)
- Cognos
- Informatica
- SAS - Enterprise Miner
- Silicon Graphics Mine Set
- Oracle - Thinking Machines - Darwin
- Angoss knowledge SEEKER

Software and Hardware Requirements

HARDWARE SPECIFICATIONS:

```
Processor        :   Pentium Dual Core

RAM              :   1GB

Hard Disk        :   160GB
```

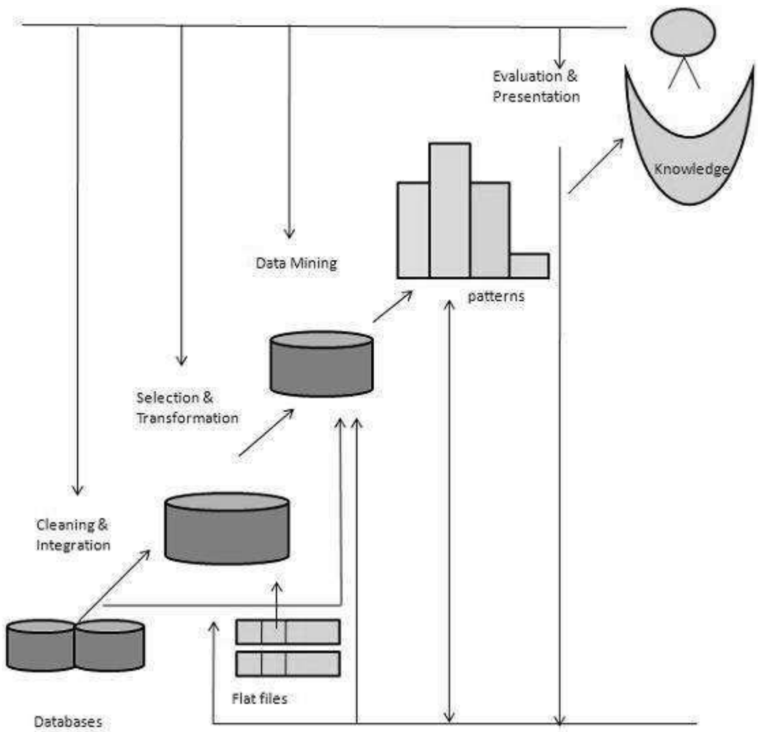
SOFTWARE SPECIFICATIONS:

```
Operating System     :  Windows XP

Utility Software     :  Java 5.0 and above

Data Mining Software : weka 3.6.13
```

What Motivated Data Mining

- Data collection and data availability
  - Automated data collection tools, Database systems, Web, computerized society
- Major sources of abundant data
  - Business: Web, e-commerce, transactions, stocks,
  - Science: Remote sensing, bioinformatics, scientific simulation,
  - Society and everyone: news, digital cameras, YouTube

Architecture of Data Mining Process



Part II

Programs

Program 1

DATA PROCESSING USING WEKA

Problem Definition

Perform data preprocessing using weka

Problem Description

Data Preprocessing in Data mining is a technique that involves transforming raw data into an understandable form. It is used to filter out the data in the desired form by using techniques like cleaning, normalization, transformation, feature extraction and selection, etc .

- Data in the real world is dirty
  - **incomplete:** lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    e.g., occupation=" "
  - **noisy:** containing errors or outliers
    e.g., Salary=-10
  - **inconsistent:** containing discrepancies in codes or names e.g., Age=42 Birthday=03/07/1997
    e.g., Was rating 1,2,3, now rating A, B, C
    e.g., discrepancy between duplicate records

Following is the demonstration with respect to preprocessing commands:

1. **ADD:**

   DESCRIPTION: An instance filter that adds a new attribute to the dataset. The new attribute will contain all missing values.

   OPTIONS

   Attribute Index – The position (starting from 1) where the attribute will be inserted (first and last are valid indices).

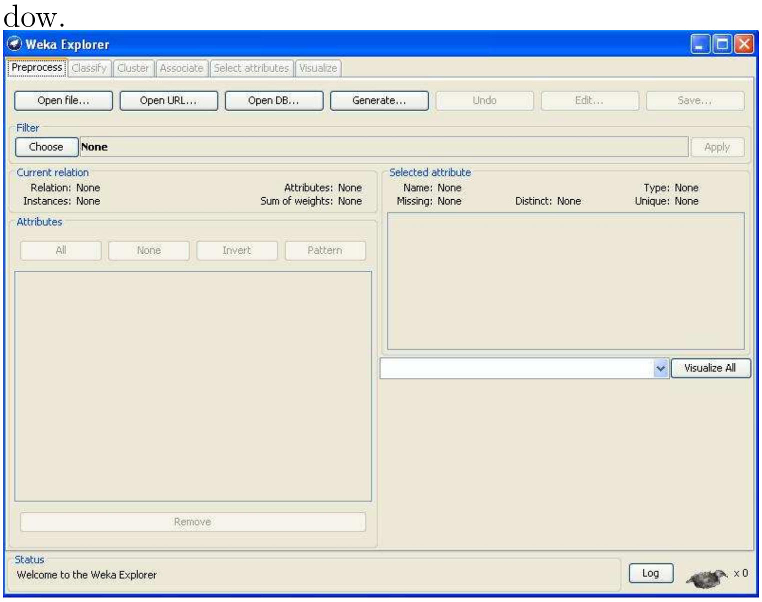
   Attribute Name – Set the new attribute's name.

   Attribute Type – Defines the type of the attribute to generate.

STEPS

Figure 1: Start Weka you get the Weka GUI chooser window.

Figure 2: Click on the Explorer button and you get the Weka Knowledge Explorer window.



You see here the default parameters of this filter. Click on More to get more

Figure 3: Click on the Open File. button and open an ARFF file (try it first with an example supplied in Weka-3-6/data, e.g. Weather.arff). You get the following:

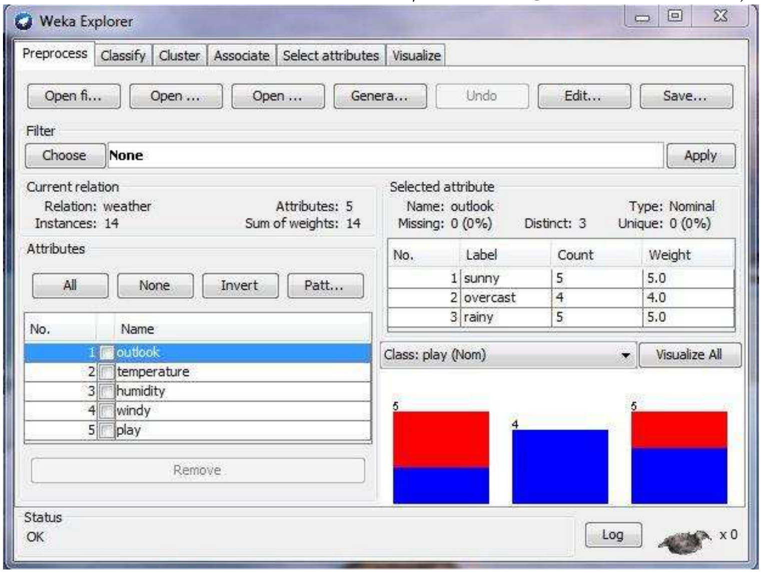
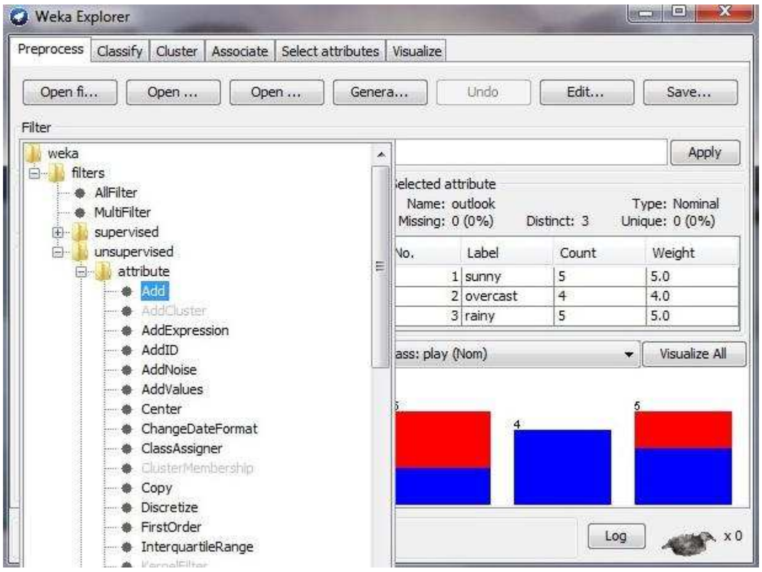


Figure 4: VALIDATION : Click on Choose and select filters/unsupervised/attribute/Add.

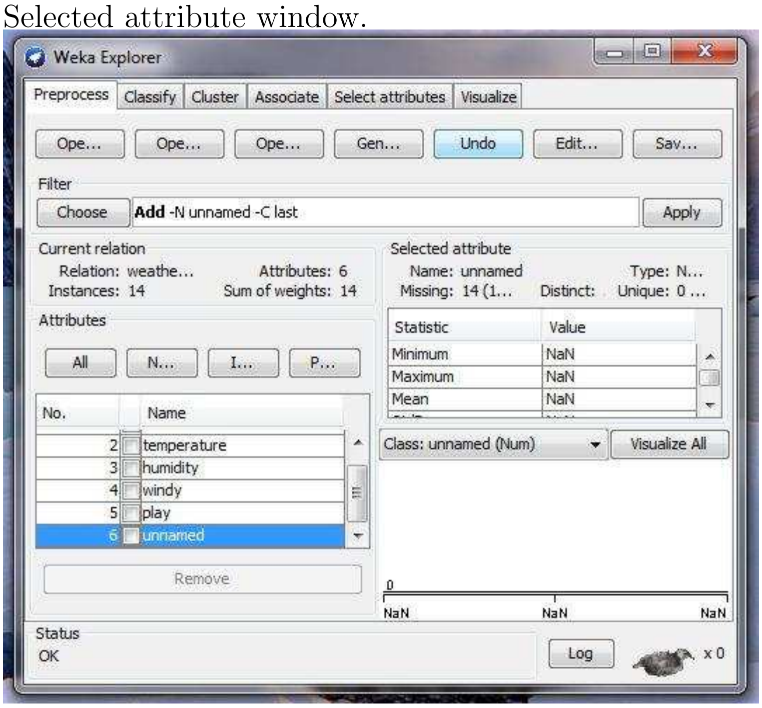


information about these parameters.

Try other parameters for the filter and see how the Addition changes. Dont forget to reload the original (numeric) relation or Undo the Addition before applying another one.

Figure 5: Then click on the area right of the Choose button. You get the following:

Figure 6: Click on the Apply button to do the Addition and see how it is Added in the Selected attribute window.



2. **Discretization :**

   DESCRIPTION: An instance filter that discretizes a range of numeric attributes in the dataset into nominal attributes. Discretization is by simple binning. Skips the class attributes if set.

OPTIONS

Attribute Indices – Specify range of attributes to act on. This is a comma separated list of attribute indices, with "first" and "last" valid values. Specify an inclusive range with "-". E.g: "first-3,5,6-10,last".

bins – Number of bins.

Desired Weight Of Instances Per Interval – Sets the desired weight of instances per interval for equal-frequency binning.

Find Num Bins – Optimize number of equal-width bins using leave-one-out. Doesn't work for equal-frequency binning

ignore Class – The class index will be unset temporarily before the filter is applied.

Invert Selection – Set attribute selection mode. If false, only selected (numeric) attributes in the range will be discretized; if true, only non-selected attributes will be discretized.

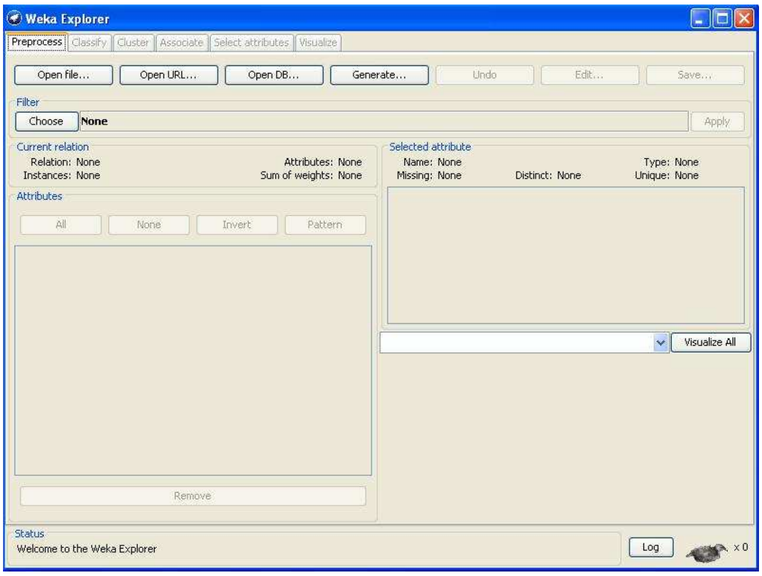
Make Binary – Make resulting attributes binary.

Use Equal Frequency – If set to true, equal-frequency binning will be used instead of equal-width binning.

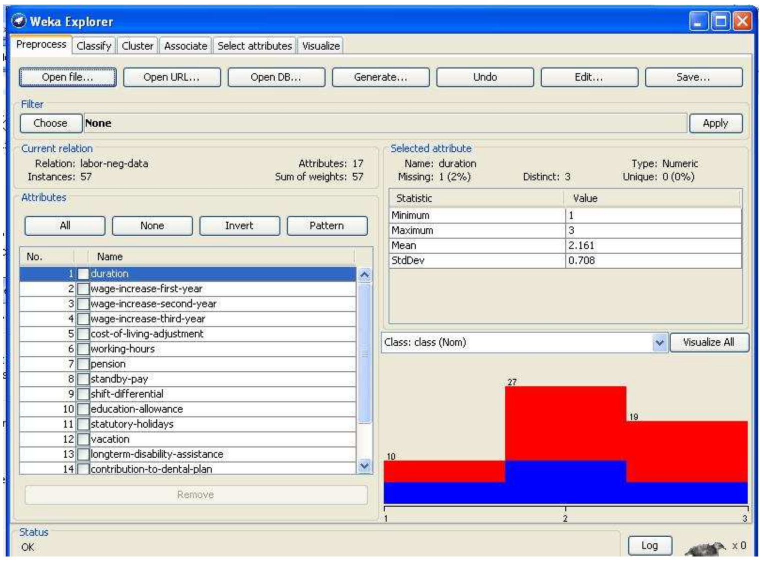
STEPS

a. Start Weka you get the Weka GUI chooser window.

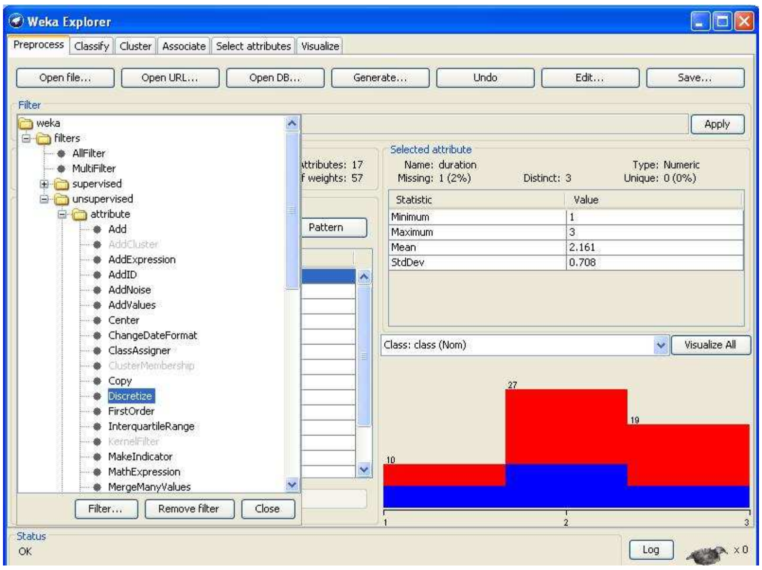
b. Click on the Explorer button and you get the Weka Knowledge Explorer window.



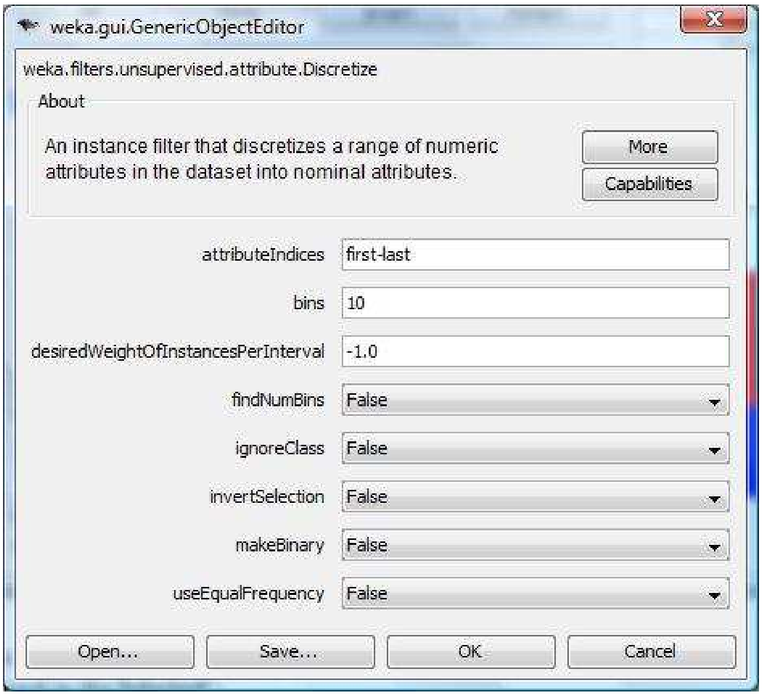
c. **VALIDATION :** Click on the Open File. button and open an ARFF file (try it first with an example supplied in Weka-3-6/data, e.g. Weather.arff). You get the following:


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Current relation
Relation: labor-neg-data
Instances: 57
Attributes: 17
Sum of weights: 57
Selected attribute
Name: duration
Missing: 1 (2%)
Distinct: 3
Type: Numeric
Unique: 0 (0%)
Attributes
All
None
Invert
Pattern
No.
Name
1 duration
2 wage-increase-first-year
3 wage-increase-second-year
4 wage-increase-third-year
5 cost-of-living-adjustment
6 working-hours
7 pension
8 standby-pay
9 shift-differential
10 education-allowance
11 statutory-holidays
12 vacation
13 longterm-disability-assistance
14 contribution-to-dental-plan
Remove
Statistic
Value
Minimum
1
Maximum
3
Mean
2.161
StdDev
0.708
Class: class (Nom)
Visualize All
27
19
10
1
2
3
Status
OK
Log
x 0

d. **VALIDATION :** Click on Choose and select filters/unsupervised/ attribute/Discretize.

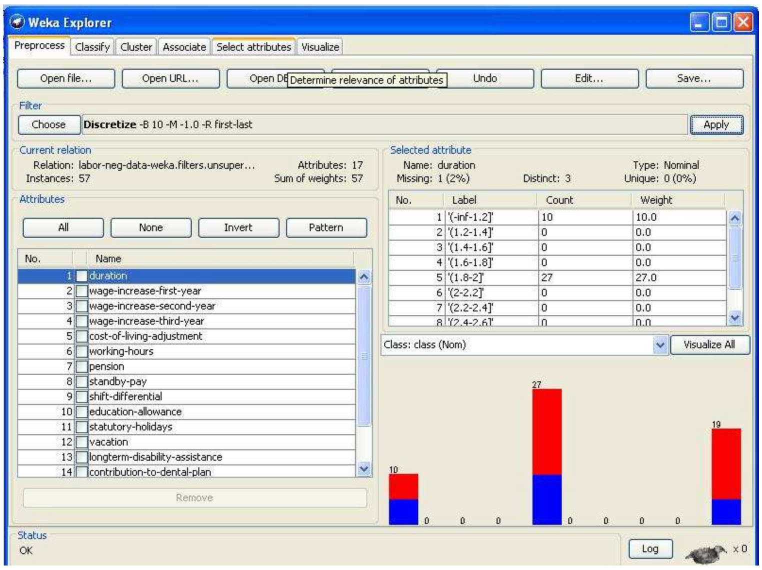


e. Then click on the area right of the Choose button. You get the following:



You see here the default parameters of this filter. Click on More to get more information about these parameters.

f. **VALIDATION :** Click on the Apply button to do the discretization. Then select one of the original numeric attributes (e.g. temperature) and see how it is discretized in the Selected attribute window..



g. Try other parameters for the filter and see how the discretization changes. Dont forget to reload the original (numeric) relation or Undo the discretization before applying another one.

3. <u>Normalize</u>:

DESCRIPTION:

Normalizes all numeric values in the given data set (apart from the class attribute, if set). The resulting values are by default in [0,1] for the data used to compute the normalization intervals. But with the scale and translation parameters one can change that.
e.g., with scale = 2.0 and translation = -1.0 you get values in the range [-1,+1].

OPTIONS

ignoreClass – The class index will be unset temporarily before the filter is applied.

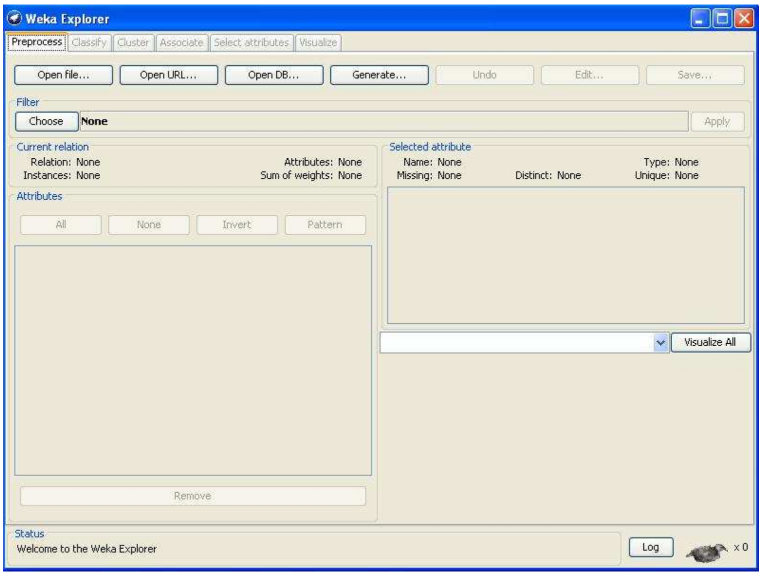
scale – The factor for scaling the output range (default: 1).

translation – The translation of the output range (default: 0).

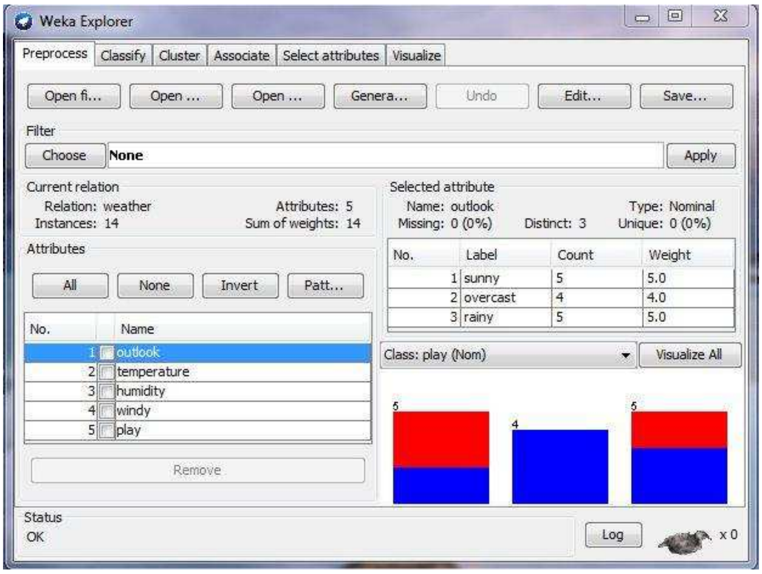
STEPS

a. Start Weka you get the Weka GUI chooser window.

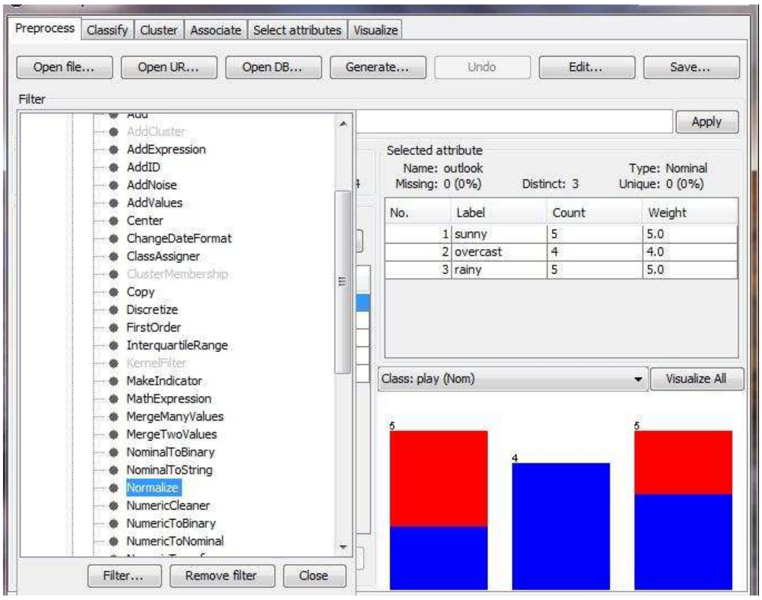
b. Click on the Explorer button and you get the Weka Knowledge Explorer window.



c. Click on the Open File. button and open an ARFF file (try it first with an example supplied in Weka-3-6/data, e.g. whether.arff). You get the following:



d. Click on Choose and select filters/unsupervised/attribute/Normalize.

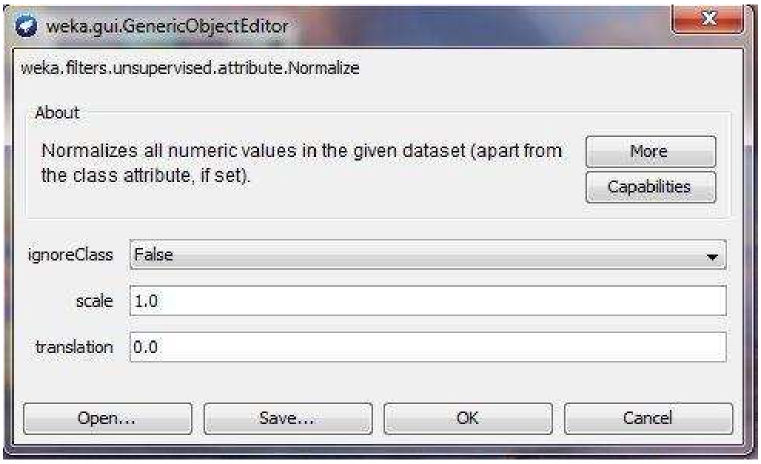


Viewer

Relation: weather

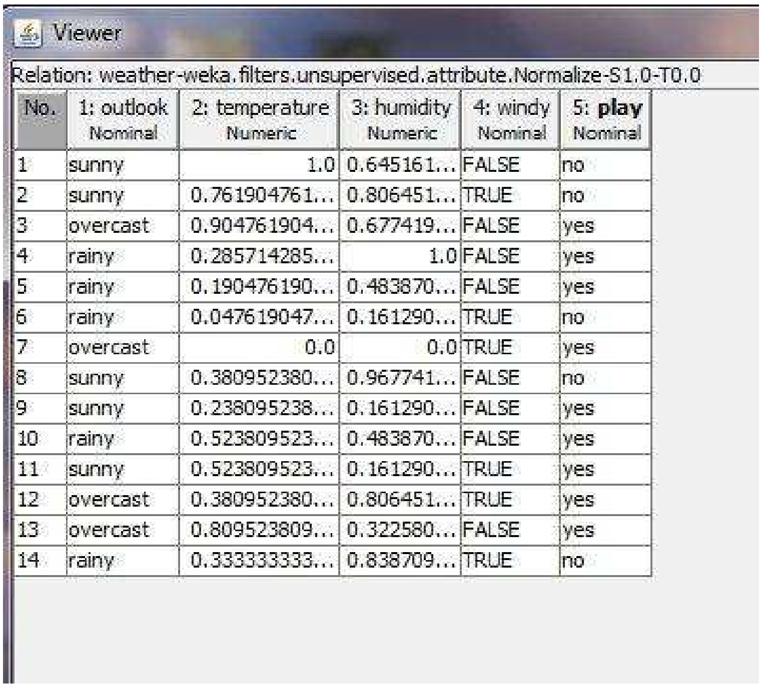
No.	1: outlook Nominal	2: temperature Numeric	3: humidity Numeric	4: windy Nominal	5: play Nominal
1	sunny	85.0	85.0	FALSE	no
2	sunny	80.0	90.0	TRUE	no
3	overcast	83.0	86.0	FALSE	yes
4	rainy	70.0	96.0	FALSE	yes
5	rainy	68.0	80.0	FALSE	yes
6	rainy	65.0	70.0	TRUE	no
7	overcast	64.0	65.0	TRUE	yes
8	sunny	72.0	95.0	FALSE	no
9	sunny	69.0	70.0	FALSE	yes
10	rainy	75.0	80.0	FALSE	yes
11	sunny	75.0	70.0	TRUE	yes
12	overcast	72.0	90.0	TRUE	yes
13	overcast	81.0	75.0	FALSE	yes
14	rainy	71.0	91.0	TRUE	no

e. Then click on the area right of the Choose button. You get the following:



You see here the default parameters of this filter. Click on More to get more information about these parameters.

f. **VALIDATION :** Click on the Apply button to do the normalization. Then select edit tab to view data and see how it is normalized in the data window.



Viewer

Relation: weather-weka.filters.unsupervised.attribute.Normalize-S1.0-T0.0

No.	1: outlook Nominal	2: temperature Numeric	3: humidity Numeric	4: windy Nominal	5: play Nominal
1	sunny	1.0	0.645161...	FALSE	no
2	sunny	0.761904761...	0.806451...	TRUE	no
3	overcast	0.904761904...	0.677419...	FALSE	yes
4	rainy	0.285714285...	1.0	FALSE	yes
5	rainy	0.190476190...	0.483870...	FALSE	yes
6	rainy	0.047619047...	0.161290...	TRUE	no
7	overcast	0.0	0.0	TRUE	yes
8	sunny	0.380952380...	0.967741...	FALSE	no
9	sunny	0.238095238...	0.161290...	FALSE	yes
10	rainy	0.523809523...	0.483870...	FALSE	yes
11	sunny	0.523809523...	0.161290...	TRUE	yes
12	overcast	0.380952380...	0.806451...	TRUE	yes
13	overcast	0.809523809...	0.322580...	FALSE	yes
14	rainy	0.333333333...	0.838709...	TRUE	no

g. Try other parameters for the filter and see how the normalization changes. Dont forget to reload the original (numeric) relation or Undo the normalization before applying another one.

4. <u>Remove</u>:

DESCRIPTION:

A filter that removes a range of attributes from the dataset. Will re-order the remaining attributes if invert matching sense is turned on and the attribute column indices are not specified in ascending order.

OPTIONS

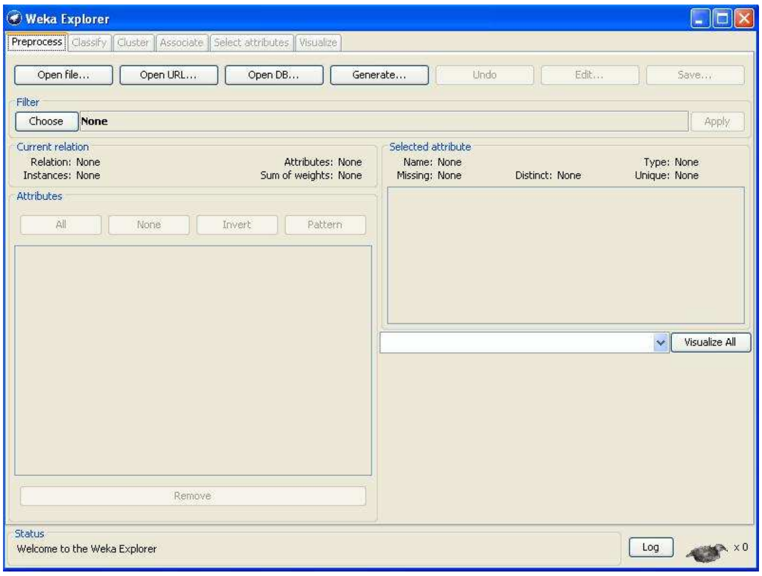
Attribute Indices – Specify range of attributes to act on. This is a comma separated list of attribute indices, with "first" and "last" valid values. Specify an inclusive range with "-". E.g: "first-3,5,6-10,last".

Invert Selection – Determines whether action is to select or delete. If set to true, only the specified attributes will be kept; If set to false, specified attributes will be deleted.

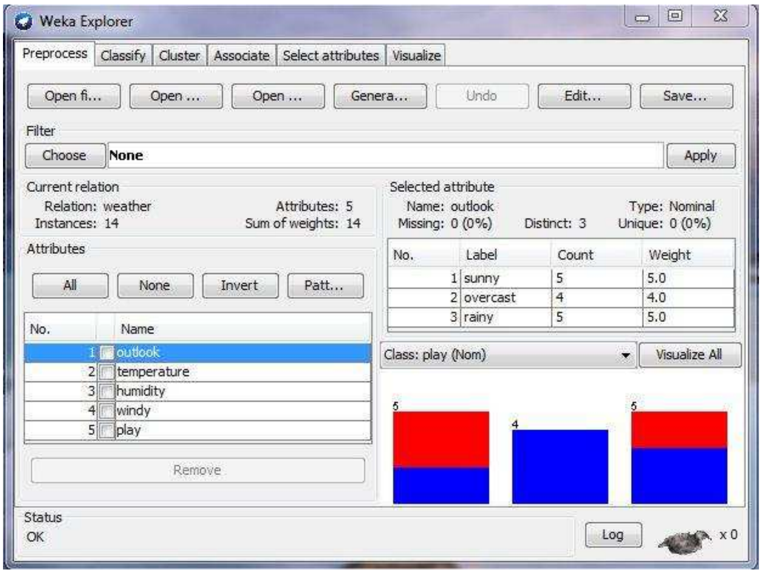
STEPS

a. Start Weka you get the Weka GUI chooser window.

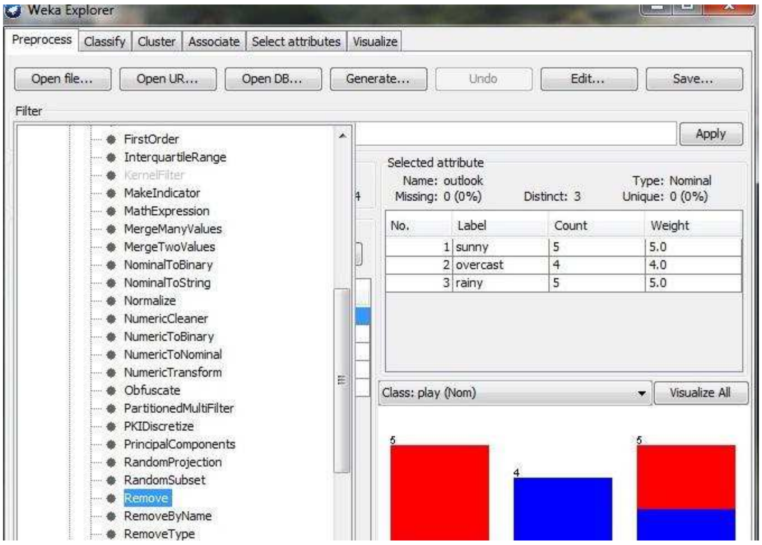
b. Click on the Explorer button and you get the Weka Knowledge Explorer window.



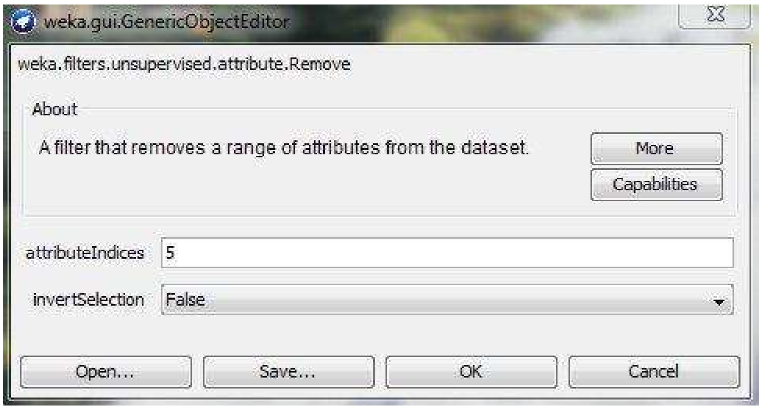
c. Click on the Open File. button and open an ARFF file (try it first with an example supplied in Weka-3-6/data, e.g. Weather.arff). You get the following:



d. Click on Choose and select filters/unsupervised/attribute/Remove.

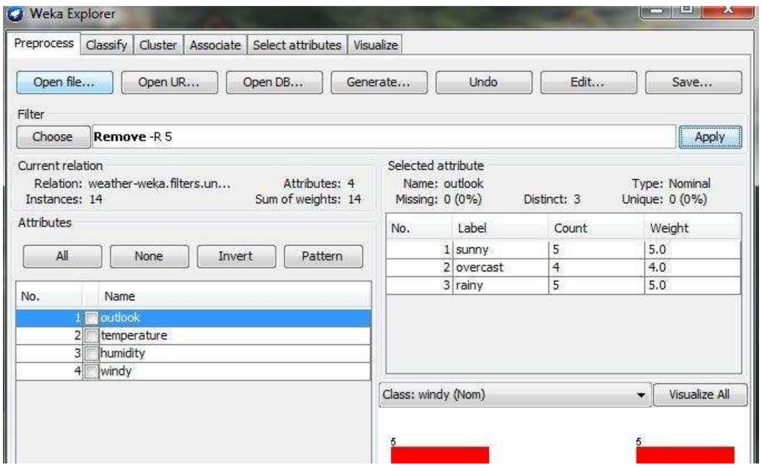


e. Then click on the area right of the Choose button. You get the following:



f. **VALIDATION :** Click on the Apply button to do the remove.

You see here the default parameters of this filter. Enter the Indices of attribute to be remove Click on more to get more information about these parameters.



g. Try other parameters for the filter and see how the remove changes. Dont forget to reload the original (numeric) relation or Undo the remove before applying another one.

h. **Replace Missing Values**

Description: Replaces all missing values for nominal and numeric attributes in a dataset with the modes and means from the training data.

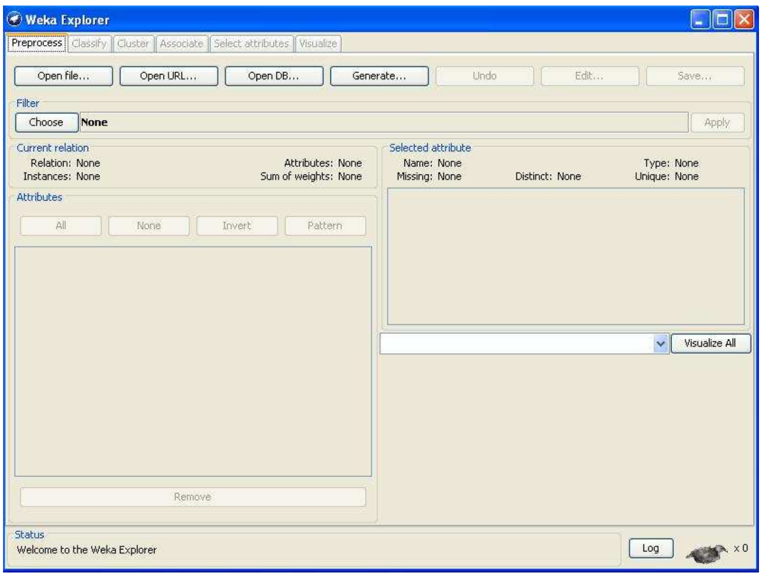
OPTIONS

Ignore Class – The class index will be unset temporarily before the filter is applied.

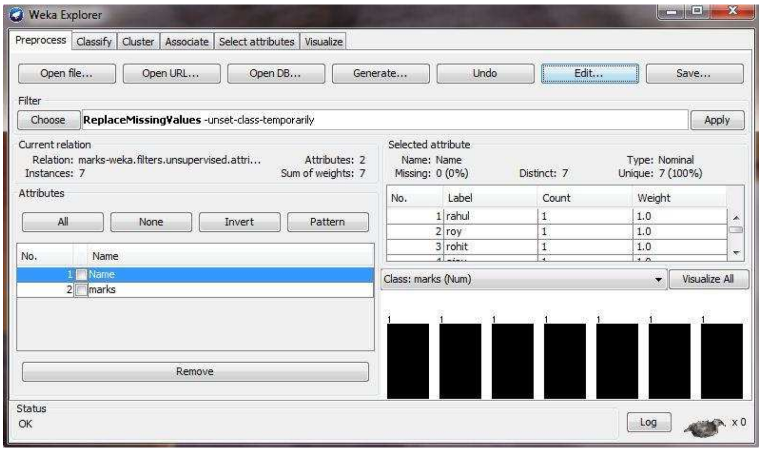
STEPS

i. Start Weka you get the Weka GUI chooser window.

ii. Click on the Explorer button and you get the Weka Knowledge Explorer window.

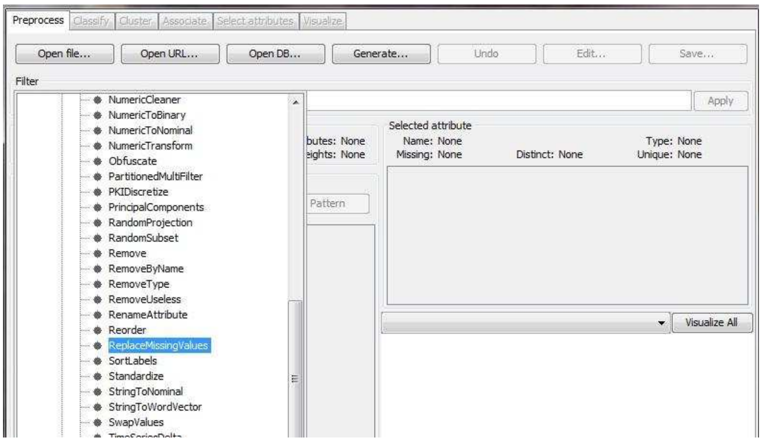


iii. Click on the Open File. button and open an ARFF file (try it first with an example supplied in Weka-3-6/data, e.g. weather.arff). You get the following:

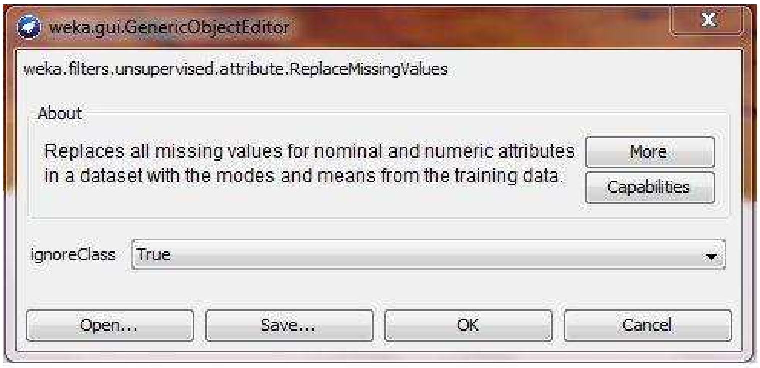


No.	1: Name Nominal	2: marks Numeric
1	rahul	
2	roy	55.0
3	rohit	22.0
4	ajay	87.0
5	asad	
6	asha	65.0
7	seema	

iv. Click on Choose and select filters/ unsupervised/attribute/ReplaceMissingValues.



v. Then click on the area right of the Choose button. You get the following:



You see here the default parameters of this filter. Click on more to get more information about these parameters.

vi. **VALIDATION :** Click on the Apply button to do the Replace Missing Values. Then select edit tab to view data and see how it Replaced missing values in the data window.

vii. Try other parameters for the filter and see how the replace values changes. Dont forget to reload the original (numeric) relation or Undo the replaced before applying another one.

i. **Standardize:**

DESCRIPTION:

Standardizes all numeric attributes in the given dataset to have zero mean and unit variance (apart from the class attribute, if set).

OPTIONS

Ignore Class – The class index will be unset temporarily before the filter is applied.

STEPS

i. Start Weka you get the Weka GUI chooser window.

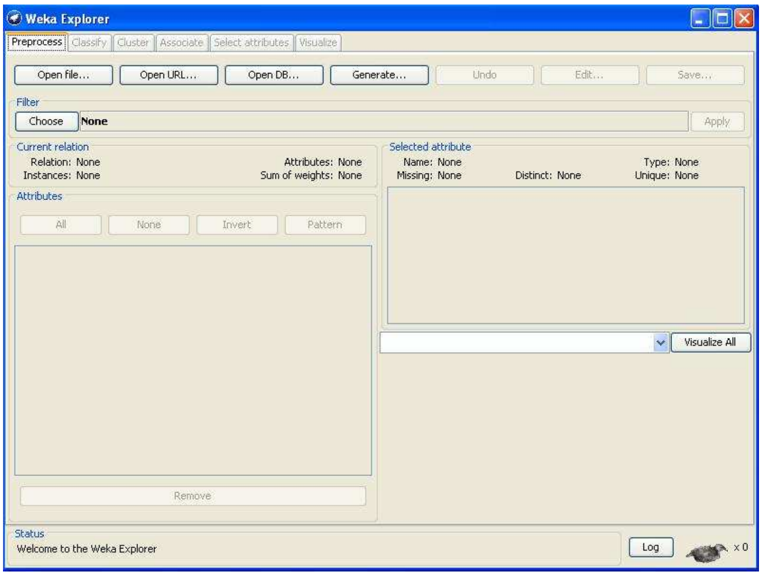
ii. Click on the Explorer button and you get the Weka Knowledge Explorer window.

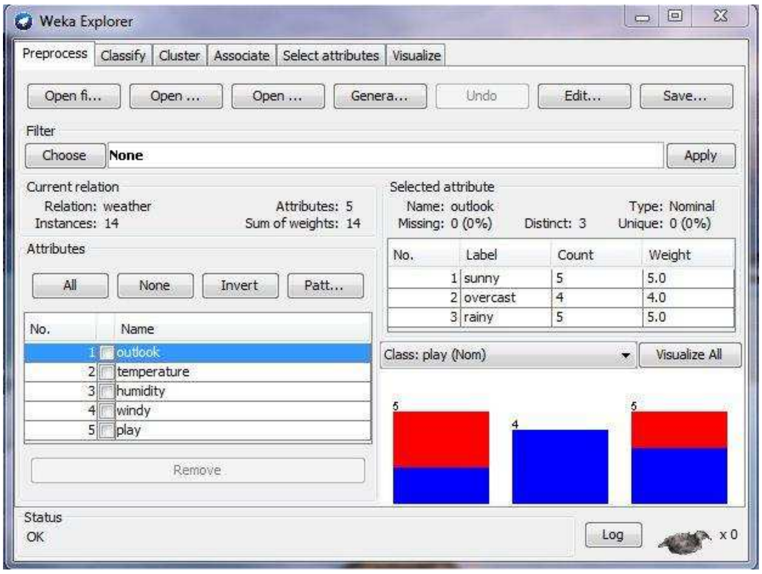
iii. Click on the Open File. button and open an ARFF file (try it first with an example supplied in Weka-3-6/data, e.g. Weather.arff). You get the following:

iv. Click on Choose and select filters/unsupervised/attribute/Standardize.

v. Then click on the area right of the Choose button. You get the following:

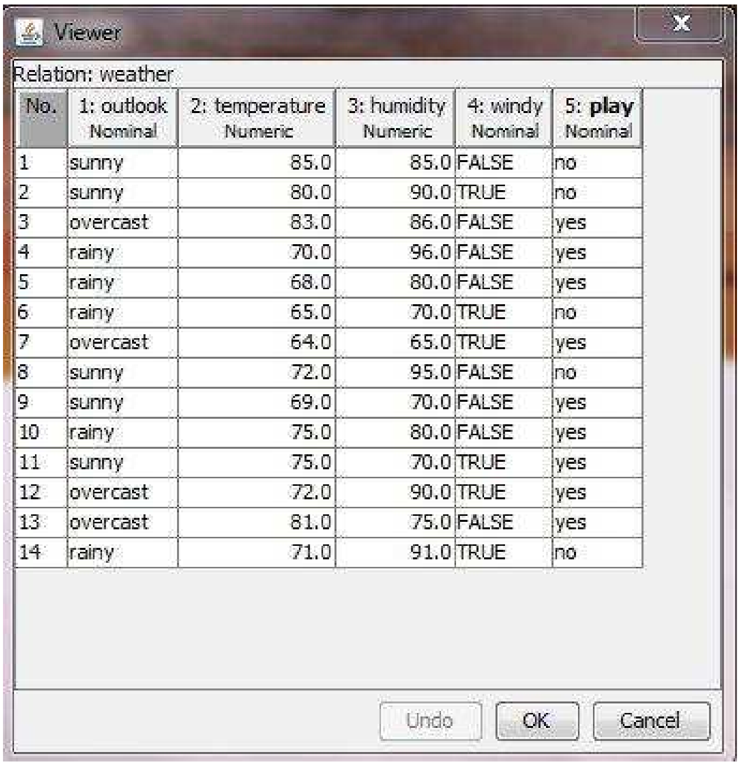
You see here the default parameters of this filter. Enter the Indices of attribute to be remove Click on more to get more information about these parameters.



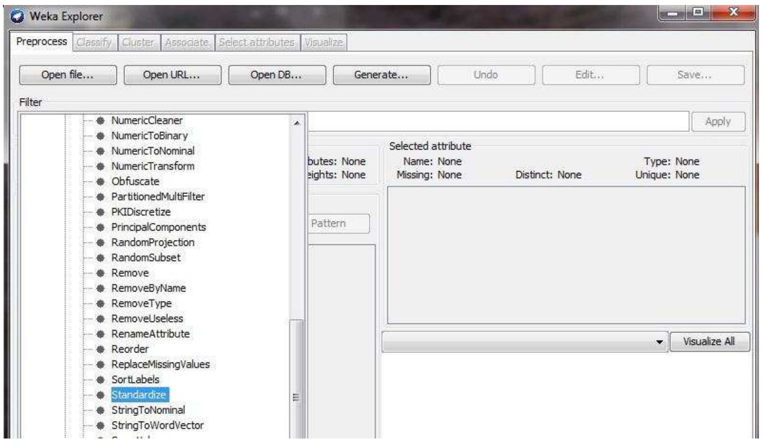


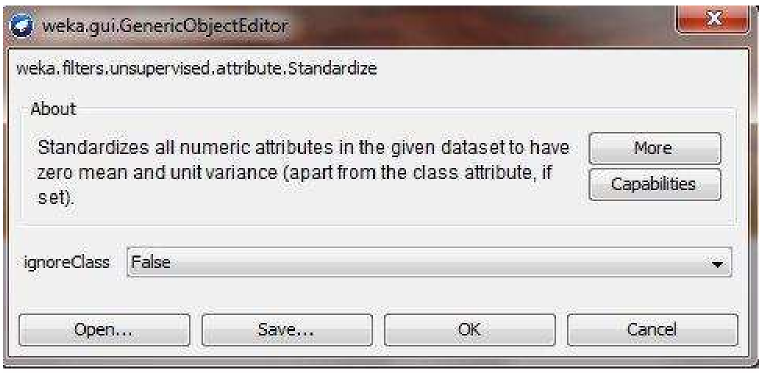
vi. VALIDATION : Click on the Apply button to do the Standardization. Then select edit tab to view data and see how it standard the values in the data window.

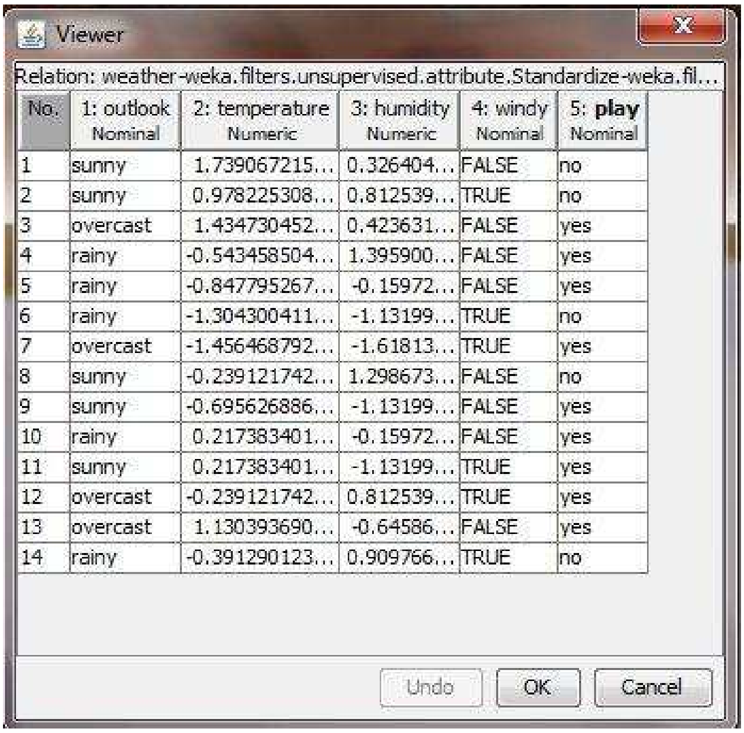
vii. Try other parameters for the filter and see how the standardize changes. Dont forget to reload the original (numeric) relation or Undo the standardize before applying another one.



No.	1: outlook Nominal	2: temperature Numeric	3: humidity Numeric	4: windy Nominal	5: play Nominal
1	sunny	85.0	85.0	FALSE	no
2	sunny	80.0	90.0	TRUE	no
3	overcast	83.0	86.0	FALSE	yes
4	rainy	70.0	96.0	FALSE	yes
5	rainy	68.0	80.0	FALSE	yes
6	rainy	65.0	70.0	TRUE	no
7	overcast	64.0	65.0	TRUE	yes
8	sunny	72.0	95.0	FALSE	no
9	sunny	69.0	70.0	FALSE	yes
10	rainy	75.0	80.0	FALSE	yes
11	sunny	75.0	70.0	TRUE	yes
12	overcast	72.0	90.0	TRUE	yes
13	overcast	81.0	75.0	FALSE	yes
14	rainy	71.0	91.0	TRUE	no







Viewer

Relation: weather-weka.filters.unsupervised.attribute.Standardize-weka.fil...

No.	1: outlook Nominal	2: temperature Numeric	3: humidity Numeric	4: windy Nominal	5: play Nominal
1	sunny	1.739067215...	0.326404...	FALSE	no
2	sunny	0.978225308...	0.812539...	TRUE	no
3	overcast	1.434730452...	0.423631...	FALSE	yes
4	rainy	-0.543458504...	1.395900...	FALSE	yes
5	rainy	-0.847795267...	-0.15972...	FALSE	yes
6	rainy	-1.304300411...	-1.13199...	TRUE	no
7	overcast	-1.456468792...	-1.61813...	TRUE	yes
8	sunny	-0.239121742...	1.298673...	FALSE	no
9	sunny	-0.695626886...	-1.13199...	FALSE	yes
10	rainy	0.217383401...	-0.15972...	FALSE	yes
11	sunny	0.217383401...	-1.13199...	TRUE	yes
12	overcast	-0.239121742...	0.812539...	TRUE	yes
13	overcast	1.130393690...	-0.64586...	FALSE	yes
14	rainy	-0.391290123...	0.909766...	TRUE	no

Undo OK Cancel

Program 2

OBTAIN DECISION TREE FOR DIFFERENT DATA SETS USING WEKA

Problem Definition

Obtain decision tree for different data sets using WEKA

Problem Description

Class for generating a pruned or unpruned C4.5 decision tree. Ross Quinlan (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA.

OPTIONS

Binary Splits – Whether to use binary splits on nominal attributes when building the trees.

debug – If set to true, classifier may output additional info to the console.

Min Num Obj – The minimum number of instances per leaf.

Num Folds – Determines the amount of data used for reduced-error pruning. One fold is used for pruning, the rest for growing the tree.

Reduced Error Pruning – Whether reduced-error pruning is used instead of C.4.5 pruning.

Save Instance Data – Whether to save the training data for visualization.

seed – The seed used for randomizing the data when reduced-error pruning is used.

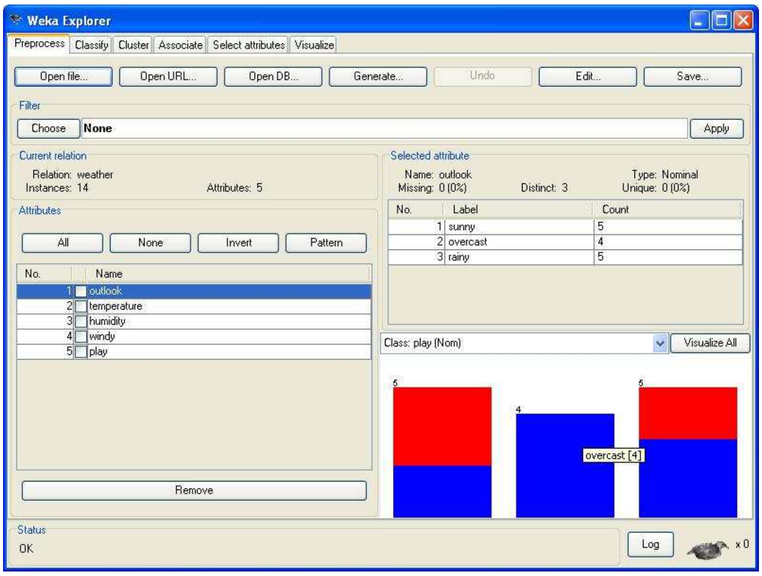
Sub tree Raising – Whether to consider the subtree raising operation when pruning.

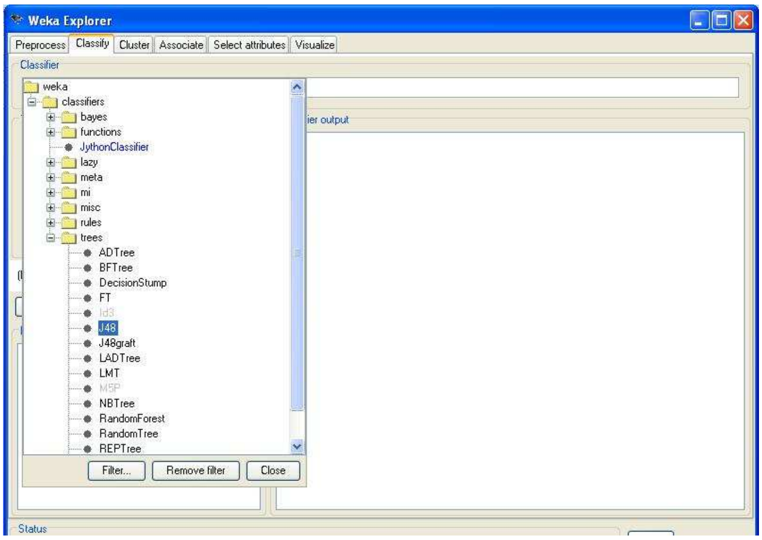
Un pruned – Whether pruning is performed.

Use Laplace – Whether counts at leaves are smoothed based on Laplace.

Click on classify, then choose

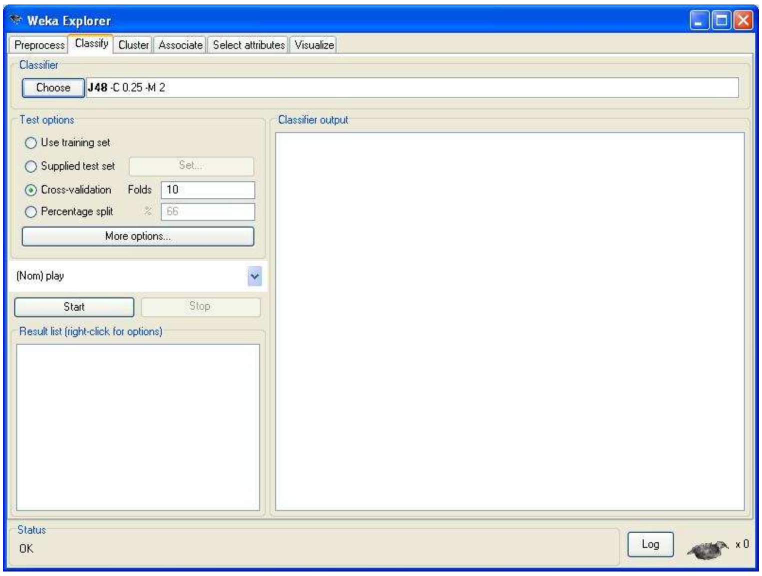
Click on start

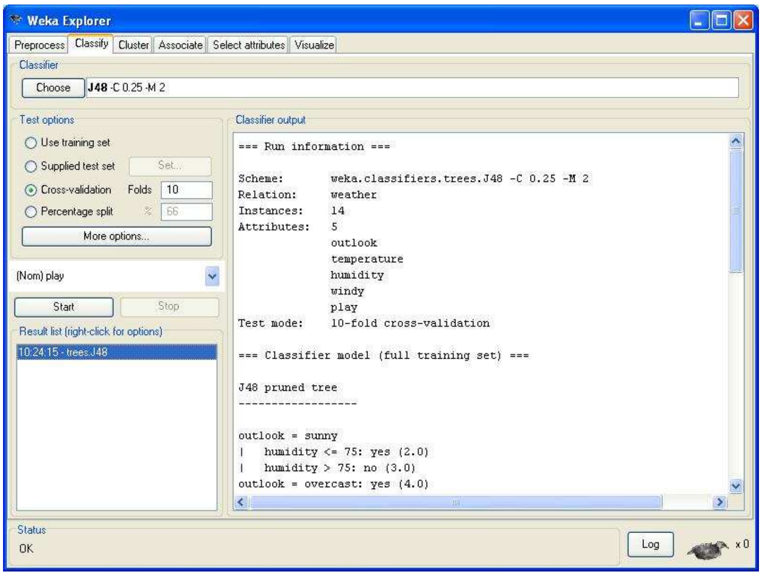


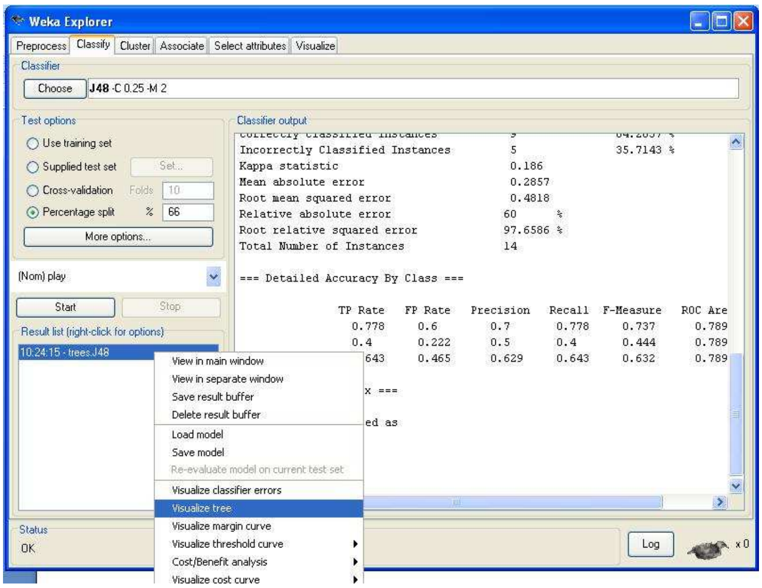


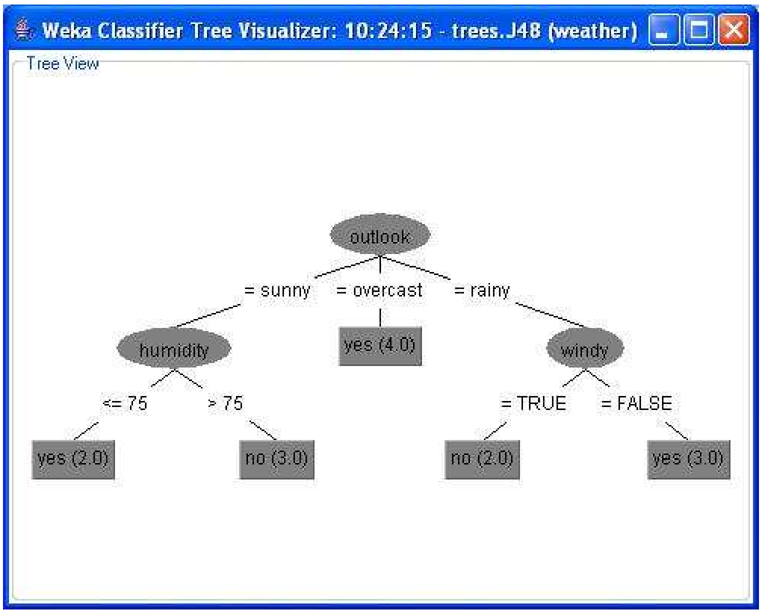
Click on percentage split

Click on 102415-treesJ48


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) play
Start
Stop
Result list (right-click for options)
Classifier output
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) play
Start
Stop
Result list (right-click for options)
10:24:15 - trees.J48
Classifier output
=== Run information ===
Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: weather
Instances: 14
Attributes: 5
outlook
temperature
humidity
windy
play
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
----
outlook = sunny
| humidity <= 75: yes (2.0)
| humidity > 75: no (3.0)
outlook = overcast: yes (4.0)
Status
OK
Log
x 0

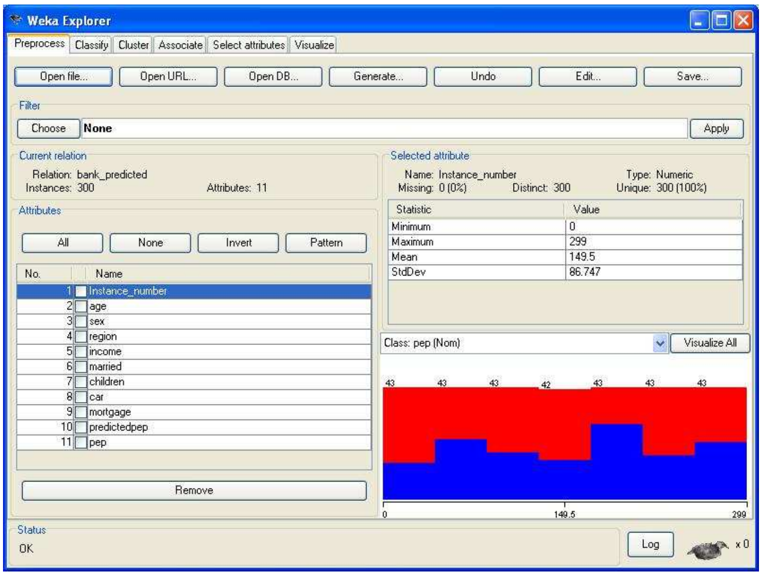

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) play
Start
Stop
Result list (right-click for options)
10:24:15 - trees.J48
Classifier output
Incorrectly Classified Instances 5 35.7143 %
Kappa statistic 0.186
Mean absolute error 0.2857
Root mean squared error 0.4818
Relative absolute error 60 %
Root relative squared error 97.6586 %
Total Number of Instances 14
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Are
0.778 0.6 0.7 0.778 0.737 0.789
0.4 0.222 0.5 0.4 0.444 0.789
643 0.465 0.629 0.643 0.632 0.789
View in main window
View in separate window
Save result buffer
Delete result buffer
Load model
Save model
Re-evaluate model on current test set
Visualize classifier errors
Visualize tree
Visualize margin curve
Visualize threshold curve
Cost/Benefit analysis
Visualize cost curve
Status
OK
Log
x 0



Weka Classifier Tree Visualizer: 10:24:15 - trees.J48 (weather)
Tree View
outlook
= sunny
= overcast
= rainy
humidity
yes (4.0)
windy
<= 75
> 75
= TRUE
= FALSE
yes (2.0)
no (3.0)
no (2.0)
yes (3.0)

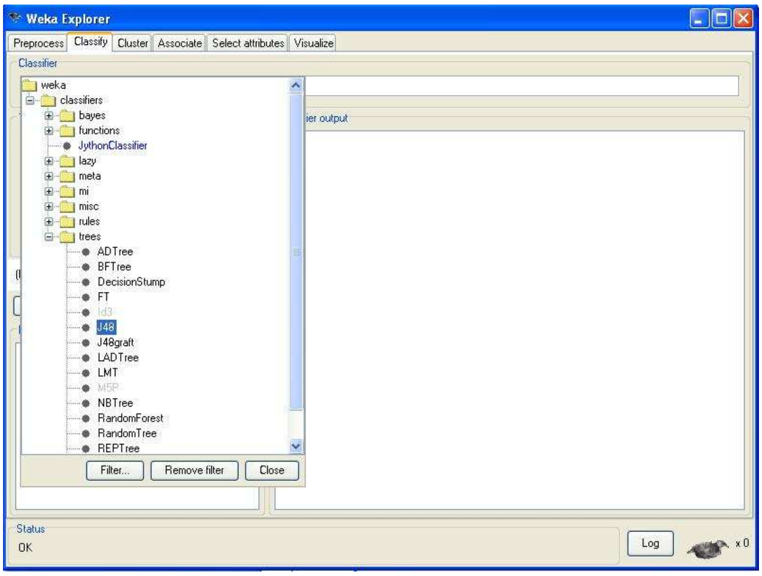
Second data set:

Bankprediction data set

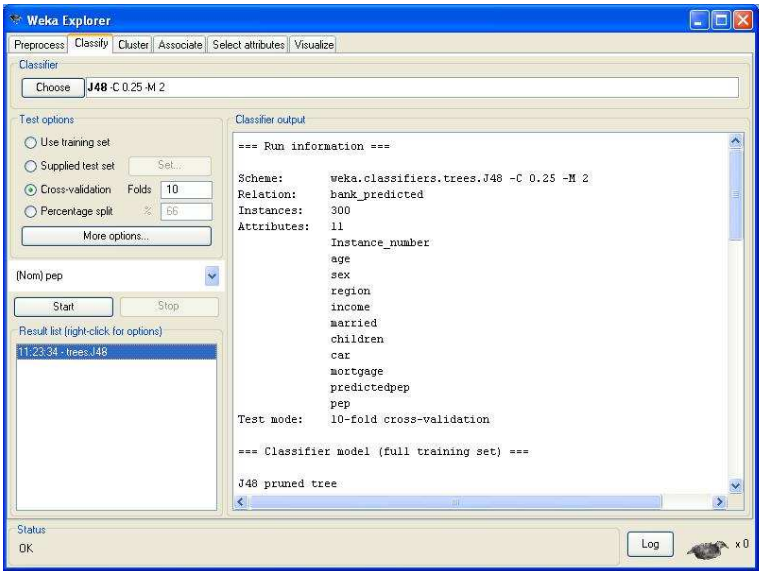


Click on classify

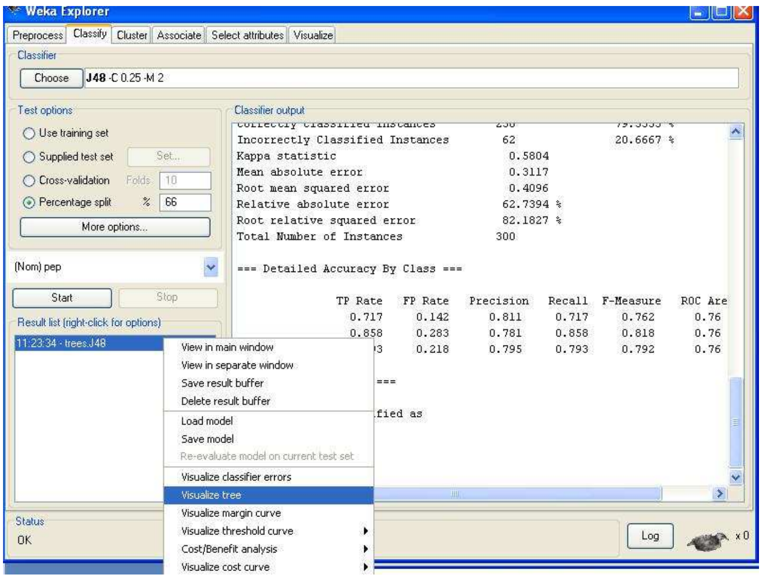
Then click on choose and in trees choose j48

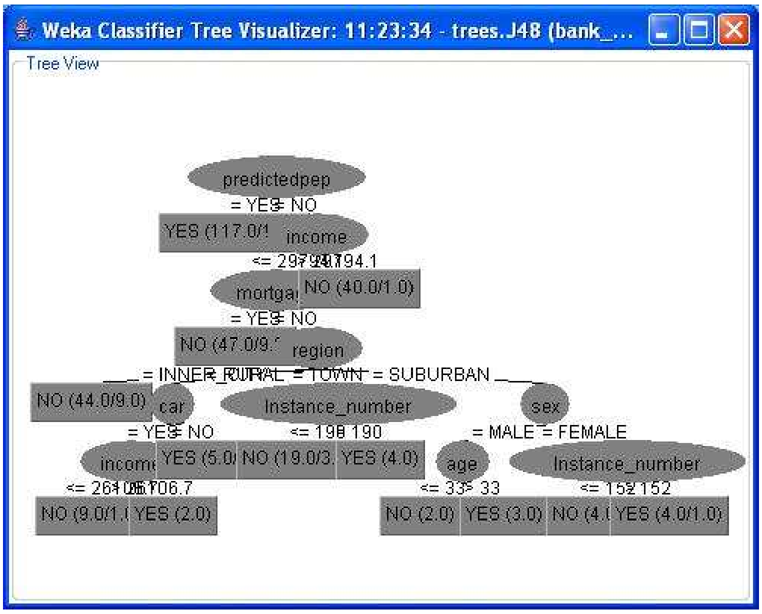


Click on start



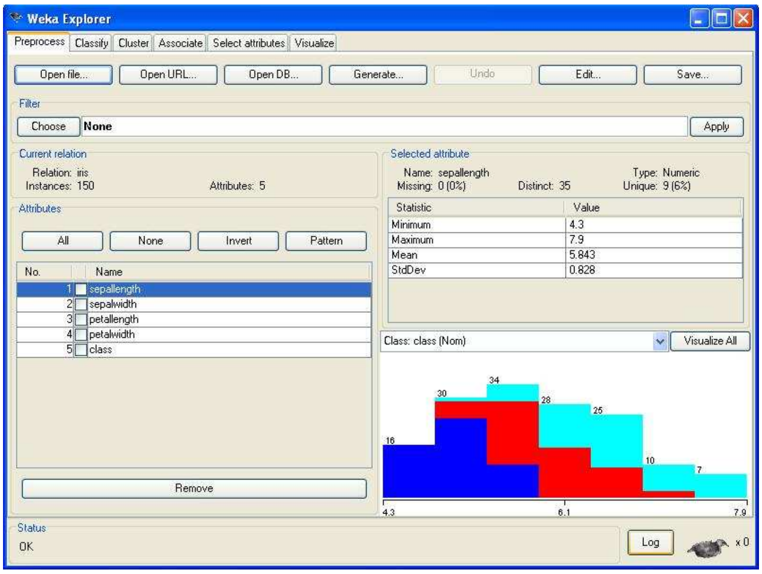
Click on percentage split



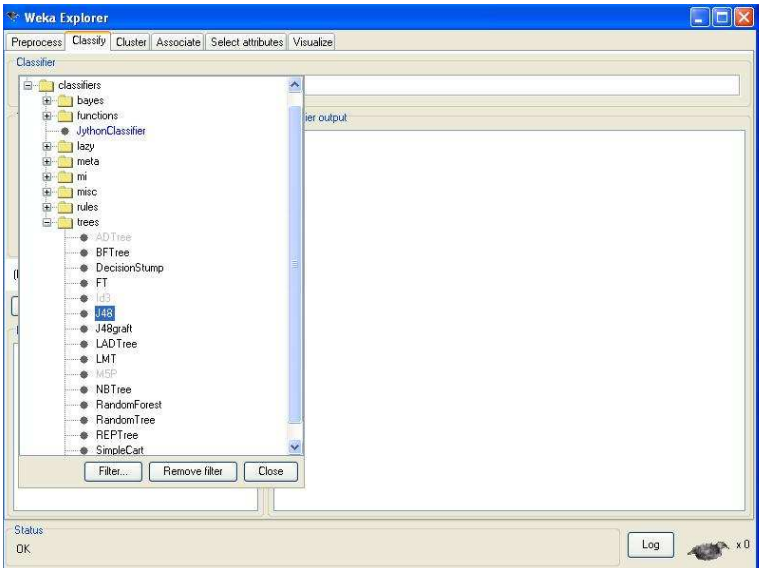

Weka Classifier Tree Visualizer: 11:23:34 - trees.J48 (bank_...
Tree View
predictedpep
income
mortgage
region
car
Instance_number
sex
age

Third data set:

Sleep data set

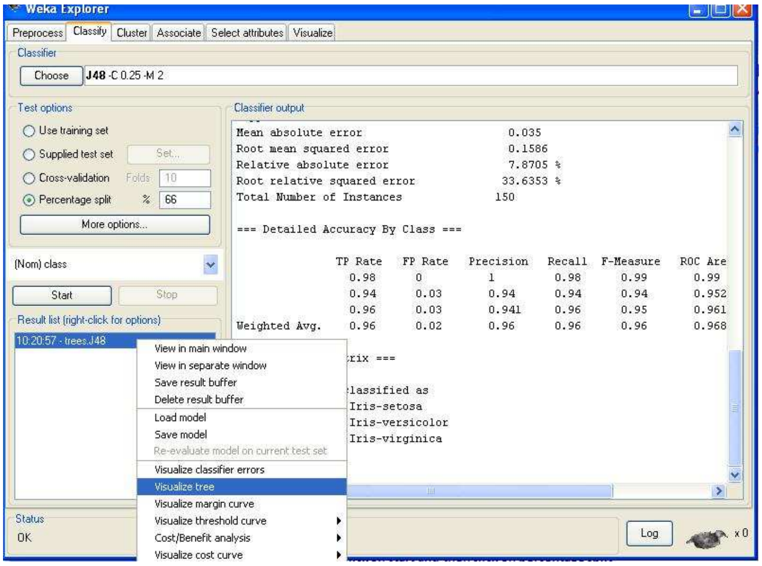


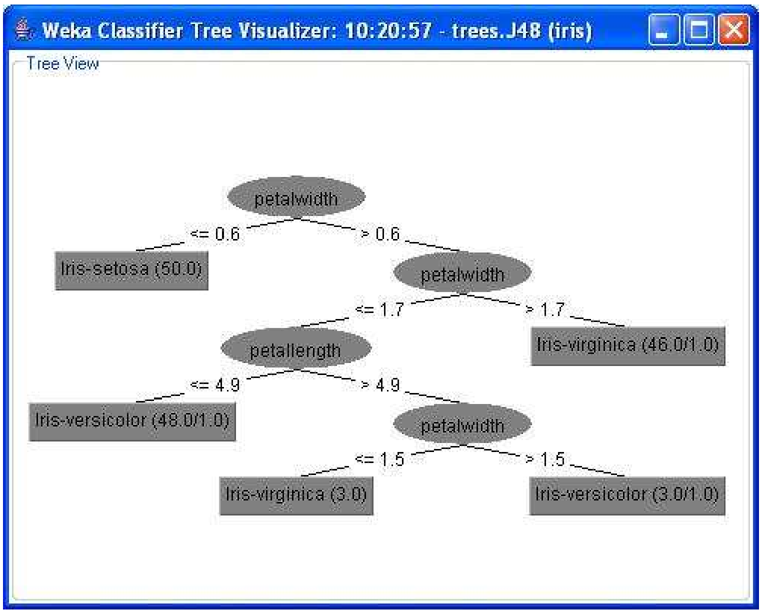
Click on classify and choose from trees choose j48 algorithm



Click on start and then click on percentage split

VALIDATION :


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
10:20:57 - trees.J48
Classifier output
Mean absolute error 0.035
Root mean squared error 0.1586
Relative absolute error 7.8705 %
Root relative squared error 33.6353 %
Total Number of Instances 150
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Are
0.98 0 1 0.98 0.99 0.99
0.94 0.03 0.94 0.94 0.94 0.952
0.96 0.03 0.941 0.96 0.95 0.961
Weighted Avg. 0.96 0.02 0.96 0.96 0.96 0.968
View in main window
View in separate window
Save result buffer
Delete result buffer
Load model
Save model
Re-evaluate model on current test set
Visualize classifier errors
Visualize tree
Visualize margin curve
Visualize threshold curve
Cost/Benefit analysis
Visualize cost curve
Iris-setosa
Iris-versicolor
Iris-virginica
Status
OK
Log
x 0

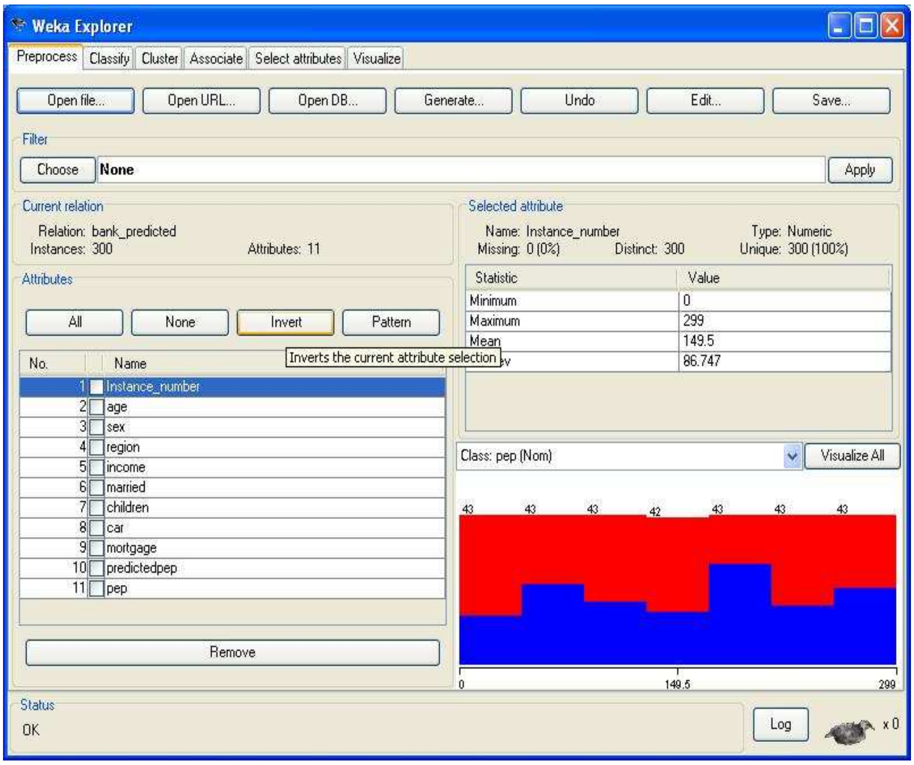

Weka Classifier Tree Visualizer: 10:20:57 - trees.J48 (iris)
Tree View
petalwidth
<= 0.6
> 0.6
Iris-setosa (50.0)
petalwidth
<= 1.7
> 1.7
Iris-virginica (46.0/1.0)
petallength
<= 4.9
> 4.9
Iris-versicolor (48.0/1.0)
petalwidth
<= 1.5
> 1.5
Iris-virginica (3.0)
Iris-versicolor (3.0/1.0)

Program 3 CLASSIFICATION ALGORITHMS USING WEKA

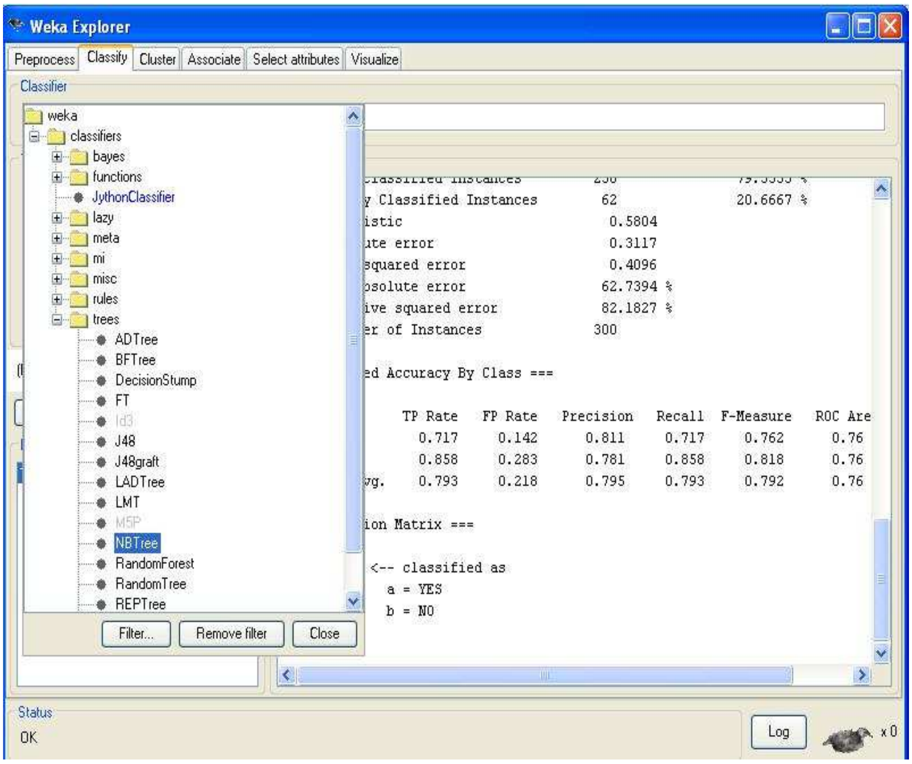
Problem Definition

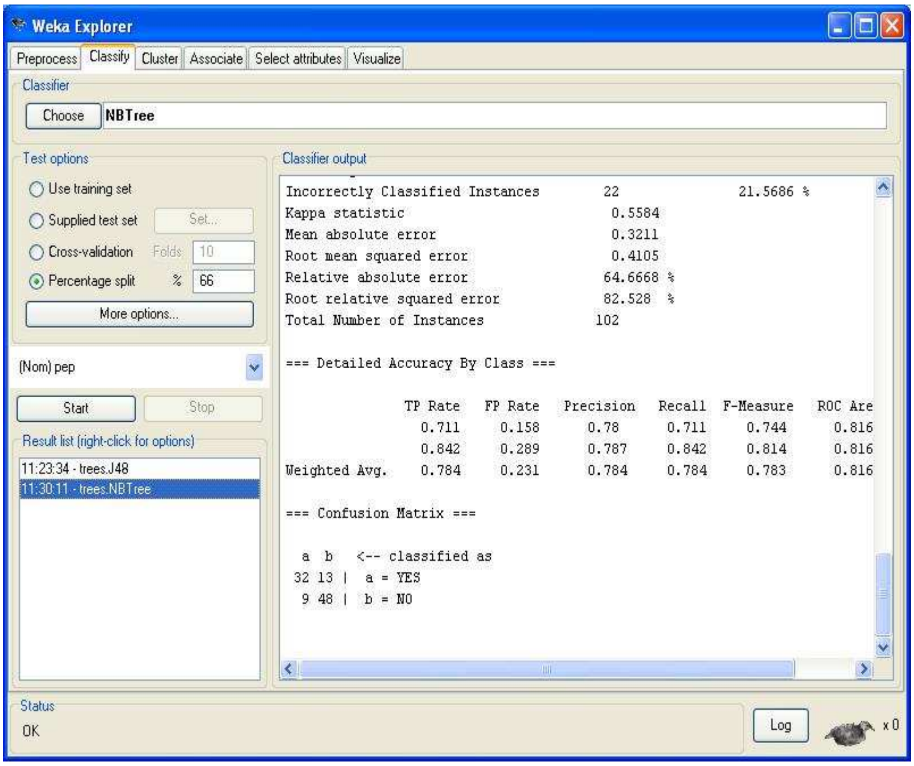
Classification algorithms using WEKA

Bank prediction data set using nb tree:

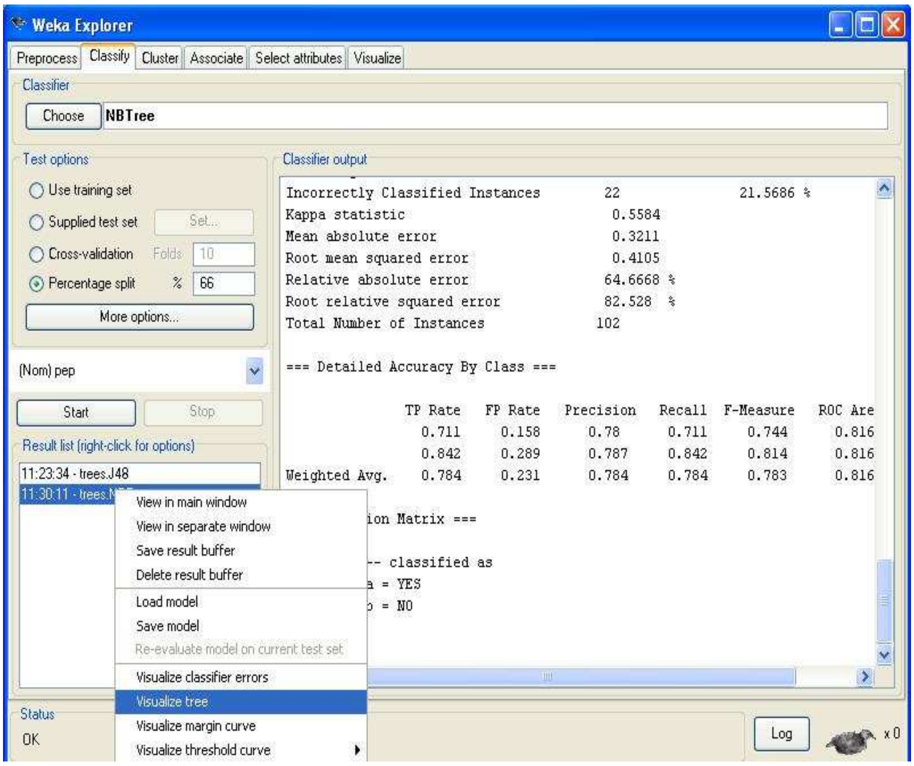


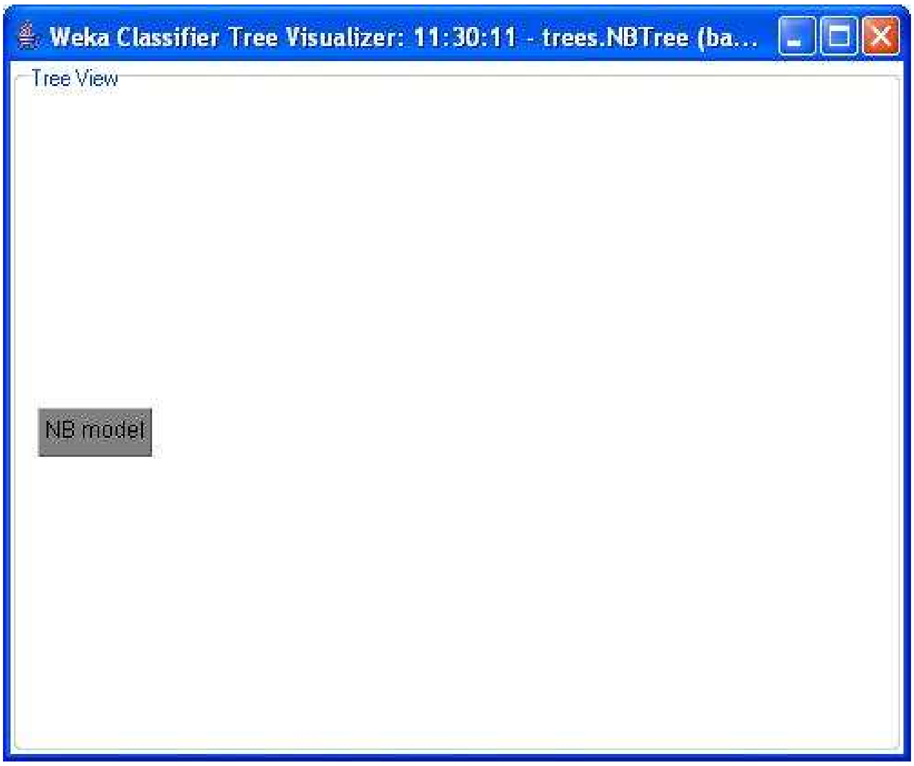
Bank prediction data set using j48graft :

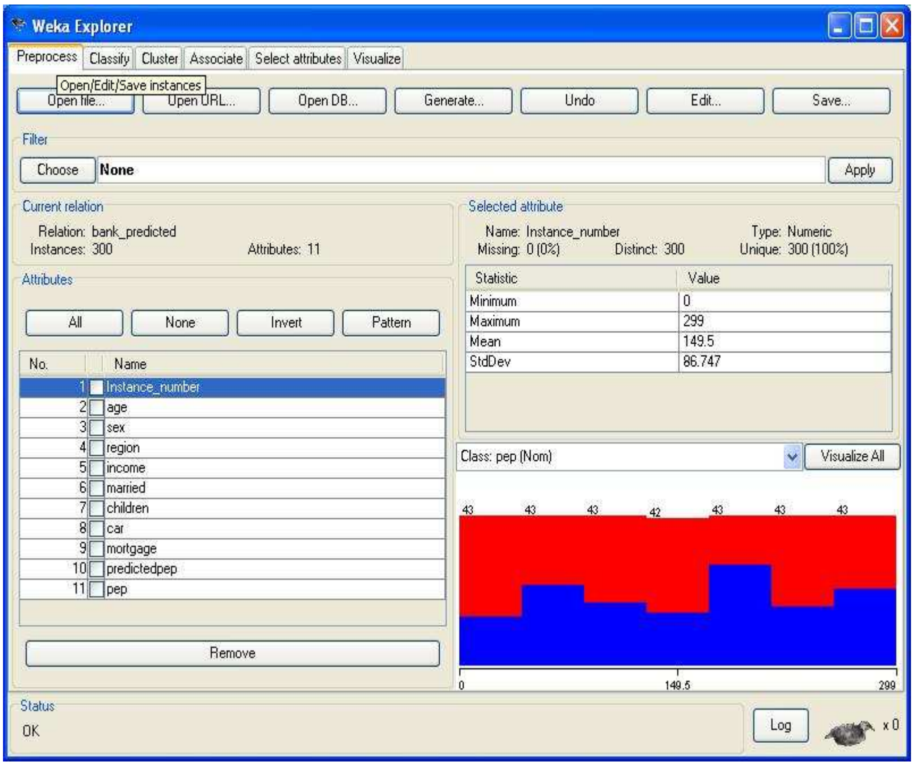


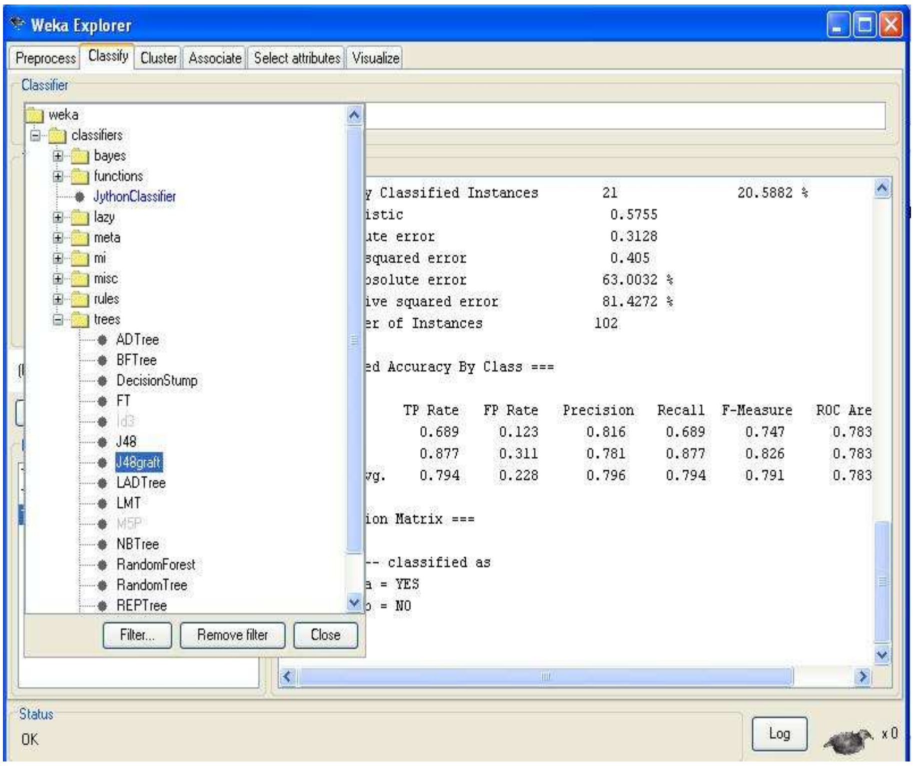


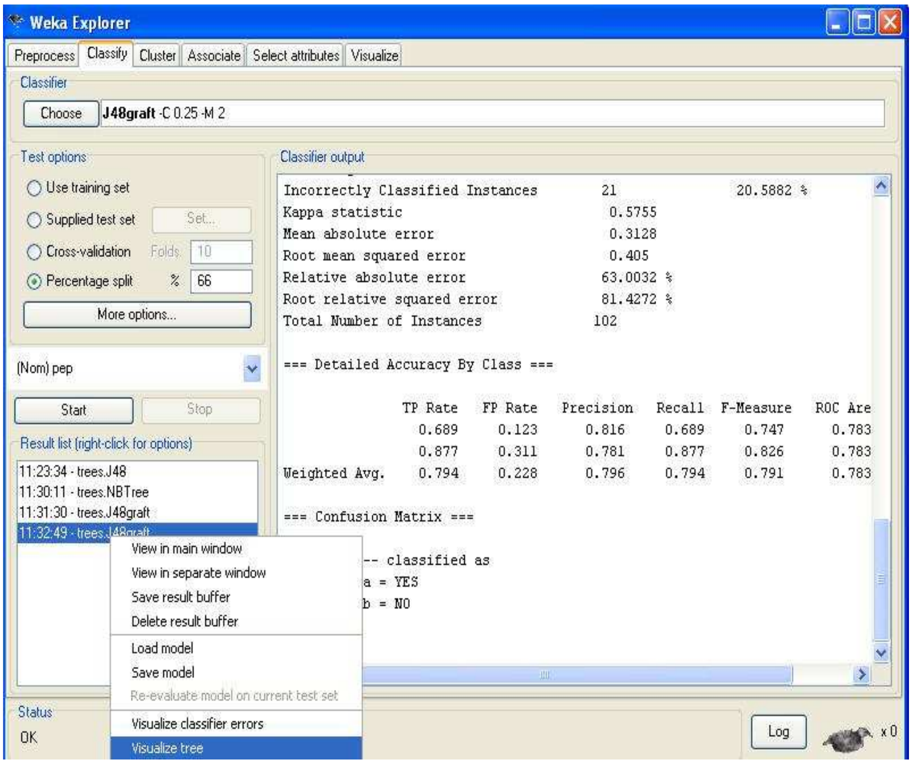
Bank prediction data set using NBtree :
Bank prediction data set using nb tree:

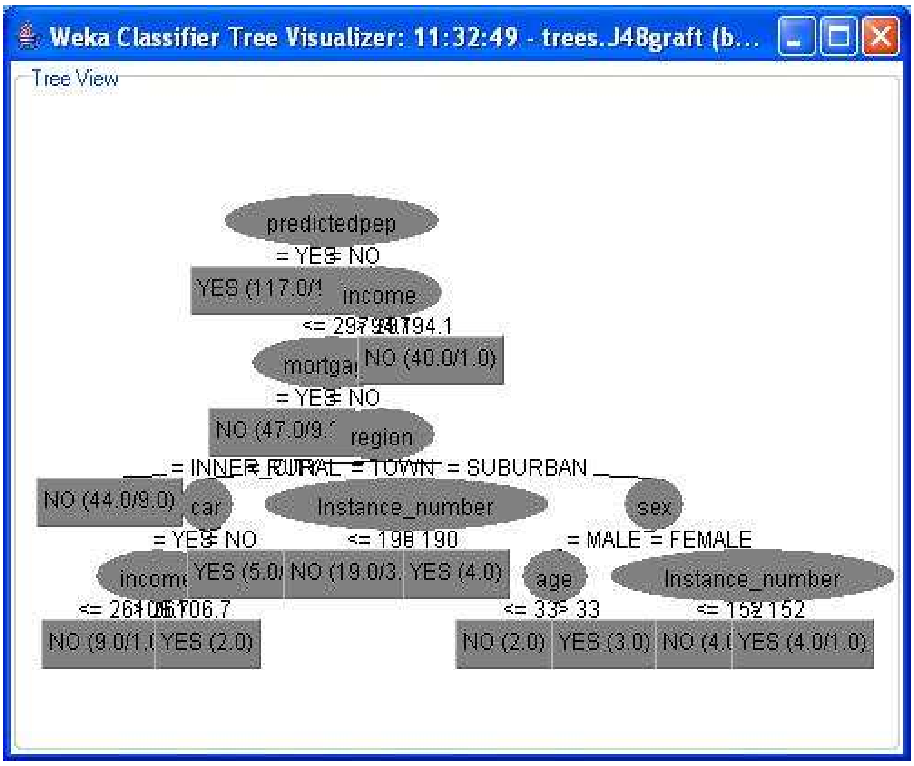

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NBTree
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) pep
Start
Stop
Result list (right-click for options)
11:23:34 - trees.J48
Classifier output
Incorrectly Classified Instances 22 21.5686 %
Kappa statistic 0.5584
Mean absolute error 0.3211
Root mean squared error 0.4105
Relative absolute error 64.6668 %
Root relative squared error 82.528 %
Total Number of Instances 102
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Are
0.711 0.158 0.78 0.711 0.744 0.816
0.842 0.289 0.787 0.842 0.814 0.816
Weighted Avg. 0.784 0.231 0.784 0.784 0.783 0.816
View in main window
View in separate window
Save result buffer
Delete result buffer
Load model
Save model
Re-evaluate model on current test set
Visualize classifier errors
Visualize tree
Visualize margin curve
Visualize threshold curve
Status
OK
Log

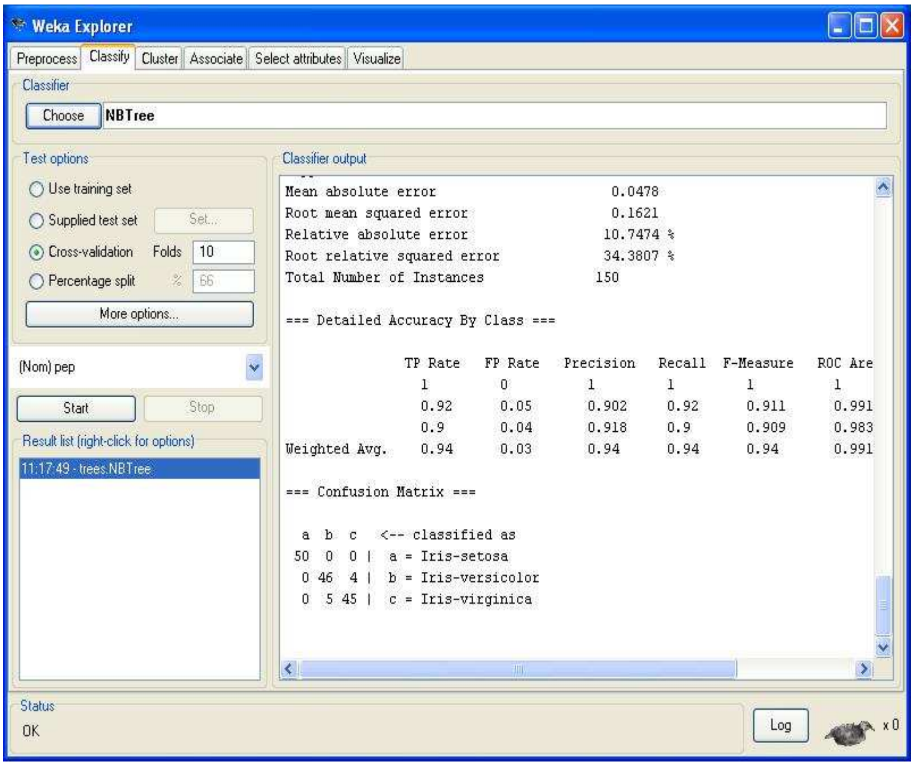

Weka Classifier Tree Visualizer: 11:30:11 - trees.NBTree (ba...
Tree View
NB model

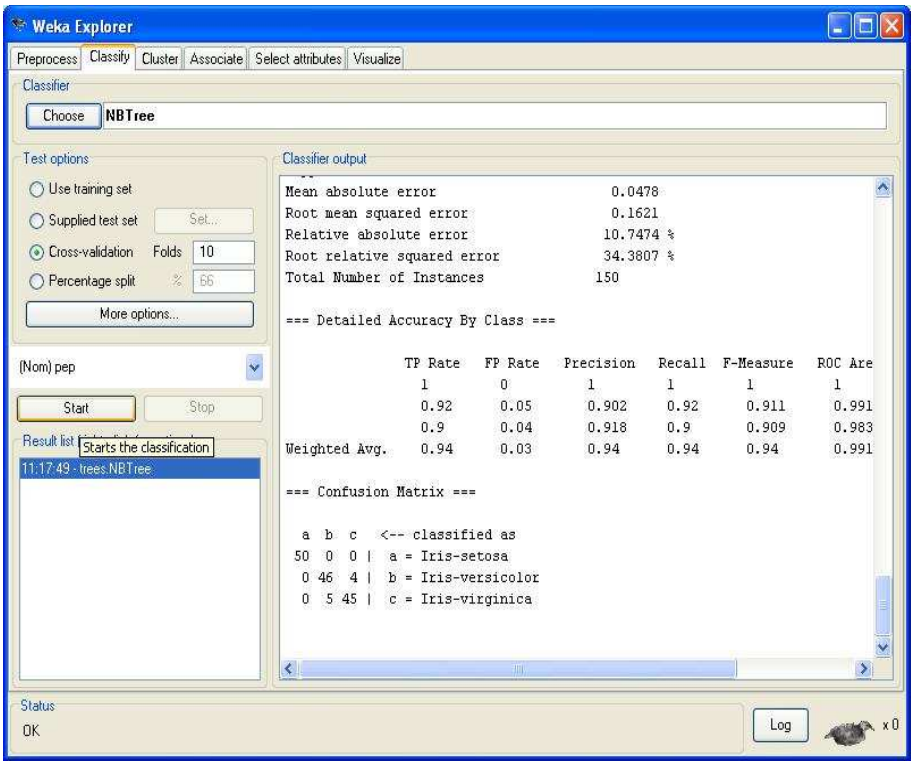

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open/Edit/Save instances
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Current relation
Relation: bank_predicted
Instances: 300
Attributes: 11
Selected attribute
Name: Instance_number
Type: Numeric
Missing: 0 (0%)
Distinct: 300
Unique: 300 (100%)
Attributes
All
None
Invert
Pattern
No.
Name
1 Instance_number
2 age
3 sex
4 region
5 income
6 married
7 children
8 car
9 mortgage
10 predictedpep
11 pep
Statistic
Value
Minimum
0
Maximum
299
Mean
149.5
StdDev
86.747
Class: pep (Nom)
Visualize All
43
43
43
42
43
43
43
0
149.5
299
Remove
Status
OK
Log
x 0

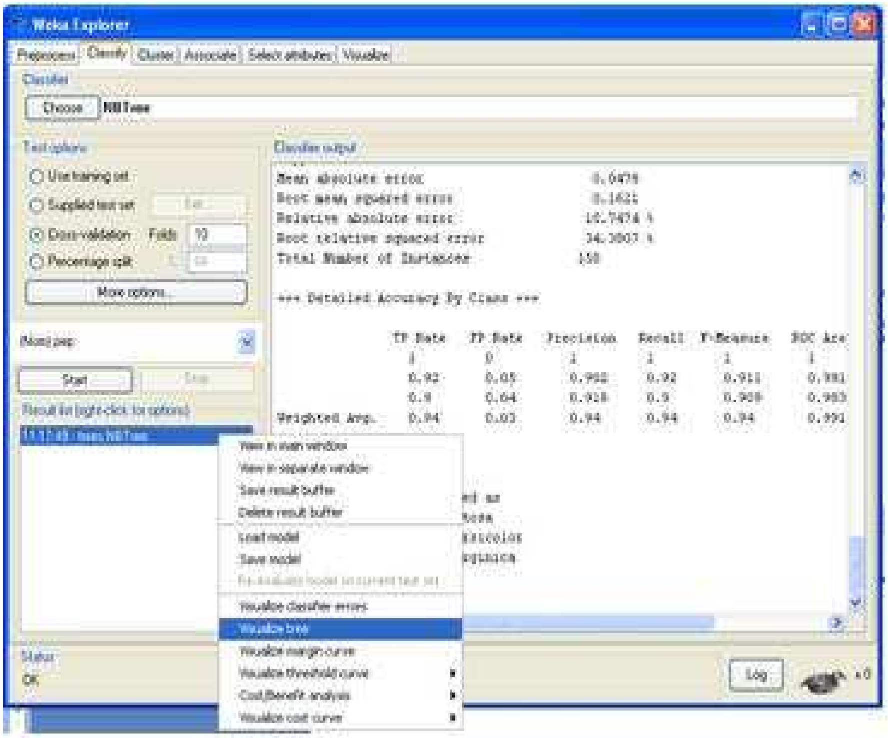

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
weka
classifiers
bayes
functions
JythonClassifier
lazy
meta
mi
misc
rules
trees
ADTree
BFTree
DecisionStump
FT
Id3
J48
J48graft
LADTree
LMT
M5P
NBTree
RandomForest
RandomTree
REPTree
Filter...
Remove filter
Close
y Classified Instances 21 20.5882 %
istic 0.5755
ute error 0.3128
squared error 0.405
bsolute error 63.0032 %
ive squared error 81.4272 %
er of Instances 102
ed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Are
0.689 0.123 0.816 0.689 0.747 0.783
0.877 0.311 0.781 0.877 0.826 0.783
vg. 0.794 0.228 0.796 0.794 0.791 0.783
ion Matrix ===
-- classified as
a = YES
b = NO
Status
OK
Log
x 0

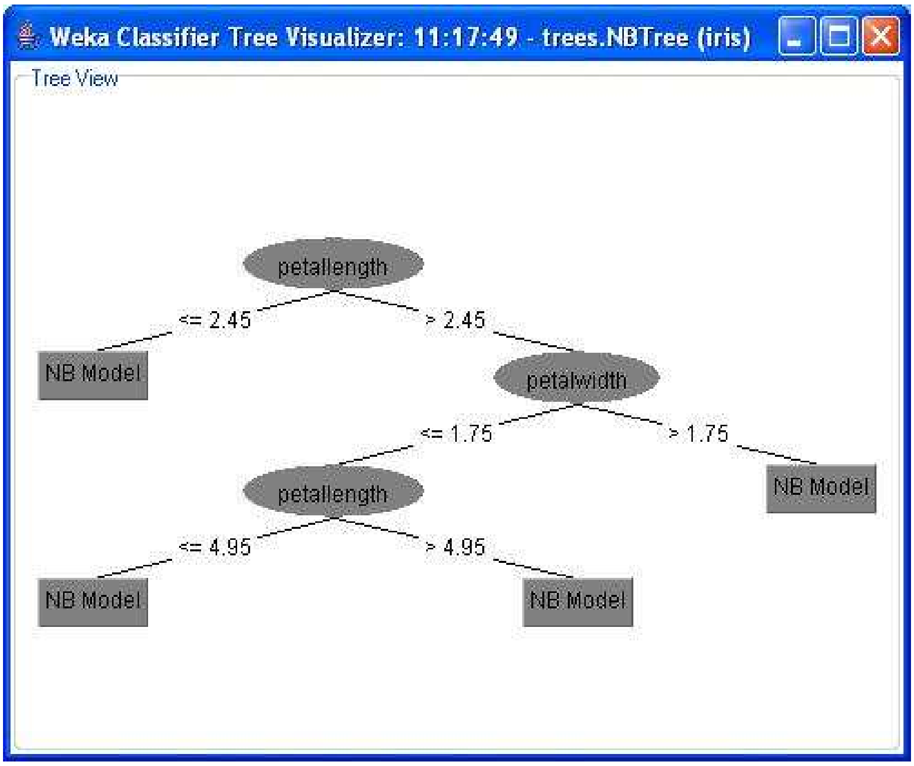

Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose J48graft -C 0.25 -M 2
Test options
Use training set
Supplied test set Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) pep
Start Stop
Result list (right-click for options)
11:23:34 - trees.J48
11:30:11 - trees.NBTree
11:31:30 - trees.J48graft
View in main window
View in separate window
Save result buffer
Delete result buffer
Load model
Save model
Re-evaluate model on current test set
Visualize classifier errors
Visualize tree
Classifier output
Incorrectly Classified Instances 21 20.5882 %
Kappa statistic 0.5755
Mean absolute error 0.3128
Root mean squared error 0.405
Relative absolute error 63.0032 %
Root relative squared error 81.4272 %
Total Number of Instances 102
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Are
0.689 0.123 0.816 0.689 0.747 0.783
0.877 0.311 0.781 0.877 0.826 0.783
Weighted Avg. 0.794 0.228 0.796 0.794 0.791 0.783
=== Confusion Matrix ===
-- classified as
a = YES
b = NO
Status
OK
Log
x 0


Weka Classifier Tree Visualizer: 11:32:49 - trees.J48graft (b...
Tree View
predictedpep
income
NO (40.0/1.0)
region
= SUBURBAN
NO (44.0/9.0)
car
Instance_number
sex
= MALE = FEMALE
age
Instance_number
YES (4.0)
NO (2.0)
YES (3.0)
YES (4.0/1.0)
YES (2.0)






Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NBTree
Test options
Classifier output
Use training set
Supplied test set
Cross-validation
Folds
10
Percentage split
More options...
Start
Stop
Mean absolute error
0.0479
Root mean squared error
0.1621
Relative absolute error
10.7474 %
Root relative squared error
34.3007 %
Total Number of Instances
150
=== Detailed Accuracy By Class ===
TP Rate
FP Rate
Precision
Recall
F-Measure
ROC Area
Weighted Avg.
View in main window
View in separate window
Save result buffer
Delete result buffer
Load model
Save model
Visualize classifier errors
Visualize tree
Visualize margin curve
Visualize threshold curve
Cost/Benefit analysis
Visualize cost curve
Status
OK
Log


Weka Classifier Tree Visualizer: 11:17:49 - trees.NBTree (iris)
Tree View
petallength
<= 2.45
> 2.45
NB Model
petalwidth
<= 1.75
> 1.75
petallength
NB Model
<= 4.95
> 4.95
NB Model
NB Model

Program 4

OBTAIN THE ASSOCIATION RULES FOR GIVEN DATASET USING WEKA

Problem Definition

Obtain The Association Rules For Given Dataset Using WEKA.

Problem Description

Class implementing an Apriori-type algorithm. Iteratively reduces the minimum support until it finds the required number of rules with the given minimum confidence.

The algorithm has an option to mine class association rules.

R. Agrawal, R. Srikant: Fast Algorithms for Mining Association Rules in Large Databases. In: 20th International Conference on Very Large Data Bases, 1994.

Bing Liu, Wynne Hsu, Yiming Ma: Integrating Classification and Association Rule Mining. In: Fourth International Conference on Knowledge Discovery and Data Mining, 1998.

OPTIONS

car – If enabled class association rules are mined instead of (general) association rules.

Class Index – Index of the class attribute. If set to -1, the last attribute is taken as class attribute.

delta – Iteratively decrease support by this factor. Reduces support until min support is reached or required number of rules has been generated.

Lower Bound Min Support – Lower bound for minimum support.

Metric Type – Set the type of metric by which to rank rules. Confidence is the proportion of the examples covered by the premise that are also covered by the consequence(Class association rules can only be mined using confidence). Lift is confidence divided by the proportion of all examples that are covered by the consequence. This is a measure of the importance of the association that is independent of support. Leverage is the proportion of additional examples covered by both the premise and consequence above those expected if the premise and consequence were independent of each other. The total number of examples that this represents is presented in brackets following the leverage. Conviction is another measure of departure from independence. Conviction is given by P(premise)P(!consequence) / P(premise, !consequence).

min Metric – Minimum metric score. Consider only rules with scores higher than this value.

Num Rules – Number of rules to find.

Output Item Sets – If enabled the item sets are output as well.

Remove All Missing Cols – Remove columns with all missing values.

Significance Level – Significance level. Significance test (confidence metric only).

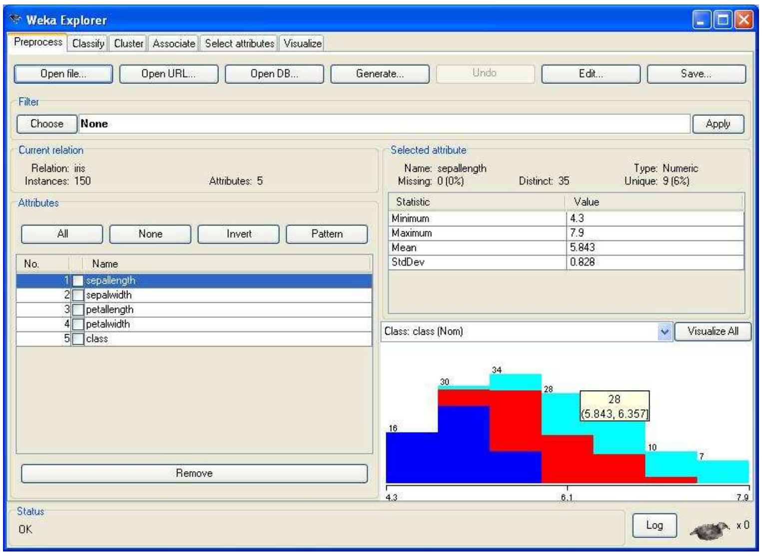
Upper Bound Min Support – Upper bound for minimum support. Start iteratively decreasing minimum support from this value.

verbose – If enabled the algorithm will be run in verbose mode.

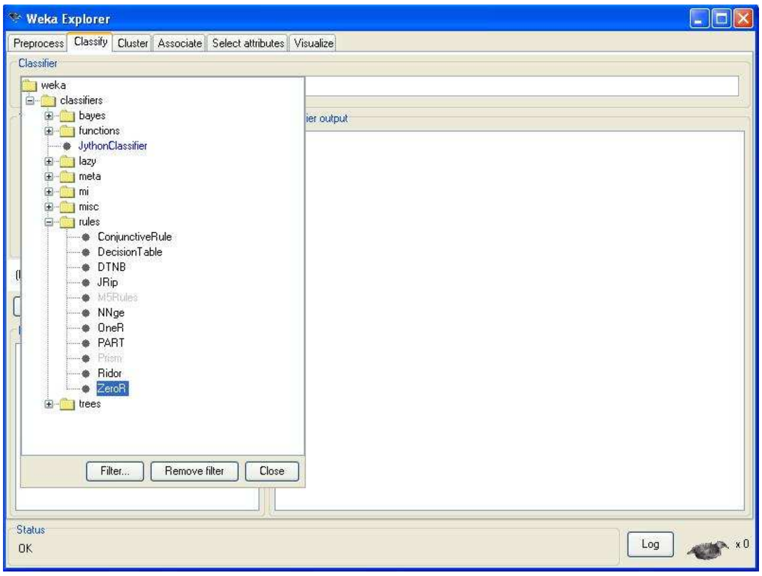
STEPS

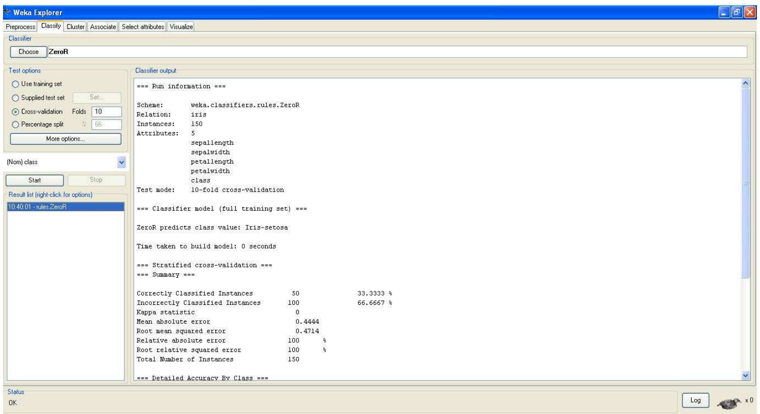
Choose a data set

Iris data set

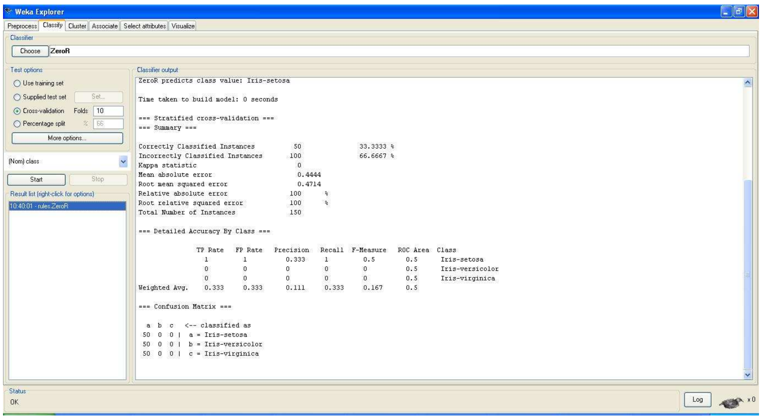


Click on classify and choose ZeroR





VALIDATION :


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
ZeroR
Test options
Use training set
Supplied test set
Cross-validation
Folds
10
Percentage split
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
10:40:01 - rules.ZeroR
Classifier output
ZeroR predicts class value: Iris-setosa
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 50 33.3333 %
Incorrectly Classified Instances 100 66.6667 %
Kappa statistic 0
Mean absolute error 0.4444
Root mean squared error 0.4714
Relative absolute error 100 %
Root relative squared error 100 %
Total Number of Instances 150
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
1 1 0.333 1 0.5 0.5 Iris-setosa
0 0 0 0 0 0.5 Iris-versicolor
0 0 0 0 0 0.5 Iris-virginica
Weighted Avg. 0.333 0.333 0.111 0.333 0.167 0.5
=== Confusion Matrix ===
a b c <-- classified as
50 0 0 | a = Iris-setosa
50 0 0 | b = Iris-versicolor
50 0 0 | c = Iris-virginica
Status
OK
Log

Program 5

PERFORM DATA TRANSFORMATIONS USING ETL TOOL

Problem Definition

Perform data transformations using an ETL tool.

Problem Description

Extraction, Transformation, and Loading (ETL) Processes

- The plumbing work of data warehousing
- Data are moved from source to target data bases
- A very costly, time consuming part of data warehousing

Sample ETL Tools

- Tera data Warehouse Builder from Tera data
- Data Stage from Ascential Software
- SAS System from SAS Institute
- Power Mart/Power Center from Informatica
- Sagent Solution from Sagent Software
- Hummingbird Genio Suite from Humming bird Communications

Introduction to MySQL:

- MySQL is a very popular, open source database.
- Officially pronounced my Ess Que Ell (not my sequel).
- Handles very large databases; very fast performance.
- Why are we using MySQL?
  - Free (much cheaper than Oracle!)
  - Each student can install MySQL locally.
  - Easy to use Shell for creating tables, querying tables, etc.
  - Easy to use with Java JDBC

A) For products and orders

```
mysql> use nida;
Database changed

mysql> create table orders(orderid varchar(5),
         -> productid varchar(5),quantity int(5),
         -> unitsaleprice int(5),discountprice int(5),
        -> numoffreeservice int(3));
Query OK, 0 rows affected (0.09 sec)

mysql> desc orders;
```

Field	Type	Null	Key	Default	Extra
orderid	varchar(5)	YES		NULL	
productid	varchar(5)	YES		NULL	
quantity	int(5)	YES		NULL	
unitsaleprice	int(5)	YES		NULL	
discountprice	int(5)	YES		NULL	
numoffreeservice	int(3)	YES		NULL	

```
6 rows in set (0.01 sec)

mysql> insert into orders values('0101','p121',10,5000,10,2);
Query OK, 1 row affected (0.01 sec)

mysql> insert into orders values('0101','p121',10,5000,10,2);
Query OK, 1 row affected (0.02 sec)

mysql> insert into orders values('0101','p01',1,234,5,3);
Query OK, 1 row affected (0.01 sec)

mysql> insert into orders values('0101','p180',2,4000,12,3);
Query OK, 1 row affected (0.03 sec)

mysql> insert into orders values('0102','p02',5,2500,3,2);
Query OK, 1 row affected (0.02 sec)

mysql> insert into orders values('0102','p122',2,2800,3,2);
Query OK, 1 row affected (0.03 sec)
```

```
mysql> select * from orders;
```

orderid	productid	quantity	unitsaleprice	discountprice	numoffreeservice
0101	p121	10	5000	10	2
0101	p121	10	5000	10	2
0101	p01	1	34	5	3
0101	p180	2	4000	12	3
0102	p02	5	2500	3	2
0102	p122	2	2800	3	2

```
6 rows in set (0.00 sec)

mysql> create table products(productid varchar(5) primary key,
         -> companyid varchar(5),
        -> productname varchar(10),
        -> producttype varchar(10),
        -> productprice int(5),
        -> productdom date,
       -> productinstock int(5));

Query OK, 0 rows affected (0.06 sec)
```

Field	Type	Null	Key	Default	Extra
productid	varchar(5)	NO	PRI	NULL	
companyid	varchar(5)	YES		NULL	
productname	varchar(10)	YES		NULL	
producttype	varchar(10)	YES		NULL	
productprice	int(5)	YES		NULL	
productdom	date	YES		NULL	
productinstock	int(5)	YES		NULL	

```
7 rows in set (0.02 sec)

mysql> insert into products values('P01','30','CABLE',121,234,
       '1991-01-09',25);
Query OK, 1 row affected (0.03 sec)

mysql> insert into products values('P02','28','OPCABLE',122,500,
       '1998-08-04',35);
Query OK, 1 row affected (0.01 sec)

mysql> insert into products  values('P121','12','MONITOR',147,5000,
       '2001-09-25',19);
Query OK, 1 row affected (0.02 sec)

mysql> insert into products  values('P122','11','BATTERY',124,1400,
```

```
       '2003-08-15',15);
Query OK, 1 row affected (0.03 sec)

mysql> insert into products values('P180','2','FAX',168,2100,
       '2003-03-12',12);
Query OK, 1 row affected (0.03 sec)

mysql> select * from products;
```

product id	company id	product name	product type	product price	product dom	product instock
P01	30	CABLE	121	234	1991-01-09	25
P02	28	OPCABLE	122	500	1998-08-04	35
P121	12	MONITOR	147	5000	2001-09-25	19
P122	11	BATTERY	124	1400	2003-08-15	15
P180	2	FAX	168	2100	2003-03-12	12

5 rows in set (0.00 sec)

B) For birth certificate

```
mysql> use nida;
Database changed

mysql> create table birthcertificate(name varchar(30),
       Fathername varchar(30), Birth date,Location varchar(30),
 SSN varchar(3), primary key(SSN));
Query OK, 0 rows affected (0.09 sec)

mysql> insert into birthcertificate values('KLM','STR','1988-03-27',
       'Delhi','665');
Query OK, 1 row affected (0.02 sec)

mysql>insertintobirthcertificatevalues('ABC','XYZ','1989-05-03',
  'HYDERABAD','667');
Query OK, 1 row affected (0.03 sec)

mysql> select * from birthcertificate;
```

Name	Fathername	Birth	Location	SSN
GHI	MNO	1990-11-17	BANGLORE	645
KLM	STR	1988-03-27	Delhi	665
ABC	XYZ	1989-05-03	HYDERABAD	667

```
3 rows in set (0.00 sec)
```

```
mysql> insert into birthcertificate values('DEF','PQR','1987-06-07',
     'HYDERABAD','785');
Query OK, 1 row affected (0.02 sec)

mysql> create table college(Name varchar(30),Fathername varchar(20),
 Birth date, SSN varchar(3),Rollno int(3));
Query OK, 0 rows affected (0.06 sec)

mysql> insert into college values('ABC','XYZ','1989-05-03','667',10);
Query OK, 1 row affected (0.03 sec)

mysql> insert into college values('KLM','STR','1988-03-27','665',11);
Query OK, 1 row affected (0.01 sec)

mysql> insert into college values('GHI','MNO','1990-11-17','645',15);
Query OK, 1 row affected (0.02 sec)

mysql> insert into college values('DEF','PQR','1987-06-07','785',14);
Query OK, 1 row affected (0.03 sec)

mysql> insert into college values('GTH','THR','1989-06-01','765',12);
Query OK, 1 row affected (0.02 sec)

mysql> insert into college values('RTU','YHN','1994-07-24','378',19);
Query OK, 1 row affected (0.03 sec)

mysql> create table addrproof(Name1 varchar(30),Name2 varchar(30),
           Address varchar(30),SSN varchar(30) references
 birthcertificate(SSN));
Query OK, 0 rows affected (0.08 sec)

mysql> insert into addrproof values
('ABC','XYZ','HYDERABAD','667');
Query OK, 1 row affected (0.02 sec)

mysql> insert into addrproof values
('RTU','YHN','HYDERABAD','378');
Query OK, 1 row affected (0.02 sec)

mysql> insert into addrproof values
('GHI','MNO','DELHI','645');
Query OK, 1 row affected (0.02 sec)

mysql> insert into addrproof values
     ('KLM','STR','BANGLORE','665');
Query OK, 1 row affected (0.02 sec)
```

```
mysql> insert into addrproof values
      ('GTH','THR','BANGLORE','765');
Query OK, 1 row affected (0.01 sec)

mysql> insert into addrproof values
      ('DEF','PQR','CHENNAI','785');
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM ADDRPROOF;
```

Name1	Name2	Address	SSN
ABC	XYZ	HYDERABAD	667
RTU	YHN	HYDERABAD	378
GHI	MNO	DELHI	645
KLM	STR	BANGLORE	665
GTH	THR	BANGLORE	765
DEF	PQR	CHENNAI	785

```
6 rows in set (0.02 sec)

mysql> create table citizeninfosystem as
          (select birthcertificate.Name,birthcertificate.Fathername,
   birthcertificate.Birth, birthcertificate.SSN,
 (birthcertificate.Location) as birthloc,addrproof.Address,
           college.Rollno from birthcertificate inner join addrproof on
           birthcertificate.Name=addrproof.Name1 inner join college on
           birthcertificate.Name=college.Name);
Query OK, 4 rows affected (0.08 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from citizeninfosystem;
```

Name	Fathername	Birth	SSN	BirthLocation	Address	RollNo
ABC	XYZ	1989-05-03	667	HYDERABAD	HYDERABAD	10
KLM	STR	1988-03-27	665	DELHI	BANGLORE	11
GHI	MNO	1990-11-17	645	BANGLORE	DELHI	15
DEF	PQR	1987-06-07	785	HYDERABAD	CHENNAI	14

4 rows in set (0.00 sec)

mysql> select * from birthcertificate;

Name	Fathername	Birth	Location	SSN
GHI	MNO	1990-11-17	BANGLORE	645
KLM	STR	1988-03-27	Delhi	665
ABC	XYZ	1989-05-03	HYDERABAD	667
DEF	PQR	1987-06-07	HYDERABAD	785

4 rows in set (0.00 sec)

mysql> select * from college;

Name	Fathername	Birth	SSN	RollNo
GHI	MNO	1990-11-17	645	10
KLM	STR	1988-03-27	665	11
ABC	XYZ	1989-05-03	667	15
DEF	PQR	1987-06-07	785	14
GTH	THR	1989-06-01	667	12
RTU	YHN	1994-07-24	785	19

6 rows in set (0.00 sec)

mysq> select * from birthcertificate;

Name	Fathername	Birth	Location	SSN
GHI	MNO	1990-11-17	BANGLORE	645
KLM	STR	1988-03-27	Delhi	665
ABC	XYZ	1989-05-03	HYDERABAD	667
DEF	PQR	1987-06-07	HYDERABAD	785

4 rows in set (0.00 sec)

mysql> select * from addrproof;

Name1	Name2	Address	SSN
ABC	XYZ	HYDERABAD	667
RTU	YHN	HYDERABAD	378
GHI	MNO	DELHI	645
KLM	STR	BANGLORE	665
GTH	THR	BANGLORE	765
DEF	PQR	CHENNAI	785

6 rows in set (0.00 sec)

c) For bank table

mysql> create table bank(Custname varchar(20), Bankname varchar(20), Accno int(10)); Query OK, 0 rows affected (0.13 sec)

mysql> desc bank;

Field	Type	Null	Key	Default	Extra
Custname	varchar(20)	YES		NULL	
Bankname	varchar(20)	YES		NULL	
Accno	int(10)	NO	PRI	0	

```
3 rows in set (0.00 sec)

mysql> insert into bank values('JOHN','SBH',123);
Query OK, 1 row affected (0.02 sec)

mysql> insert into bank values('MARRY','SBI',346);
Query OK, 1 row affected (0.02 sec)

mysql> insert into bank values('DANY','SBH',768);
Query OK, 1 row affected (0.02 sec)

mysql> insert into bank values('NEHA','SBH',512);
Query OK, 1 row affected (0.02 sec)

mysql> select * from bank;
```

Custname	Bankname	Accno
JOHN	SBH	123
MARRY	SBH	346
NEHA	SBH	512
DANY	SBH	768

```
4 rows in set (0.00 sec)

mysql> create table customer(Custname varchar(20),Address varchar(20),
       Accno int(10)
references bank(Accno));
Query OK, 0 rows affected (0.06 sec)

mysql> desc Customer;
```

Field	Type	Null	Key	Default	Extra
Custname	varchar(20)	YES		NULL	
Address	varchar(20)	YES		NULL	
Accno	int(10)	YES		NULL	

```
3 rows in set (0.02 sec)

mysql> alter table customer add foreign key(Accno) references
       bank(Accno);
```

```
Query OK, 0 rows affected (0.16 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc customer;
```

Field	Type	Null	Key	Default	Extra
Custname	varchar(20)	YES		NULL	
Address	varchar(20)	YES		NULL	
Accno	int(10)	YES	MUL	NULL	

```
3 rows in set (0.00 sec)

mysql> alter  table customer add Balance int(30);
Query OK, 0 rows affected (0.16 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc customer;
```

Field	Type	Null	Key	Default	Extra
Custname	varchar(20)	YES		NULL	
Address	varchar(20)	YES		NULL	
Accno	int(10)	YES	MUL	NULL	
Acctype	varchar(20)	YES		NULL	
Balance	int(30)	YES		NULL	

```
5 rows in set (0.00 sec)

mysql> insert into customer values('JOHN','HYDERABAD',123,
       'CURRENT',8000);
Query OK, 1 row affected (0.03 sec)

mysql> insert into customer values('MARRY','HYDERABAD',346,
       'SAVINGS',10000);
Query OK, 1 row affected (0.02 sec)

mysql> insert into customer values('NEHA','NARAYANAGUDA',512,
       'SAVINGS',100000);
Query OK, 1 row affected (0.02 sec)

mysql> insert into customer values('DANY','MEHDIPATNAM',768,
       'CURRENT',15000);
Query OK, 1 row affected (0.03 sec)

mysql> SELECT * FROM CUSTOMER;
```

Custname	Bankname	Accno	Acctype	Balance
JOHN	SBH	123	CURRENT	8000
MARRY	SBH	346	SAVINGS	10000
NEHA	SBH	512	SAVINGS	100000
DANY	SBH	768	CURRENT	15000

```
4 rows in set (0.00 sec)

mysql> select distinct bank.Custname,bank.Accno,Balance from bank
       inner join customer where bank.Accno=customer.Accno;
```

Custname	Accno	Balance
JOHN	123	8000
MARRY	346	10000
NEHA	512	100000
DANY	768	15000

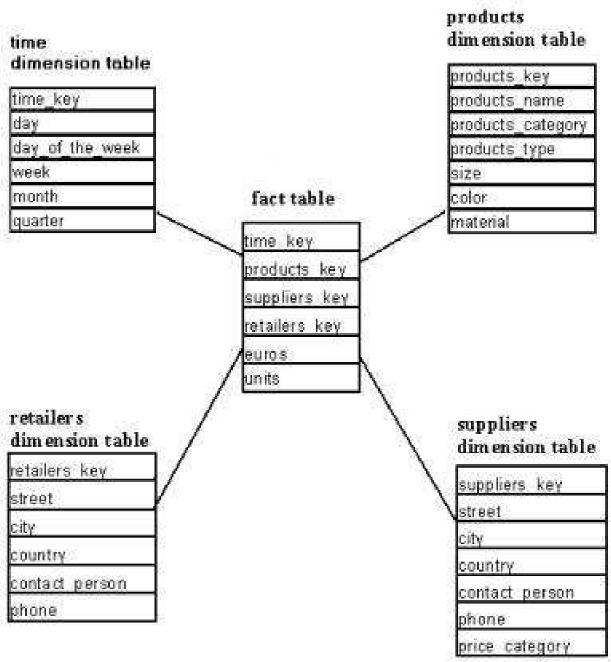
4 rows in set (0.02 sec)

PROGRAM-6:

Write a program to implement star schema.

DESCRIPTION:

The star schema (sometimes referenced as star join schema) is the simplest style of data warehouse schema. The star schema consists of a few fact tables (possibly only one, justifying the name) referencing any number of dimension tables. The star schema is considered an important special case of the snowflake schema.



SOURCE CODE:

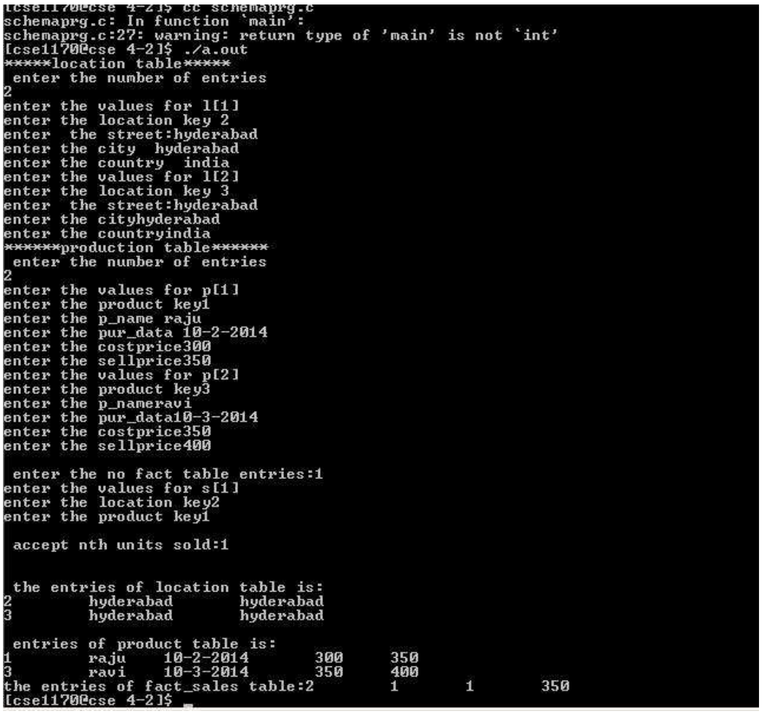
```
#include<stdio.h>
//#include<conio.h>
struct location
{
int l_key;
char street[30];
char city[30];
char p_st[100];
char country[30];
}l[10];
struct sales_fact
{
int l_key;
int p_key;
int unitsold;
int amount;
}s[20];
struct product
{
int p_key;
char p_name[30];
char pur_date[30];
int cost_price;
int sell_price;
}p[10];
void main()
{
int i,j,n2,n3,k,n1;
//clrscr();
printf("*****location table*****");
printf("\n enter the number of entries\n");
scanf("%d",&n1);
for(i=0;i<n1;)
{
printf("enter the values for l[%d]\n",i+1);
printf("enter the location key");
scanf("%d",&l[i].l_key);
if(i>0)
{
for(j=0;j<i;j++)
{
if(l[i].l_key==l[j].l_key)
{
printf("\n dups are not allowed");
```

```
break;
}
}
}
if(i==j||i==0)
{
printf("enter  the street:");
scanf("%s",l[i].street);
printf("enter the city");
scanf("%s",l[i].city);
fflush(stdin);
printf("enter the country");
scanf("%s",l[i].country);
i++;
}
}
printf("******production table******");
printf("\n enter the number of entries\n");
scanf("%d",&n2);
for(i=0;i<n2;)
{
printf("enter the values for p[%d]\n",i+1);
printf("enter the product key");
scanf("%d",&p[i].p_key);
if(i>0)
{
for(j=0;j<i;j++)
{
if(p[i].p_key==p[j].p_key)
{
printf("\n dups are not allowed");
break;
}
}
}
if(i==j||i==0)
{
printf("enter the p_name");
fflush(stdin);
scanf("%s",p[i].p_name);
printf("enter the pur_data");
scanf("%s",p[i].pur_date);
fflush(stdin);
printf("enter the costprice");
scanf("%d",&p[i].cost_price);
printf("enter the sellprice");
```

```
scanf("%d",&p[i].sell_price);
i++;
}
}
printf("\n enter the no fact table entries:");
scanf("%d",&n3);
for(i=0;i<n3;)
{
printf("enter the values for s[%d]\n",i+1);
printf("enter the location key");
scanf("%d",&s[i].l_key);
printf("enter the product key");
scanf("%d",&s[i].p_key);
for(j=0;j<n1;j++)
{
if(s[i].l_key==l[j].l_key)
break;
}
for(k=0;k<n2;k++)
{
if(s[i].p_key==p[k].p_key)
break;
}
if(!(j<n1&&k<n2))
printf("\n he daa must be there in the d tables:");
else
{
printf("\n accept nth units sold:");
scanf("%d",&s[i].unitsold);
s[i].amount=s[i].unitsold*p[k].sell_price;
i++;
}
}
printf("\n \n the entries of location table is: \n");
for(i=0;i<n2;i++)
{
printf("%d\t %s\t %s\t %s\n",l[i].l_key, l[i].street,
     l[i].city, l[i].p_st, l[i].country);
}
printf("\n entries of product table is: \n");
for(i=0;i<n2;i++)
{
printf("%d\t %s\t %s\t %d\t %d\n",p[i].p_key,p[i].p_name,
       p[i].pur_date, p[i].cost_price, p[i].sell_price);
}
printf("the entries of fact_sales table:");
```

```
for(i=0;i<n3;i++)
{
printf("%d\t %d\t %d\t %d\n", s[i].l_key, s[i].p_key,
       s[i].unitsold, s[i].amount);
}
}
```

Output:



```
[cse1170@cse 4-2]$ cc schemaprg.c
schemaprg.c: In function `main':
schemaprg.c:27: warning: return type of 'main' is not `int'
[cse1170@cse 4-2]$ ./a.out
*****location table*****
 enter the number of entries
2
enter the values for l[1]
enter the location key 2
enter  the street:hyderabad
enter the city  hyderabad
enter the country  india
enter the values for l[2]
enter the location key 3
enter  the street:hyderabad
enter the cityhyderabad
enter the countryindia
******production table******
 enter the number of entries
2
enter the values for p[1]
enter the product key1
enter the p_name raju
enter the pur_data 10-2-2014
enter the costprice300
enter the sellprice350
enter the values for p[2]
enter the product key3
enter the p_nameravi
enter the pur_data10-3-2014
enter the costprice350
enter the sellprice400

 enter the no fact table entries:1
enter the values for s[1]
enter the location key2
enter the product key1

 accept nth units sold:1


 the entries of location table is:
2        hyderabad       hyderabad
3        hyderabad       hyderabad

 entries of product table is:
1        raju    10-2-2014       300     350
3        ravi    10-3-2014       350     400
the entries of fact_sales table:2        1       1       350
[cse1170@cse 4-2]$
```

PROGRAM-6A:

write a program to implement K-Means algorithm.

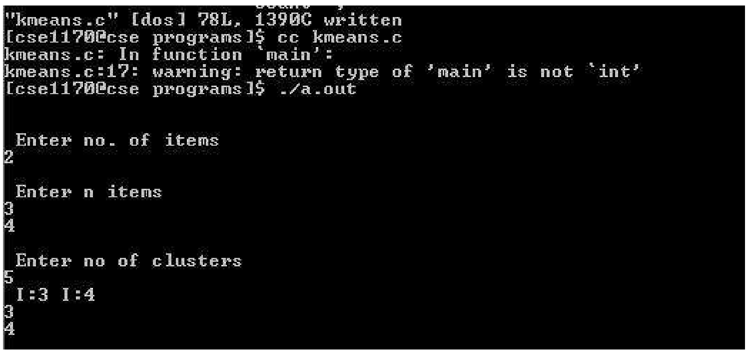
PROBLEM DEFINITION: write a program to implement K-Means algorithm.

SOURCE CODE:

```
#include<stdio.h>
int mod(int k)
{
if(k>0) return k;
else return -k;
}
int small(int b[],int n)
{
 int m,pos,r=0; m=b[0];
for(pos=0;pos<n;pos++)
{
if(m>b[pos]) { m=b[pos];
 r=pos;
}
}
 return r;
}
void main()
{
int n,j,s=0;
int x=0,y=0,z=0;
int obj[20],c[20][20],mean[20],a[20];
int i,nc,k,m,min,count;

printf("\n\n Enter no. of items");
scanf("%d",&n);
printf("\n Enter n items");
for(i=0;i<n;i++)
scanf("%d",&obj[i]);
printf("\n Enter no of clusters");
scanf("%d",&nc);
for(i=0;i<nc;i++)
for(j=0;j<n;j++)
{
c[i][j]=0;   a[i]=0; }
for(i=0;i<nc;i++)
{
```

```
c[i][0]=obj[i];
mean[i]=obj[i];
}
for(i=0;i<nc;i++)
for(j=0;j<n;j++)
 if(c[i][j]>0)
printf(" I:%d",c[i][j]);
j=nc;
for(i=1;i<n;i++)
{
if(j<n)
{
for(k=0;k<nc;k++)
a[k]=mod(obj[j]-mean[k]);
min=small(a,nc);
c[min][i]=obj[j];
for(k=0;k<nc;k++)
{
s=0;count=0;
for(m=0;m<n;m++)
{
if(c[k][m]>0)
{
 s=s+c[k][m];
count++;
}
}
mean[k]=s/count;
}
for(k=0;k<nc;k++)
printf("\n mean values..%d\t",mean[k]);
printf("\n");
 j++;
}
}
for(i=0;i<nc;i++)
{
 printf("\n");
for(j=0;j<n;j++)
{
if(c[i][j]>0)
printf("%d\t",c[i][j]);
}
}
```

Output:


"kmeans.c" [dos] 78L, 1390C written
[cse1170@cse programs]$ cc kmeans.c
kmeans.c: In function `main':
kmeans.c:17: warning: return type of 'main' is not `int'
[cse1170@cse programs]$ ./a.out
Enter no. of items
2
Enter n items
3
4
Enter no of clusters
5
I:3 I:4
3
4

PROGRAM-6B:

K-Means algorithm for two dimension data items.

PROBLEM DEFINITION: K-Means algorithm for two dimension data items.

SOURCE CODE:

```
#include<stdio.h>
#include<math.h>
double distance(int a[][2],double b[][2],int j,int k)
{
double n=0,x1,y1,total;
int x,y;
x=a[j][0]-b[k][0];
y=a[j][1]-b[k][1];
x1=x*x;
y1=y*y;
total=x1+y1;
n=sqrt(total);
return n;
}
int small(double b[],int n)
{
int pos,r=0;double m=b[0];
for(pos=0;pos<n;pos++)
{
if(m>b[pos])
{
m=b[pos];
r=pos;
}
}
return r;
}
void main()
{
int n,j,s=0;
 int x=0,y=0,z=0;
int x1,y1;
int obj[20][2],c[20][20][2];
double mean[20][2];
double a[20];
int i,nc,k,m,min,count;
printf("\n\n Enter no. of items");
```

```
scanf("%d",&n);
printf("\n Enter n items");
for(i=0;i<n;i++)
for(k=0;k<2;k++)
scanf("%d",&obj[i][k]);
printf("\n Enter no of clusters");
scanf("%d",&nc);
for(i=0;i<nc;i++)
for(j=0;j<n;j++)
{
 for(k=0;k<2;k++)
c[i][j][k]=0;   a[i]=0;
}
for(i=0;i<nc;i++)
{
j=0;
for(k=0;k<2;k++)
{
c[i][j][k]=obj[i][k];
mean[i][k]=obj[i][k];
}
}
for(i=0;i<nc;i++)
{
printf("\nI%d:",i);
for(j=0;j<n;j++)
for(k=0;k<2;k++)
if(c[i][j][k]>0)
printf("%d  ",c[i][j][k]);
printf("\n");
}
for(i=0;i<nc;i++)
{
for(k=0;k<2;k++)
printf("\n mean values...%lf  ",mean[i][k]);
printf("\n");
}
j=nc;
for(i=1;i<n;i++)
{
if(j<n)
{
for(k=0;k<nc;k++)
a[k]=distance(obj,mean,j,k);
min=small(a,nc);
c[min][i][0]=obj[j][0];
```

```
c[min][i][1]=obj[j][1];
for(m=0;m<n;m++)
{
 x1=0;y1=0;count=0;
for(k=0;k<nc;k++)
{
if(c[m][k][0]>0||c[m][k][1]>0)
{
x1=x1+c[m][k][0];
y1=y1+c[m][k][1];
count++;
}
}
if(count>0)
{
mean[k][0]=x1/count;
mean[k][1]=y1/count;
}
}
j++;
}
}
for(i=0;i<nc;i++)
{
for(j=0;j<n;j++)
for(k=0;k<2;k++)
printf("%d  ",c[i][j][k]);
printf("\n");
}
printf("final kmean values are....\n");
for(i=0;i<nc;i++)
printf("%lf....%lf\n",mean[i][0],mean[i][1]);
}
```

Output:

```
"kmeans2.c" [dos] 117L, 2291C written
[cse1170@cse programs]$ cc kmeans2.c -lm
kmeans2.c: In function `main':
kmeans2.c:29: warning: return type of 'main' is not `int'
[cse1170@cse programs]$ ./a.out


 Enter no. of items
2

 Enter n items
3
4
5
6


 Enter no of clusters
1

I0:3  4

 mean values...3.000000
 mean values...4.000000
3  4  5  6
final kmean values are....
3.000000....4.000000
[cse1170@cse programs]$
```

PROGRAM-7:

a program to implement k-mediod algorithm

PROBLEM DEFINITION: Write a program to implement k-mediod algorithm

SOURCE CODE:

```
#include<stdio.h>
#include<math.h>
int distance(int [],int []);
int i,j,n,nc=3;
void main()
{
int j,count,t;
int obj[10][2],c[10][10][2],mean[10][2],c1[10][10][2];
int i,k,m,cost=0,cost1;
printf("\n enter the no. of items:");
scanf("%d",&n);
printf("\n enter the items(%d)",n);
for(i=0;i<n;i++)
for(j=0;j<2;j++)
scanf("%d",&obj[i][j]);
for(i=0;i<nc;i++)
for(j=0;j<n;j++)
for(k=0;k<2;k++)
{
c[i][j][k]=0;
c1[i][j][k]=0;
}
```

```
printf("\n enter center points");
for(i=0;i<nc;i++)
for(j=0;j<2;j++)
{
scanf("%d",&mean[i][j]);
c[i][0][j]=mean[i][j];
}
j=0;
for(i=1;i<=n;i++)
{
if(j<n)
{
if(distance(obj[j],mean[0])<distance(obj[j],mean[1]))
if(distance(obj[j],mean[0])<distance(obj[j],mean[2]))
for(k=0;k<2;k++)
{
 c[0][i][k]=obj[j][k];
 cost=cost+distance(obj[j],mean[0]);
}
if(distance(obj[j],mean[1])<distance(obj[j],mean[0]))
if(distance(obj[j],mean[1])<distance(obj[j],mean[2]))
for(k=0;k<2;k++)
{
 c[1][i][k]=obj[j][k];
 cost=cost+distance(obj[j],mean[1]);
}
if(distance(obj[j],mean[2])<distance(obj[j],mean[0]))
if(distance(obj[j],mean[2])<distance(obj[j],mean[1]))
for(k=0;k<2;k++)
{
 c[2][i][k]=obj[j][k];
 cost=cost+distance(obj[j],mean[2]);
}
j++;
}
}
printf("\n enter the next center points:");
for(i=0;i<nc;i++)
for(j=0;j<2;j++)
{
scanf("%d",&mean[i][j]);
c1[i][0][j]=mean[i][j];
}
j=0;
for(i=1;i<=n;i++)
{
```

```
if(j<n)
{
 if(distance(obj[j],mean[0])<distance(obj[j],mean[1]))
 if(distance(obj[j],mean[0])<distance(obj[j],mean[2]))
 for(k=0;k<2;k++)
{
c1[0][i][k]=obj[j][k];
cost1=cost1+distance(obj[j],mean[0]);
}
if(distance(obj[j],mean[1])<distance(obj[j],mean[0]))
if(distance(obj[j],mean[1])<distance(obj[j],mean[2]))
for(k=0;k<2;k++)
{
c1[1][i][k]=obj[j][k];
cost1=cost1+distance(obj[j],mean[1]);
}
if(distance(obj[j],mean[2])<distance(obj[j],mean[0]))
if(distance(obj[j],mean[2])<distance(obj[j],mean[1]))
for(k=0;k<2;k++)
{
c[2][i][k]=obj[j][k];
cost1=cost1+distance(obj[j],mean[2]);
}
 j++;
}
}
if(cost<cost1)
{
for(i=0;i<nc;i++)
{
printf("\n");
for(j=0;j<n;j++)
for(k=0;k<2;k++)
{
 if(c[i][j][k]>0)
printf("%d\t",c[i][j][k]);
}
}
}
else
{
for(i=0;i<nc;i++)
{
printf("\n");
for(j=0;j<n;j++)
for(k=0;k<2;k++)
```

```
{
if(c1[i][j][k]>0)
printf("%d\t",c1[i][j][k]);
}
}
}
}
 int distance(int obj[],int mean[])
{
 int x1,x2,y1,y2,dist;
 x1=obj[0];
 x2=mean[0];
 y1=obj[1];
 y2=mean[1];
dist=(sqrt(pow((x1-x2),2)+pow((y1-y2),2)));
 return dist;
 }
```

Output:

CASE STUDY 1- K-NEAREST NEIGHBOR

DEFINITION:

Nearest-neighbor classifiers are based on learning by analogy, that is, by comparing a given test tuple with training tuples that are similar to it. The training tuples are described by n attributes. Each tuple represents a point in an n dimensional space. In this way, all of the training tuples are stored in an n dimensional pattern space. When given an unknown tuple, a k nearest neighbor classifier searches the pattern space for the k training tuples that are closest to the unknown tuple. These k training tuples are the k "nearest neighbors" of the unknown tuple.

WORKING:

The K nearest-neighbor method was first described in the early 1950s. The method is labor intensive when given large training sets, and did not gain popularity until the 1960s when increased computing power became available. It has since been widely used in the area of pattern recognition.

"Closeness" is defined in terms of a distance metric, such as Euclidean distance. The Euclidean distance between two points or tuples

```
XQ=(x11, x12 x1n) and

X2 = (x21, x22... x2n), is
```

$$d(i,j)=\sqrt{(|x_{i1}-x_{j1}|^2+|x_{i2}-x_{j2}|^2+\ldots+|x_{ip}-x_{jp}|^2)}$$

In other words, for each numeric attribute, we take the difference between the corresponding values of that attribute in tuple Xi and in tuple X2, square this difference, and accumulate it. The square root is taken of the total accumulated distance count.

Typically, we normalize the values of each attribute before using Equation This helps prevent attributes with initially large ranges (such as income) from outweighing attributes with initially smaller ranges (such as binary attributes). Min-max normalization, for example, can be used to transform a value v of a numeric attribute A to v' in the range [0, 1] by computing

$$v' = \frac{v - min_A}{max_A - min_A},$$

where minA and maxA are the minimum and maximum values of attribute A. For k-nearest neighbor classification, the unknown tuple is assigned the most common class among its k nearest neighbors. When k = 1, the unknown tuple is assigned the class of the training tuple that is closest to it in pattern space. Nearest- neighbor classifiers can also be used for prediction, that is, to return a real-valued prediction for a given unknown tuple. In this case, the classifier returns the average value of the real-valued labels associated with the k nearest neighbors of the unknown tuple.

To compute for attributes that not numeric, but categorical, such as color The above discussion assumes that the attributes used to describe the tuples are all numeric. For categorical attributes, a simple method is to compare the corresponding value of the attribute in tuple X1 with that in tuple X2. If the two are identical (e.g., tuples X1 and X2 both have the color blue), then the difference between the two is taken as 0. If the two are different (e.g., tuple X1 is blue but tuple X2 is red), then the difference is considered to be 1. Other methods may incorporate more sophisticated schemes for differential grading (e.g., where a larger difference score is assigned, say, for blue and white than for blue and black).

The procedure to compute if the there are missing values In general, if the value of a given attribute A is missing in tuple X1 and/or in tuple X2, we assume the maximum possible difference. Suppose that each of the attributes have been mapped to the range [0, 1] . For categorical attributes, we take the difference value to be 1 if either one or both of the corresponding values of A are missing. If A is numeric and missing from both tuples X1 and X2, then the difference is also taken to be 1. If only one value is missing and the other (which we'll call v') is present and normalized, then we can take the difference to be either

```
|1  v'| or |0  v'|
(i.e., 1  v' or v'), whichever is greater.
```

To determine a good value for k, the number of neighbors

This can be determined experimentally. Starting with k = 1, we use a test set to estimate the error rate of the classifier. This process can be repeated each time by incrementing k to allow for one more neighbors. The k value that gives the minimum error rate may be selected. In general, the larger the number of training tuples is, the larger the value of k will be (so that classification and prediction decisions can be based on a larger portion of the stored tuples). As the number of training tuples approaches infinity and k = 1, the error rate can be no worse then twice the Bayes error rate (the latter being the theoretical minimum). If k also approaches infinity, the error rate approaches the Bayes error rate.

DISADVANTAGES OF KNN AND TECHINIQUES TO OVERCOME THEM:

Nearest-neighbor classifiers use distance-based comparisons that intrinsically assign equal weight to each attribute. They therefore can suffer from poor accuracy when given noisy or irrelevant attributes. The method, however, has been modified to incorporate attribute weighting and the pruning of noisy data tuples. The choice of a distance metric can be critical. The Manhattan (city block) distance or other distance measurements may also be used.

Nearest neighbor classifiers can be extremely slow when classifying test tuples. If D is a training database of $\|D\|tuplesandk = 1, thenO(\|D\|)comparisonsarerequiredinordertoclass$

Parallel implementation can reduce the running time to a constant that is 0(1), which is independent of $\|D\|$.

Other techniques to speed up classification time include the use of partial distance calculations and editing the stored tuples. In the partial distance method, we compute the distance based on a subset of the n attributes. If this distance exceeds a threshold, then further computation for the given stored tuple is halted, and the process moves on to the next stored tuple. The editing method removes training tuples that prove useless. This method is also referred to as pruning or condensing because it reduces the total number of tuples stored.

CASE STUDY 2 - KDD PROCESS

DEFINITION:

KDD Process is the process of using data mining methods (algorithms) to extract (identify) what is deemed knowledge according to the specifications of measures and thresholds, using database F along with any required preprocessing, subsampling, and transformation of F.

KDD:

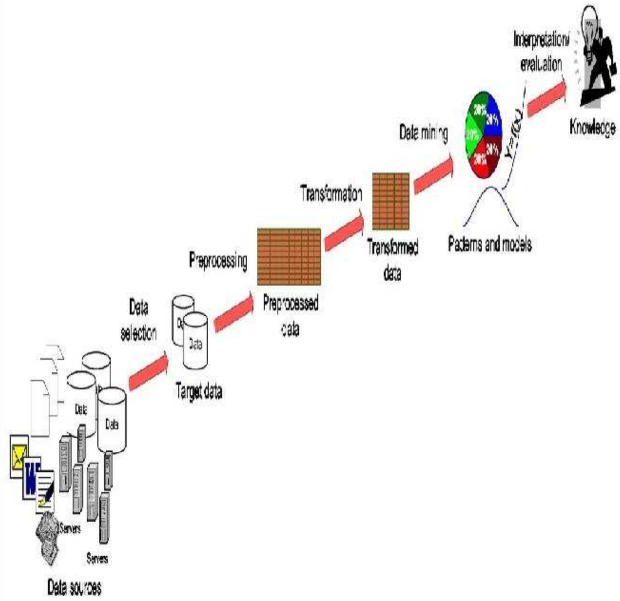
- In a multistep process many decisions are made by the user (domain expert):
- Iterative and interactive loops between any two steps are possible
- Usually the most focus is on the DM step, but other steps are of considerable importance for the successful application of KDD in practice

GOALS:

- Verification of users hypothesis (this against the EDA principle)
- Autonomous discovery of new patterns and models
- Prediction of future behavior of some entities
- Description of interesting patterns and models

STEPS OF DM:

a. Domain understanding and goal setting
b. Creating a target data set
c. Data cleaning and preprocessing
d. Data reduction and projection
e. Data mining
    - Choosing the data mining task
    - Choosing the data mining algorithm(s)
    - Use of data mining algorithms
f. Interpretation of mined patterns
g. Utilization of discovered knowledge



a. **Domain analysis**
- Development of domain understanding
- Discovery of relevant prior knowledge
- Definition of the goal of the knowledge discovery

b. **Data selection**
- Selection and integration of the target data from possibly many different and heterogeneous sources
- Interesting data may exist, e.g., in relational databases, document collections, e mails, photographs, video clips, process database, customer transaction database, web logs etc.
- Focus on the correct subset of variables and data samples
- E.g., customer behavior in a certain country, relationship between items purchased and customer income and age

c. **Data cleaning and preprocessing**
- Dirty data can confuse the mining procedures and lead to unreliable and invalid outputs
- Complex analysis and mining on a huge amount of data may take a very long time
- Preprocessing and cleaning should improve the quality of data and mining results by enhancing the actual mining process
- The actions to be taken includes
- Removal of noise or outliers
- Collecting necessary information to model or account for noise

- Using prior domain knowledge to remove the inconsistencies and duplicates from the data
- Choice or usage of strategies for handling missing data fields

d. **Data reduction and projection**

- Data transformation techniques
- Smoothing (binning, clustering, regression etc.)
- Aggregation (use of summary operations (e.g., averaging) on data)
- Generalization (primitive data objects can be replaced by higher level concepts)
- Normalization (min max-scaling, z score)
- Feature construction from the existing attributes (PCA, MDS)
- Data reduction techniques are applied to produce reduced representation of the data (smaller volume that closely maintains the integrity of the original data)
- Aggregation
- Dimension reduction (Attribute subset selection, PCA, MDS,)
- Compression (e.g., wavelets, PCA, clustering,)
- Numerosity reduction
- parametric models: regression and log linear models
- non parametric models: histograms, clustering, sampling
- Discretization (e.g., binning, histograms,cluster analysis,)
- Concept hierarchy generation (numeric value of age to a higher level concept young, middle aged, senior)

e. **Choice of data mining task**

- Define the task for data mining
- Exploration/summarization
- Summarizing statistics (mean, median, mode, std,..)
- Class/concept description
- Explorative data analysis
- Graphical techniques, low dimensional plots,
- Predictive
- Classification or regression
- Descriptive
- Cluster analysis, dependency modelling, change and outlier detection

f. **Choosing the DM algorithm(s)**

- Select the most appropriate methods to be used for the model and pattern search
- Matching the chosen method with the overall goal of the KDD process (necessites communication between the end user and method specialists)
- Note that this step requires understanding in many fields, such as computer science, statistics, machine learning, optimization, etc.

g. **Use of data mining algorithms**
   - Application of the chosen DM algorithms to the target data set
   - Search for the patterns and models of interest in a particular representational form or a set of such representations
   - Classification rules or trees, regression models, clusters, mixture models
   - Should be relatively automatic
   - Generally DM involves:
   - Establish the structural form (model/pattern) one is interested
   - Estimate the parameters from the available data
   - Interprete the fitted models

h. **Interpretation/evaluation**
   - The mined patterns and models are interpreted
   - The results should be presented in understandable form
   - Visualization techniques are important for making the results useful mathematical models or text type descriptions may be difficult for domain experts
   - Possible return to any of the previous step

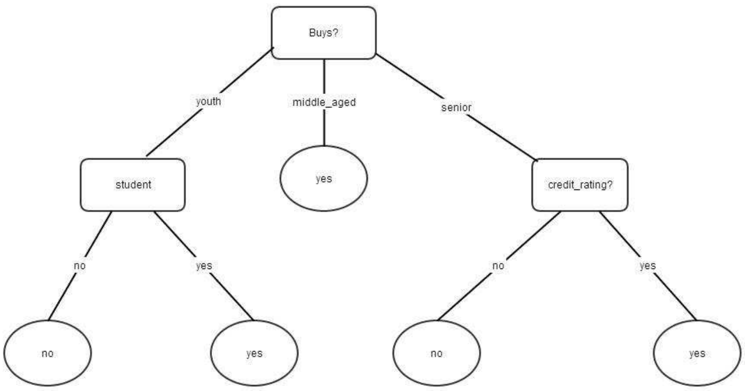
CASE STUDY 3 DECISION TREE INDUCTION

DEFINITION:

Decision tree induction is the learning of decision trees from class-labeled training tuples. A decision tree is a flow chart like tree structure, where each internal node(non leaf node) denotes a test on an attribute , each branch represents an outcome of the test and each leaf node(or terminal node) holds a class label. The topmost node in a tree is the root node.

EXAMPLE:

A decision tree for buys _ computer , indicating whether a customer at AllElectronics is likely to purchase a computer. Each internal node represents a test on an attribute. Each leaf node represents a class (either buys _ computer=yes or buys _ computer=no)



ALGORITHM:

A machine researcher named J. Ross Quinlan in 1980 developed a decision tree algorithm. This Decision Tree Algorithm is known as ID3(Iterative Dichotomiser). Later he gave C4.5 which was successor of ID3. ID3 and C4.5 adopt a greedy approach. In this algorithm there is no backtracking, the trees are constructed in a top down recursive divide-and-conquer manner.

Generating a decision tree form training tuples of data partition D

Algorithm : Generate_decision_tree

Input:

```
Data partition, D, which is a set of training tuples
and their associated class labels.
attribute\_list, the set of candidate attributes.
Attribute selection method, a procedure to determine the
splitting criterion that best partitions that the data
tuples into individual classes. This criterion includes a
splitting\_attribute and either a splitting point
 or splitting subset.
```

Output:

A Decision Tree

Method:

```
create a node N;
if tuples in D are all of the same class, C then
   return N as leaf node labeled with class C;
if attribute_list is empty then
   return N as leaf node with labeled
   with majority class in D;|| majority voting
apply attribute_selection_method(D, attribute_list)
to find the best splitting_criterion;
label node N with splitting_criterion;
if splitting_attribute is discrete-valued and
   multiway splits allowed then
// no restricted to binary trees
attribute_list = splitting attribute;
 // remove splitting attribute
for each outcome j of splitting criterion
   // partition the tuples and grow subtrees
for each partition
   let Dj be the set of data tuples in D
satisfying outcome j;
// a partition
   if Dj is empty then
      attach a leaf labeled with the majority
      class in D to node N;
   else
      attach the node returned by Generate
      decision tree(Dj, attribute list) to node N;
```

```
    end for
return N;
```

TREE PRUNING

Tree Pruning is performed in order to remove anomalies in training data due to noise or outliers. The pruned trees are smaller and less complex.

Tree Pruning Approaches

Here is the Tree Pruning Approaches listed below:

- Pre pruning - The tree is pruned by halting its construction early.
- Post pruning - This approach removes sub-tree form fully grown tree.

COST COMPLEXITY

The cost complexity is measured by following two parameters:

- Number of leaves in the tree
- Error rate of the tree

ADVANTAGES OF DECISION TREE

- It does not require any domain knowledge.
- It is easy to assimilate by human.
- Learning and classification steps of decision tree are simple and fast.

CASE STUDY 4 APRIORI

Problem Definition

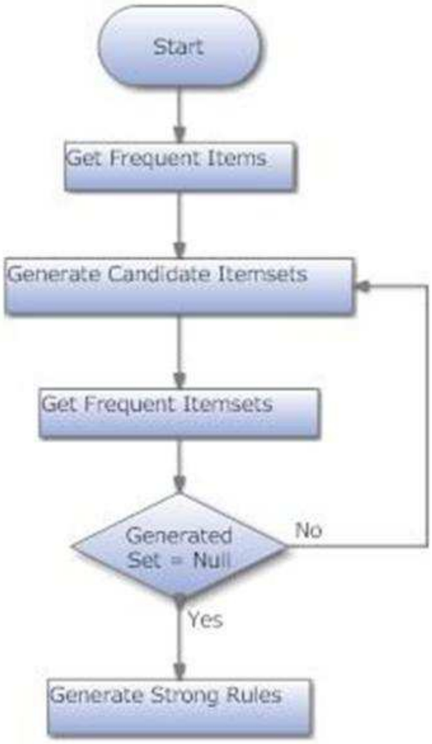
Apriori is an algorithm for frequent item set mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis.

WORKING:

Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation). Other algorithms are designed for finding association rules in data having no transactions (Winepi and Minepi), or having no timestamps (DNA sequencing). Each transaction is seen as a set of items (an itemset). Given a threshold , the Apriori algorithm identifies the item sets which are subsets of at least transactions in the database.

Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

Apriori uses breadth-first search and a Hash tree structure to count candidate item sets efficiently. It generates candidate item sets of length from item sets of length . Then it prunes the candidates which have an infrequent sub pattern. According to the downward closure lemma, the candidate set contains all frequent -length item sets. After that, it scans the transaction database to determine frequent item sets among the candidates.


Start
Get Frequent Items
Generate Candidate Itemsets
Get Frequent Itemsets
Generated
Set = Null
No
Yes
Generate Strong Rules

EXAMPLE:

Consider the following database, where each row is a transaction and each cell is an individual item of the transaction:

```
alpha beta epsilon
alpha beta Theta
alpha beta epsilon
alpha beta Theta
```

The association rules that can be determined from this database are the following:

a. 100% of sets with alpha also contain beta
b. 50% of sets with alpha, beta also have epsilon
c. 50% of sets with alpha, beta also have theta we can also illustrate this through variety of examples

LIMITATIONS:

Apriori, while historically significant, suffers from a number of inefficiencies or trade-offs, which have spawned other algorithms. Candidate generation generates large numbers of subsets (the algorithm attempts to load up the candidate set with as many as possible before each scan). Bottom-up subset exploration (essentially a breadth-first traversal of the subset lattice) finds any maximal subset 2sup $|S| - 1 onlyafterallofitspropersubsets$.

Later algorithms such as Max-Miner try to identify the maximal frequent item sets without enumerating their subsets, and perform ”jumps” in the search space rather than a purely bottom-up approach.

ANNEXURE I
List of programs according to O.U. Curriculum
WITH EFFECT FROM THE ACADEMIC YEAR 2013-2014
DATA MINING LAB CS 481

Instruction	2	Periods per week
Duration of University Examination	3	Hours
University Examination	50	Marks
Sessional	25	Marks

1. Implement the following Multidimensional Data Models
i. Star Schema
ii. Snowflake Schema
iii. Fact Constellation

2. Implement Apriori algorithm to generate frequent Item Sets

3. Implement the following clustering algorithms
i. K-means
ii. K-mediods

4. Implement the following classification algorithms
i. Decision Tree Induction
ii. KNN

5. Perform data Pre processing using WEKA

6. Perform Discretization of data using WEKA

7. Classification algorithms using WEKA

8. Apriori algorithm using WEKA

9. Perform data transformations using an ETL Tool

10. A small case study involving all stages of KDD.
(Datasets are available online like UCI Repository etc.)