Ramniranjan Jhunjhunwala College of Arts, Science and Commerce

Department of Data Science and Artificial Intelligence

CERTIFICATE

This is to certify that Ankita Adhik Kanse of Msc. Data Science and Artificial Intelligence Roll No 731 has successfully completed the practical of Paper – I(Semester-III) BIG DATA ANALYTICS during the Academic Year 2023-2024.

Date :

(Prof. Mujtaba Shaikh)
Prof-In-Charge **External Examiner**

INDEX

Practical No. 01

Kappa Statistics :
It is going to measure the precision of the data items. It is used to determine the chance agreement due to guessing a possibility in the same way the chances of correct answers is possible on multiple tests.

Absolute Error :
Amount of error calculated.

Mean Absolute Error :
The mean absolute error is the average of all Absolute Error.

Root Mean Squared Error :
It measures the difference between the values which are predicted by a model and the actual value.

Relative Absolute Error :
The absolute error gives how large the error is, while the relative error gives how large error is related to the correct value.

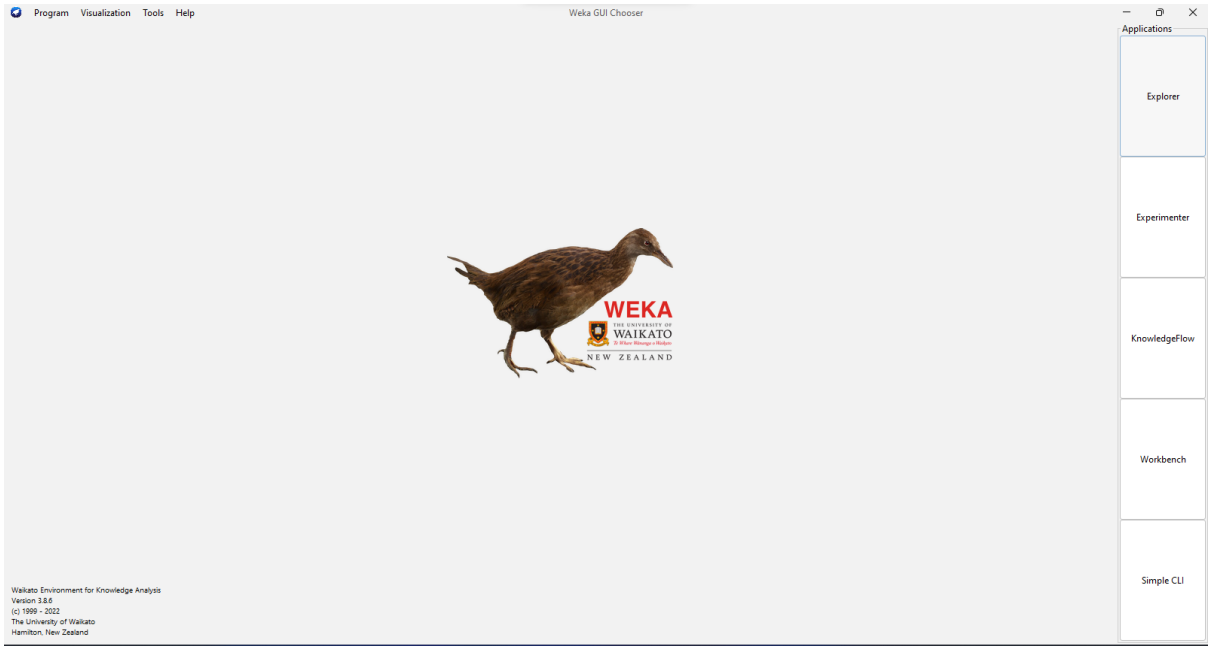
Root Relative Squared Error :
It is relative to what it would have been if a simple predictor had been used.

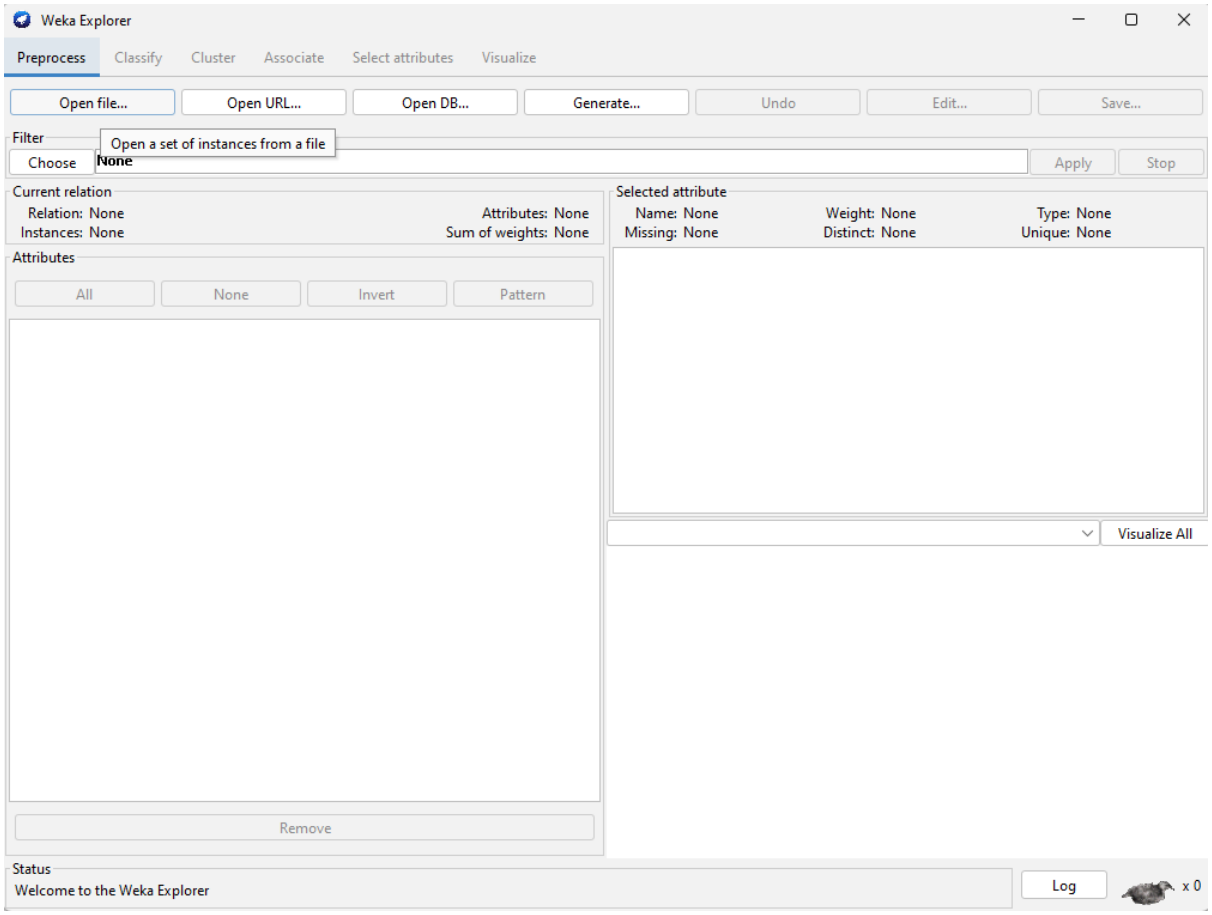
TP(True Positive)
FP(False Positive)

Precision :
Almost near to accuracy.

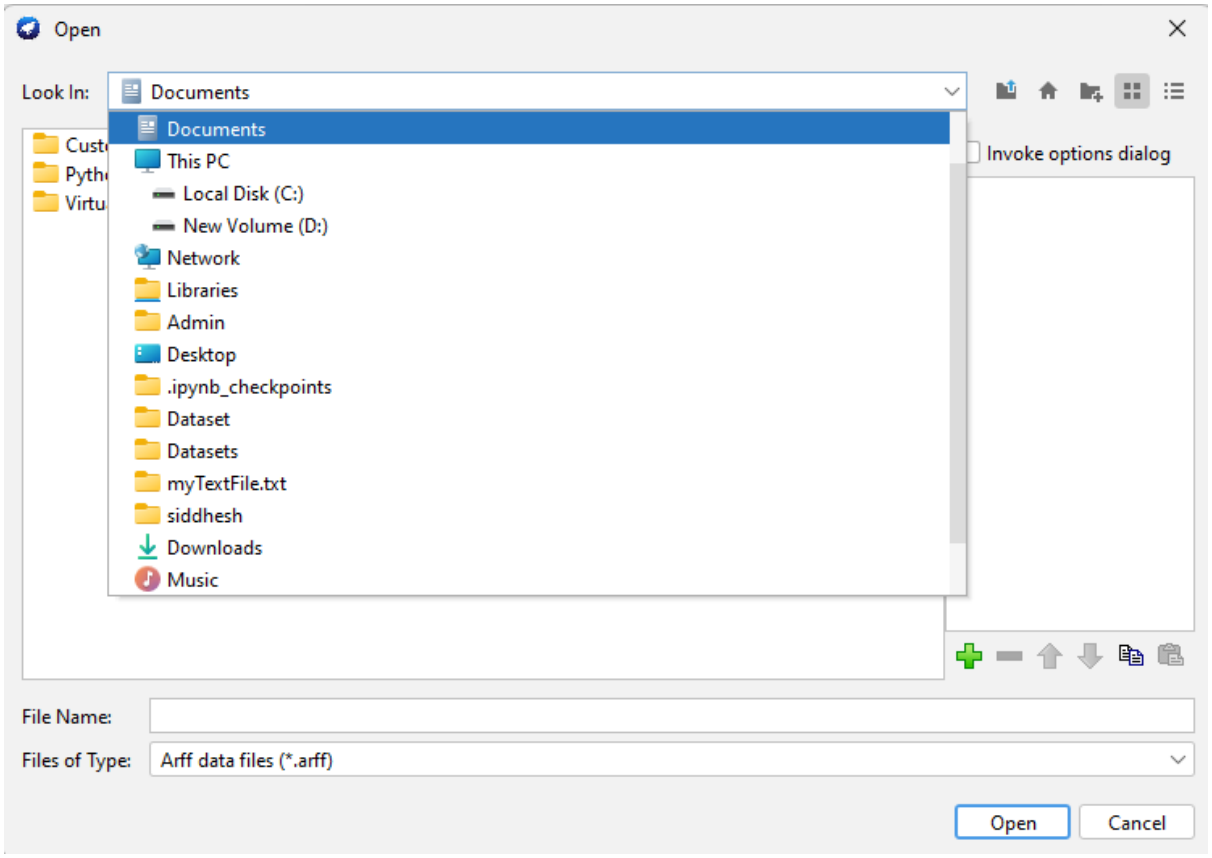
Practical :

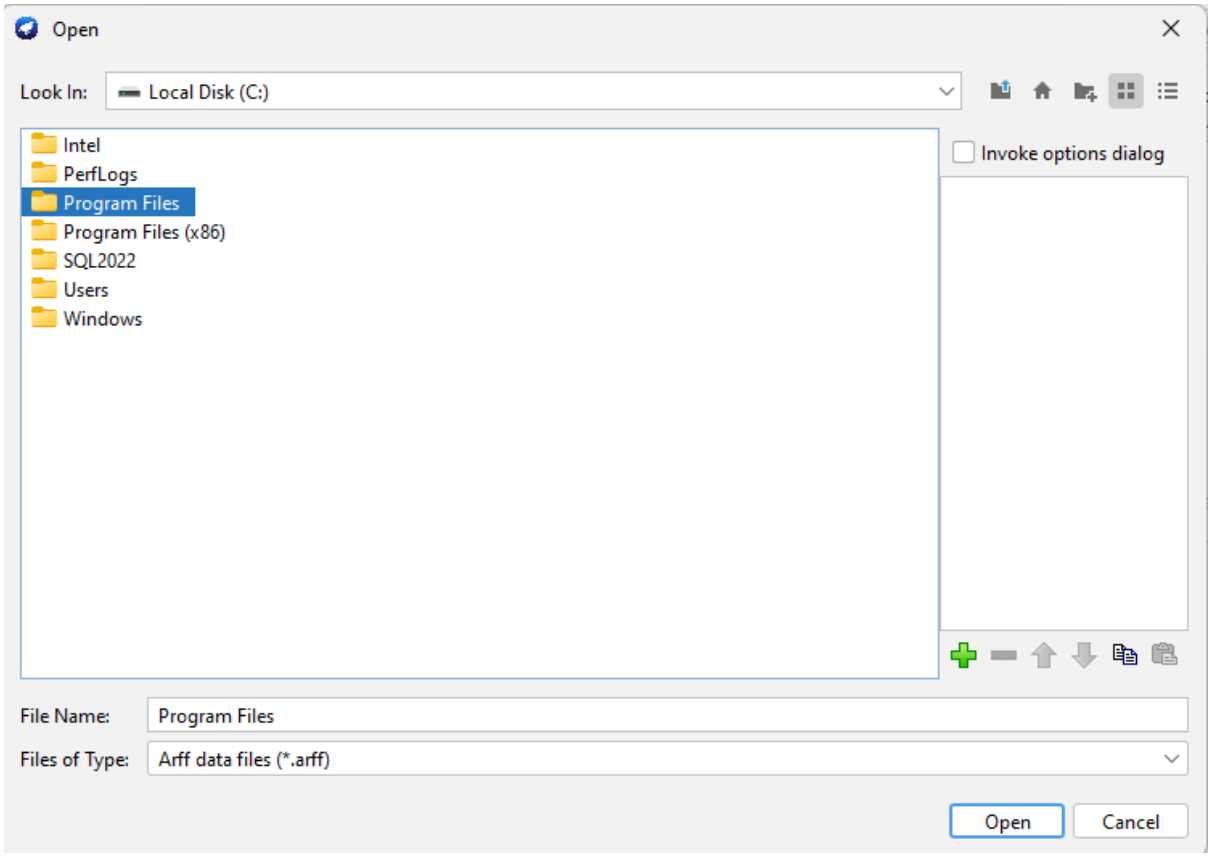
1. **Decision Tree**

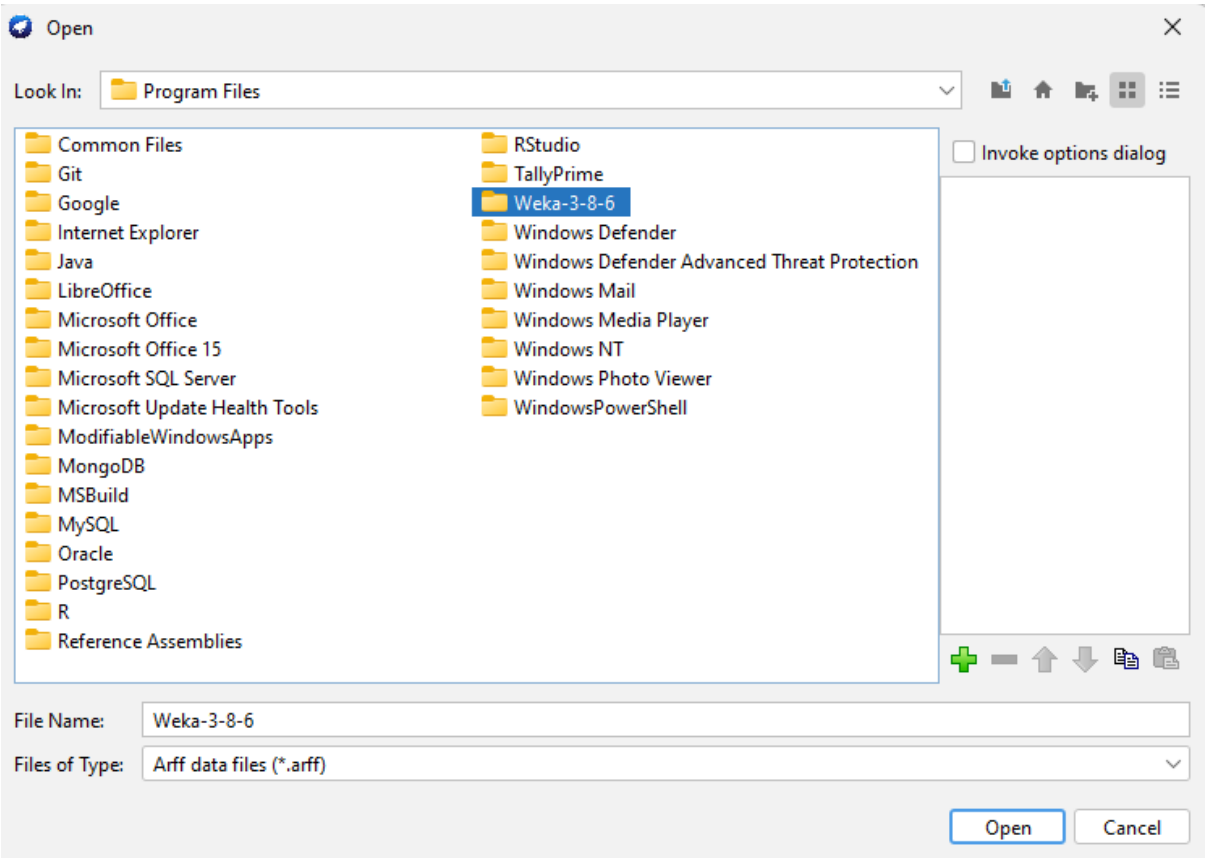


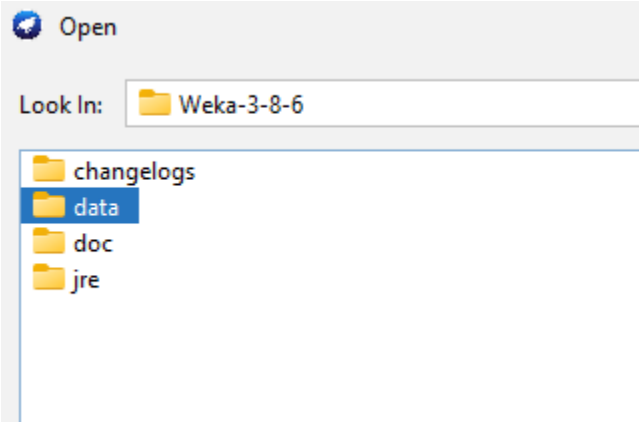


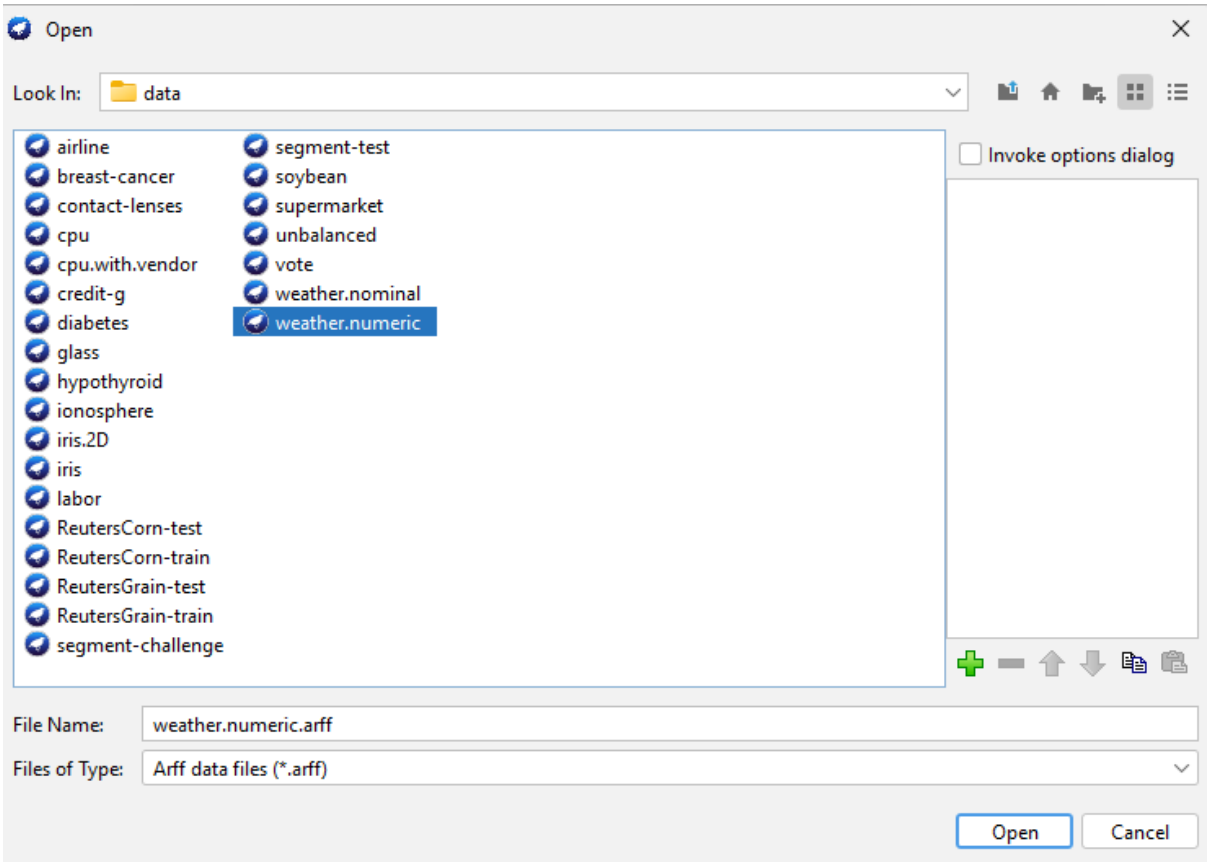
Click on Open FIle

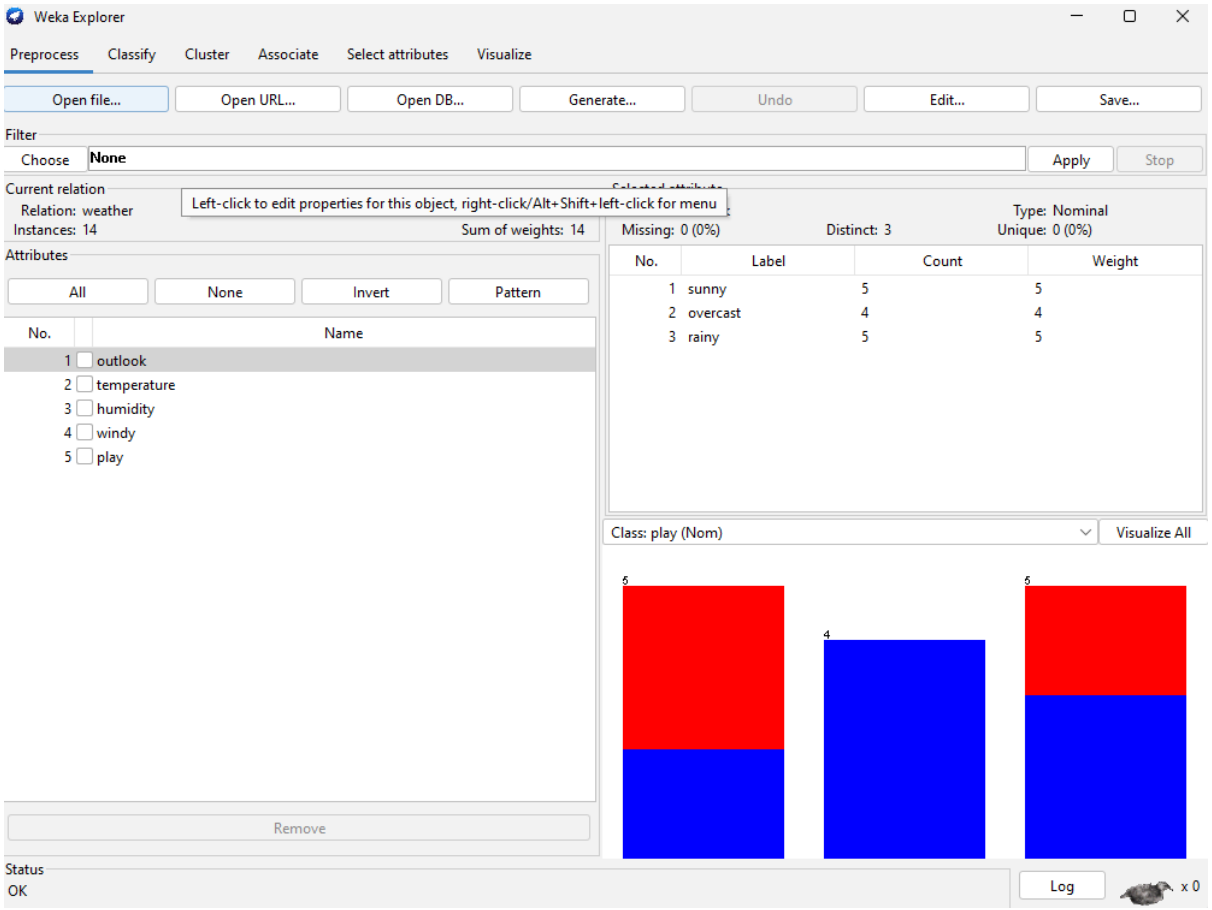

Open
Look In:
Documents
Documents
This PC
Local Disk (C:)
New Volume (D:)
Network
Libraries
Admin
Desktop
.ipynb_checkpoints
Dataset
Datasets
myTextFile.txt
siddhesh
Downloads
Music
Invoke options dialog
File Name:
Files of Type:
Arff data files (*.arff)
Open
Cancel


Open
Look In:
Local Disk (C:)
Intel
PerfLogs
Program Files
Program Files (x86)
SQL2022
Users
Windows
Invoke options dialog
File Name:
Program Files
Files of Type:
Arff data files (*.arff)
Open
Cancel

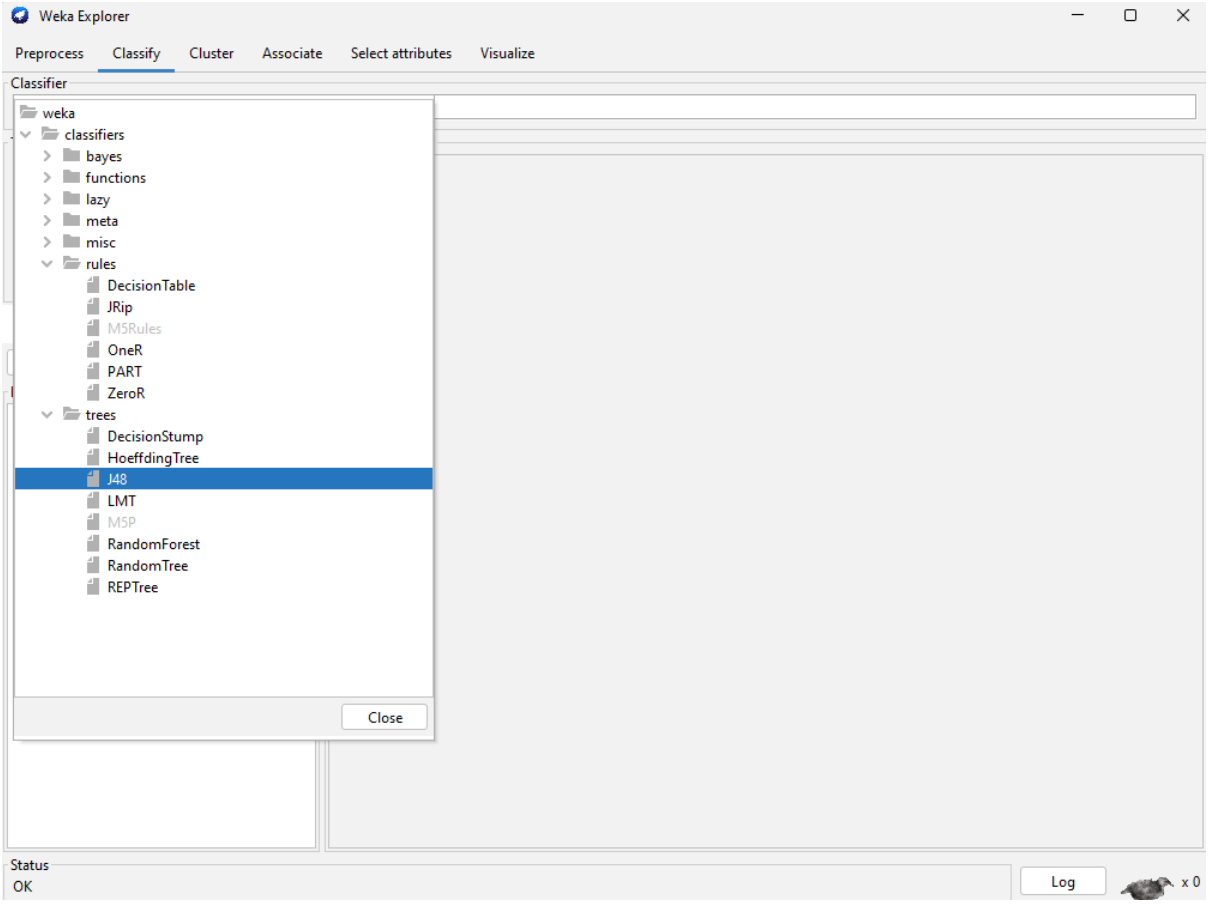

Open
Look In:
Program Files
Common Files
Git
Google
Internet Explorer
Java
LibreOffice
Microsoft Office
Microsoft Office 15
Microsoft SQL Server
Microsoft Update Health Tools
ModifiableWindowsApps
MongoDB
MSBuild
MySQL
Oracle
PostgreSQL
R
Reference Assemblies
RStudio
TallyPrime
Weka-3-8-6
Windows Defender
Windows Defender Advanced Threat Protection
Windows Mail
Windows Media Player
Windows NT
Windows Photo Viewer
WindowsPowerShell
Invoke options dialog
File Name:
Weka-3-8-6
Files of Type:
Arff data files (*.arff)
Open
Cancel

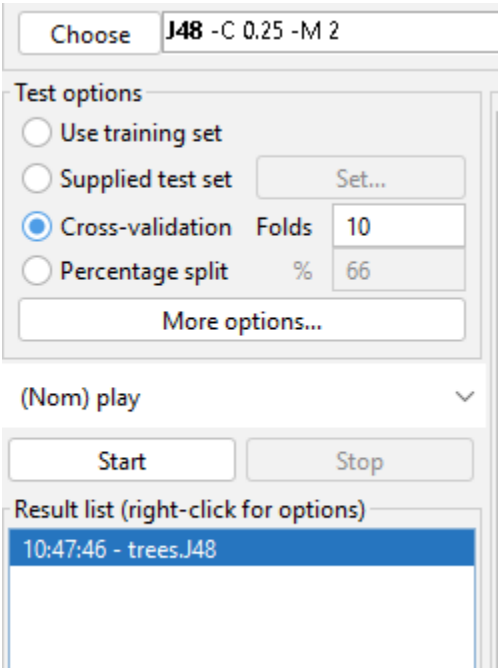

Open
Look In:
Weka-3-8-6
changelogs
data
doc
jre


Open
Look In:
data
airline
breast-cancer
contact-lenses
cpu
cpu.with.vendor
credit-g
diabetes
glass
hypothyroid
ionosphere
iris.2D
iris
labor
ReutersCorn-test
ReutersCorn-train
ReutersGrain-test
ReutersGrain-train
segment-challenge
segment-test
soybean
supermarket
unbalanced
vote
weather.nominal
weather.numeric
Invoke options dialog
File Name:
weather.numeric.arff
Files of Type:
Arff data files (*.arff)
Open
Cancel



Click on Classify Tab and Click Choose J48 and Click on Start


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
weka
classifiers
bayes
functions
lazy
meta
misc
rules
DecisionTable
JRip
M5Rules
OneR
PART
ZeroR
trees
DecisionStump
HoeffdingTree
J48
LMT
M5P
RandomForest
RandomTree
REPTree
Close
Status
OK
Log
x 0


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)
10:47:46 - trees.J48

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)

10:47:46 - trees.J48

Classifier output

```
Number of Leaves  :     5

Size of the tree :      8


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances           9               64.2857 %
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  MCC      ROC Area  PRC Area  Class
                 0.778    0.600    0.700      0.778    0.737      0.189    0.789     0.847     yes
                 0.400    0.222    0.500      0.400    0.444      0.189    0.789     0.738     no
Weighted Avg.    0.643    0.465    0.629      0.643    0.632      0.189    0.789     0.808

=== Confusion Matrix ===

 a b   <-- classified as
 7 2 | a = yes
 3 2 | b = no
```

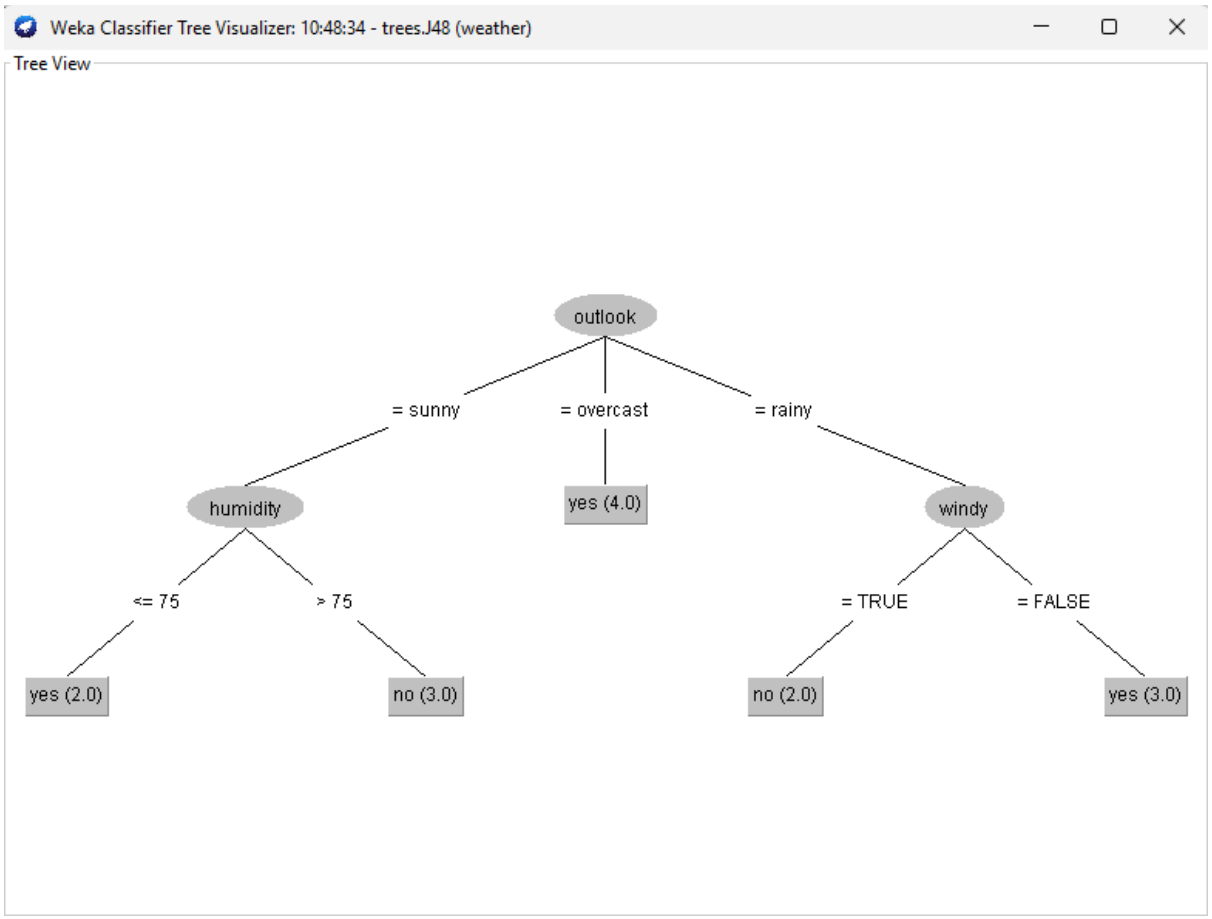
Status

OK

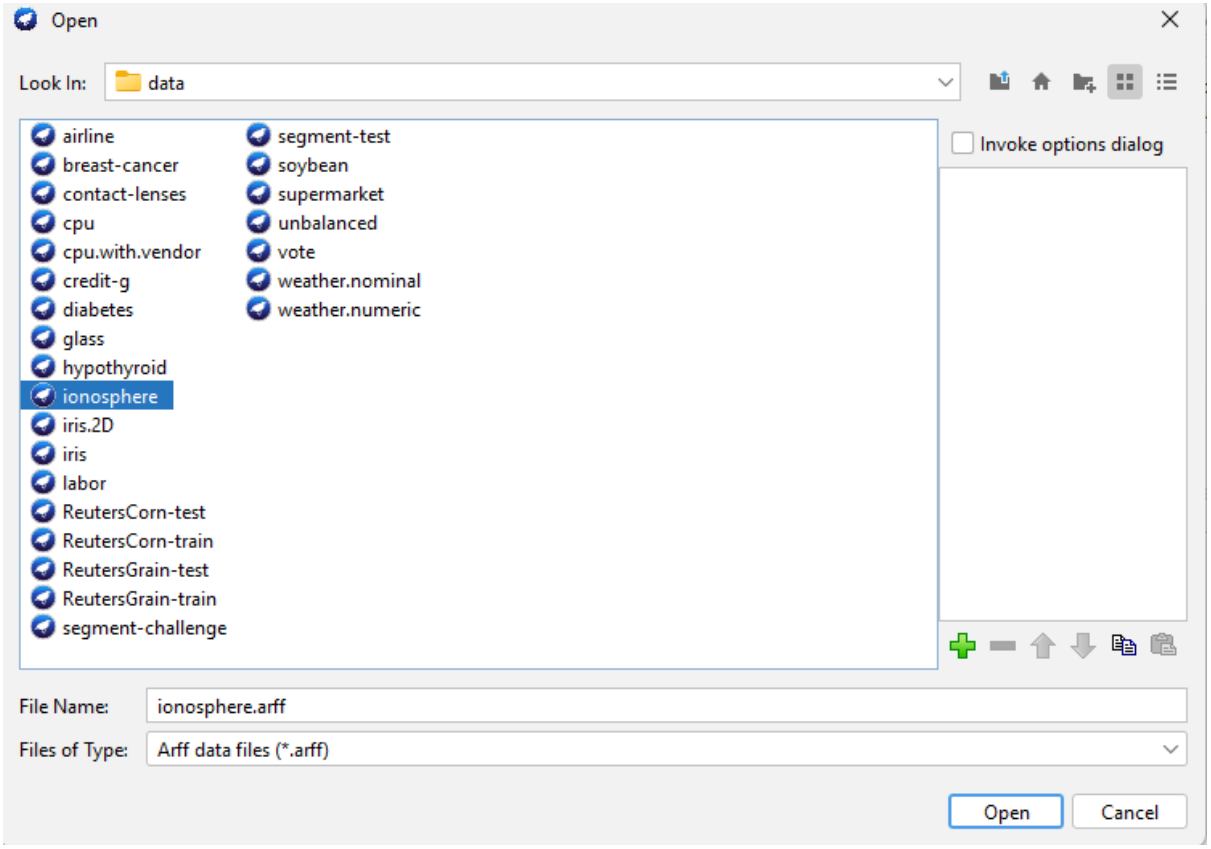
Log | x 0

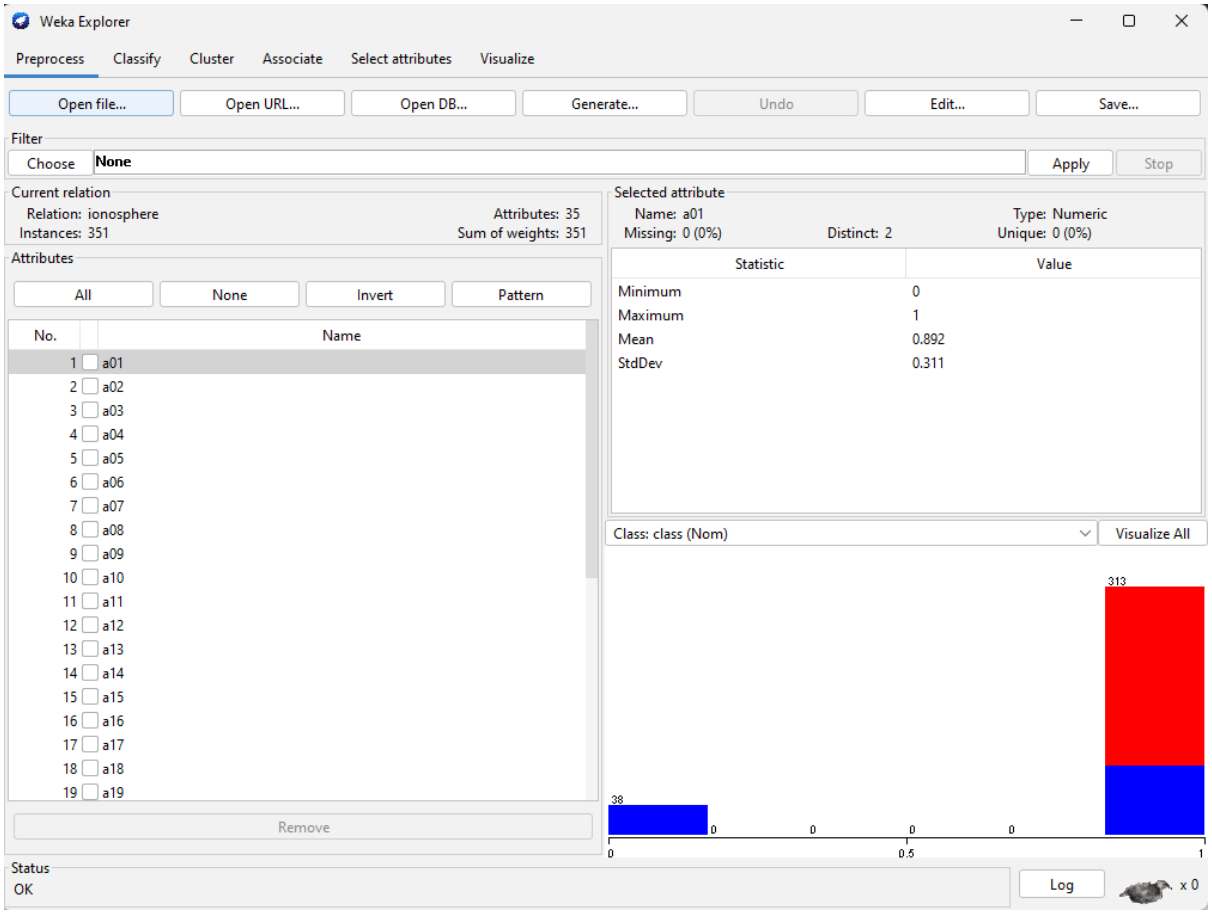
Practical No. 02

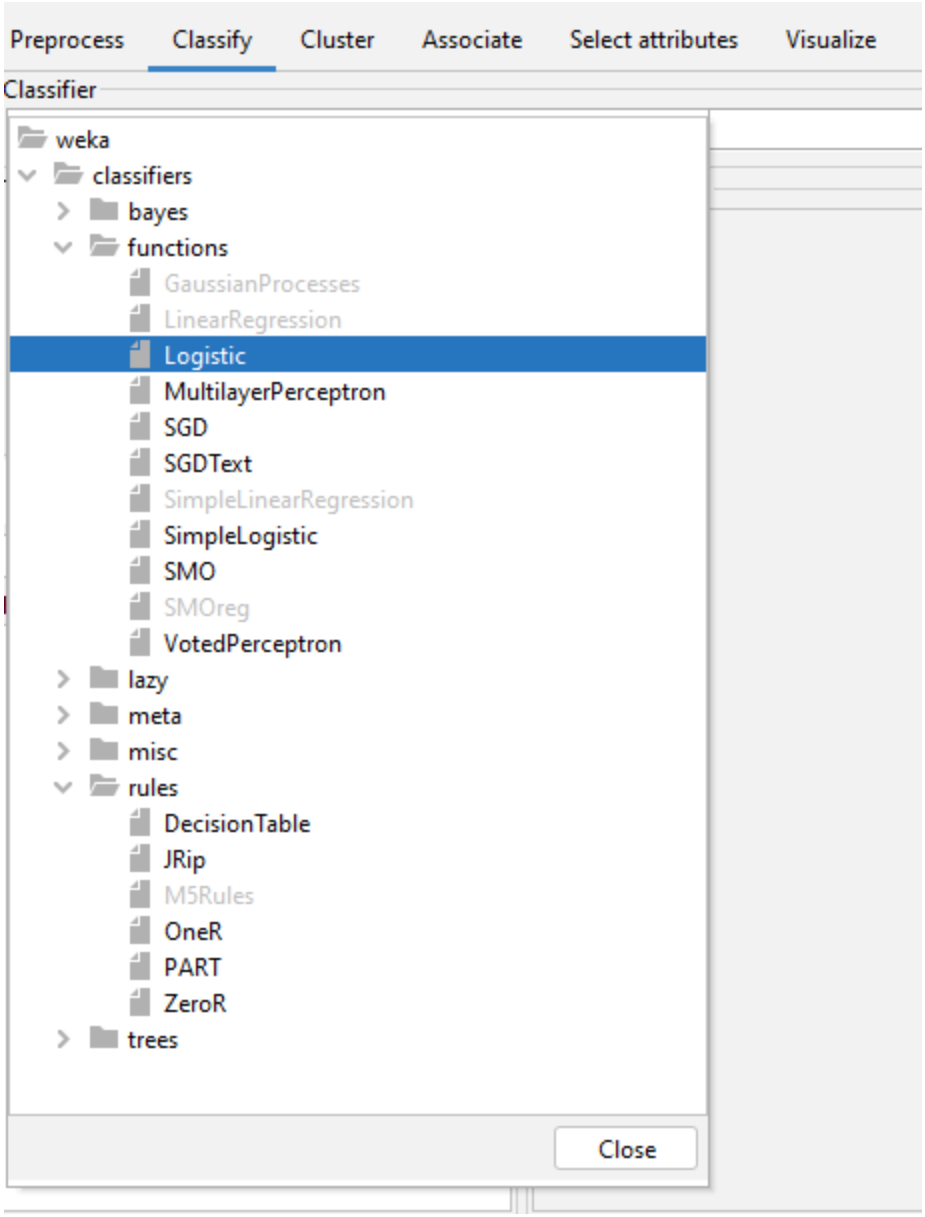
Implementation Using Ml Algorithms

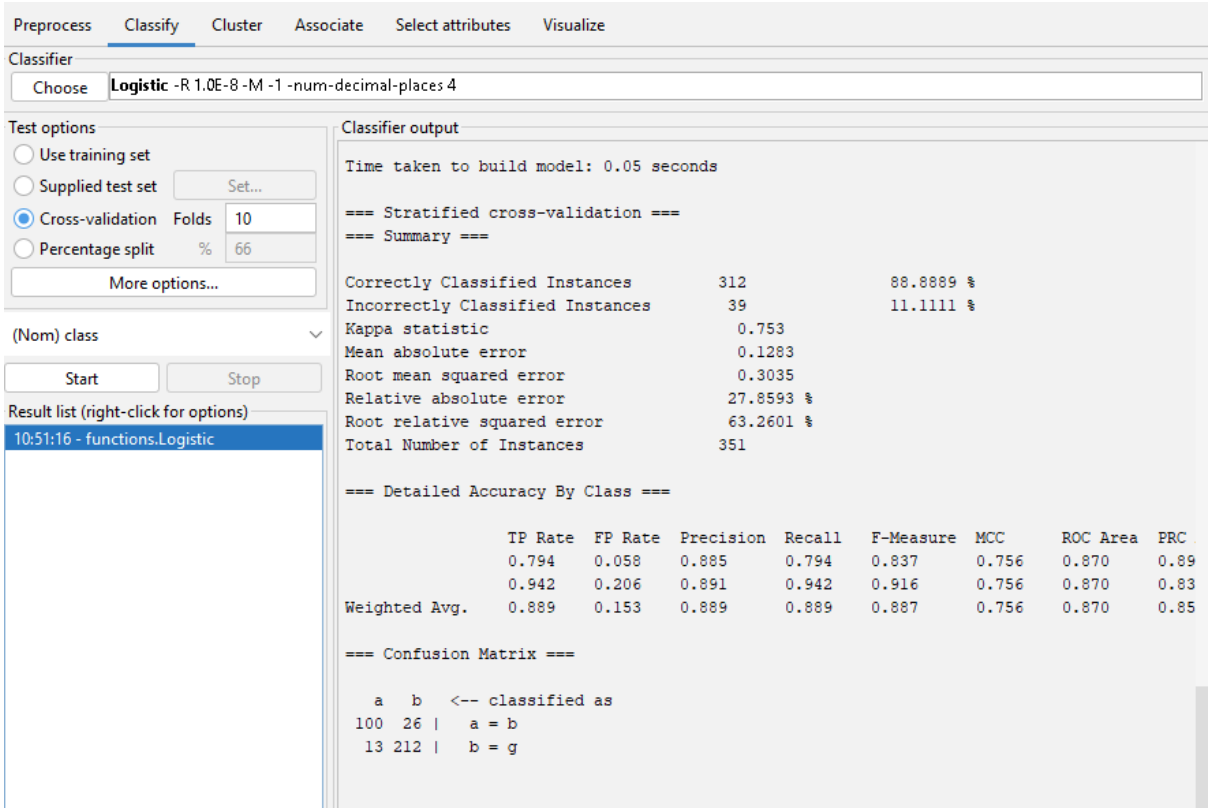


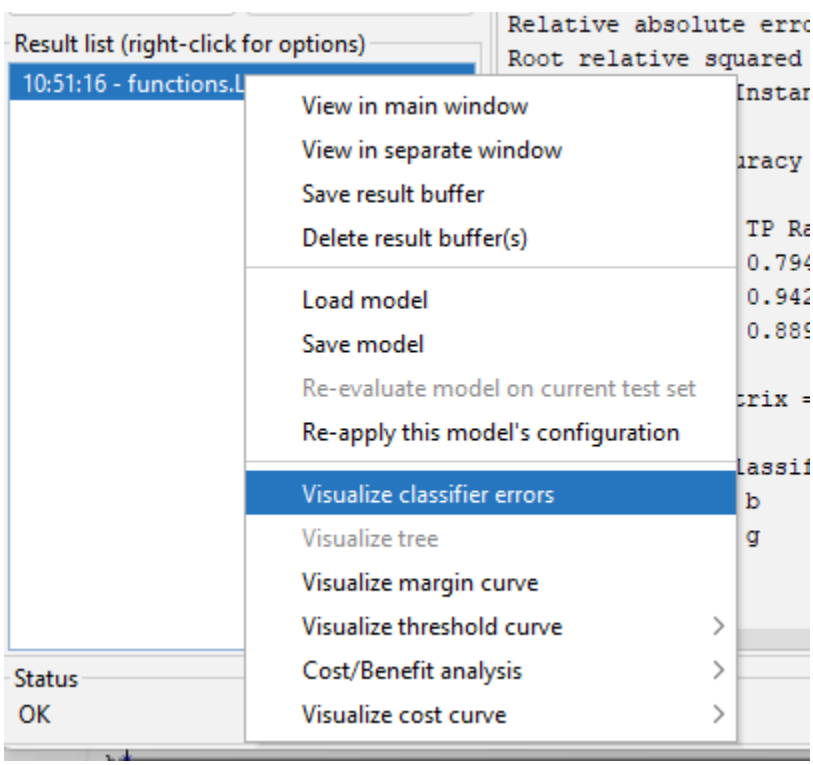
2. **Logistic**

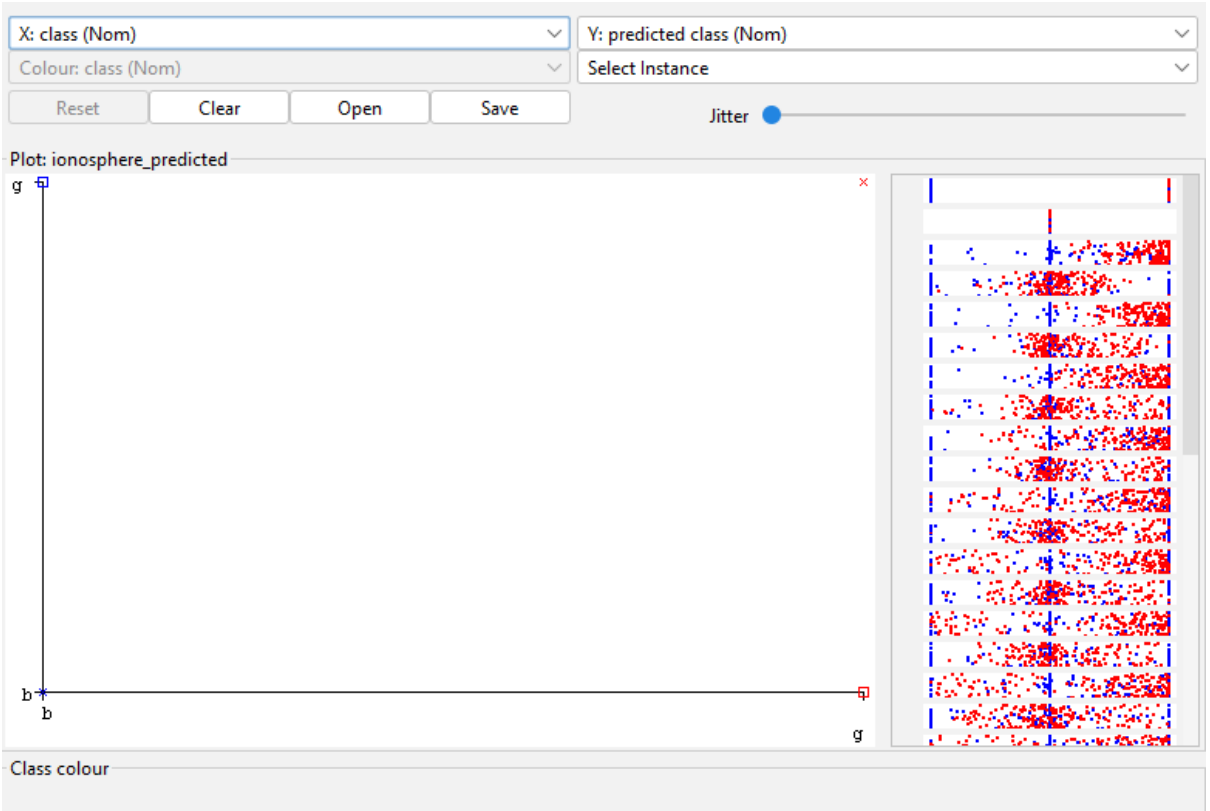

Open
Look In:
data
airline
breast-cancer
contact-lenses
cpu
cpu.with.vendor
credit-g
diabetes
glass
hypothyroid
ionosphere
iris.2D
iris
labor
ReutersCorn-test
ReutersCorn-train
ReutersGrain-test
ReutersGrain-train
segment-challenge
segment-test
soybean
supermarket
unbalanced
vote
weather.nominal
weather.numeric
Invoke options dialog
File Name:
ionosphere.arff
Files of Type:
Arff data files (*.arff)
Open
Cancel


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: ionosphere
Instances: 351
Attributes: 35
Sum of weights: 351
Attributes
All
None
Invert
Pattern
No.
Name
1 a01
2 a02
3 a03
4 a04
5 a05
6 a06
7 a07
8 a08
9 a09
10 a10
11 a11
12 a12
13 a13
14 a14
15 a15
16 a16
17 a17
18 a18
19 a19
Remove
Selected attribute
Name: a01
Missing: 0 (0%)
Distinct: 2
Type: Numeric
Unique: 0 (0%)
Statistic
Value
Minimum 0
Maximum 1
Mean 0.892
StdDev 0.311
Class: class (Nom)
Visualize All
313
38
0
0
0
0
0
0.5
1
Status
OK
Log
x 0


Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
weka
classifiers
bayes
functions
GaussianProcesses
LinearRegression
Logistic
MultilayerPerceptron
SGD
SGDText
SimpleLinearRegression
SimpleLogistic
SMO
SMOreg
VotedPerceptron
lazy
meta
misc
rules
DecisionTable
JRip
M5Rules
OneR
PART
ZeroR
trees
Close

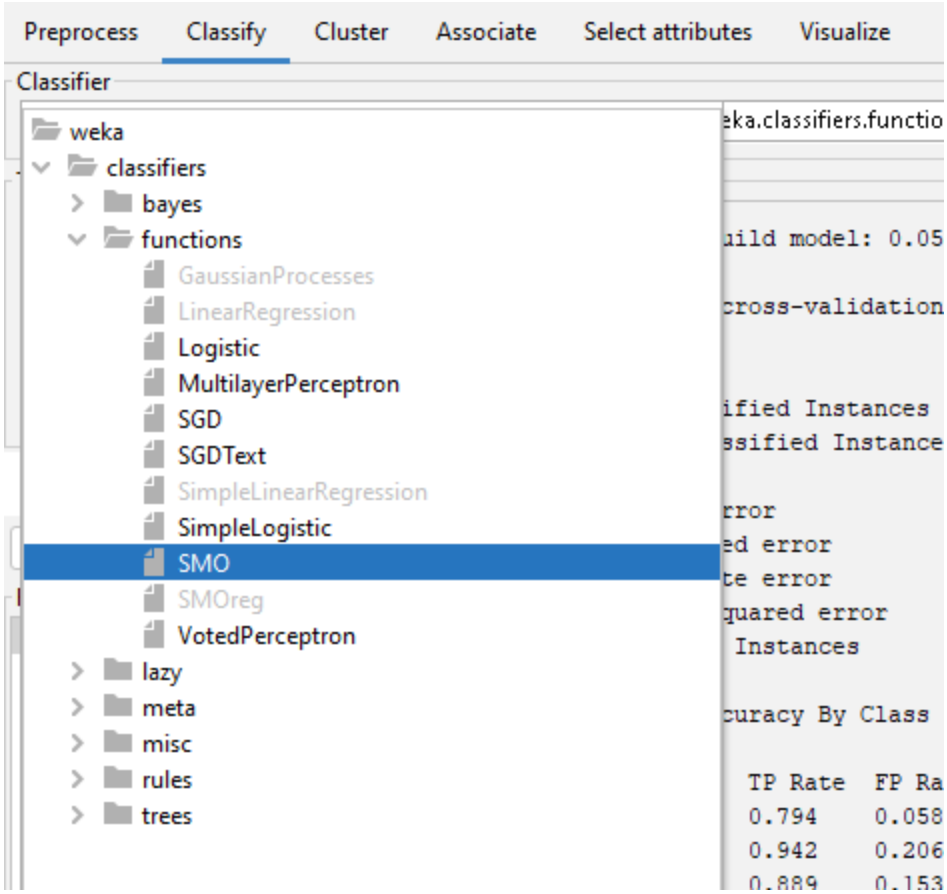

Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
Logistic -R 1.0E-8 -M -1 -num-decimal-places 4
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
10:51:16 - functions.Logistic
Classifier output
Time taken to build model: 0.05 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 312 88.8889 %
Incorrectly Classified Instances 39 11.1111 %
Kappa statistic 0.753
Mean absolute error 0.1283
Root mean squared error 0.3035
Relative absolute error 27.8593 %
Root relative squared error 63.2601 %
Total Number of Instances 351
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC
0.794 0.058 0.885 0.794 0.837 0.756 0.870 0.89
0.942 0.206 0.891 0.942 0.916 0.756 0.870 0.83
Weighted Avg. 0.889 0.153 0.889 0.889 0.887 0.756 0.870 0.85
=== Confusion Matrix ===
a b <-- classified as
100 26 | a = b
13 212 | b = g

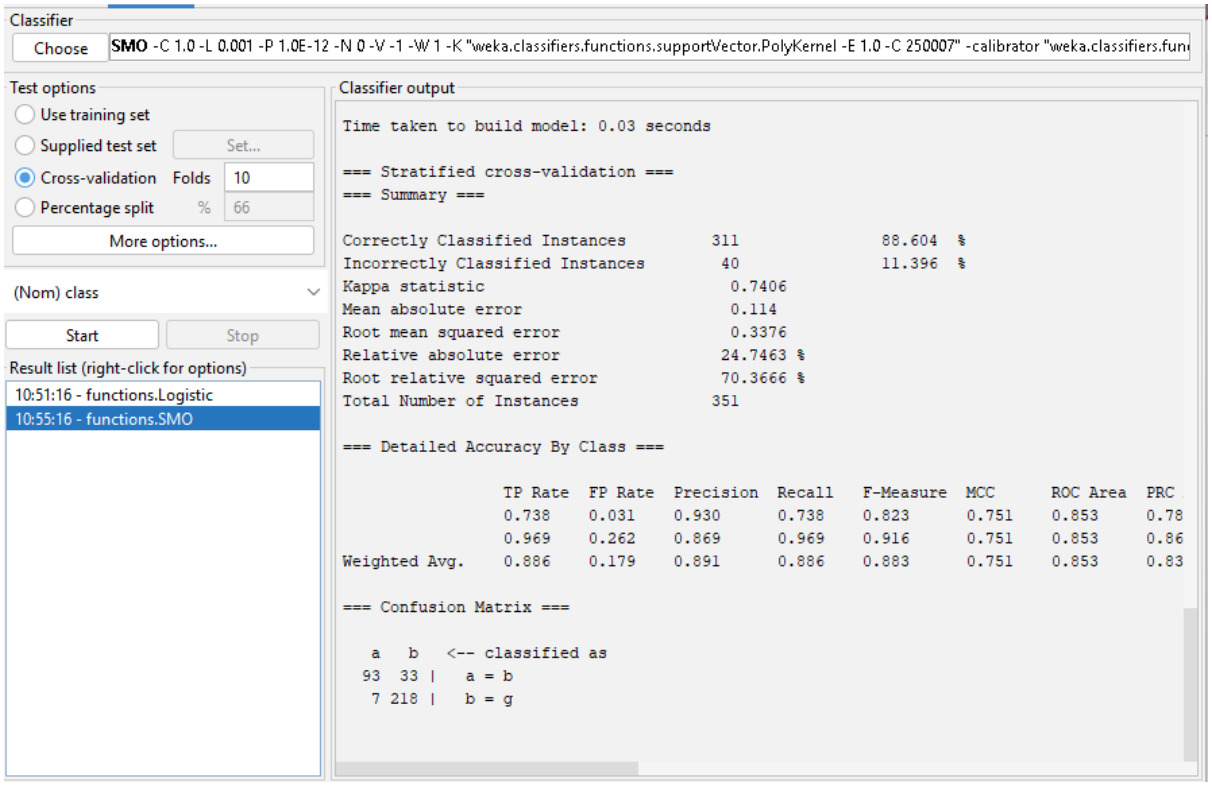

Result list (right-click for options)
10:51:16 - functions.L
View in main window
View in separate window
Save result buffer
Delete result buffer(s)
Load model
Save model
Re-evaluate model on current test set
Re-apply this model's configuration
Visualize classifier errors
Visualize tree
Visualize margin curve
Visualize threshold curve
Cost/Benefit analysis
Visualize cost curve
Status
OK
Relative absolute erro
Root relative squared
Instar
uracy
TP Ra
0.794
0.942
0.889
trix =
lassif
b
g



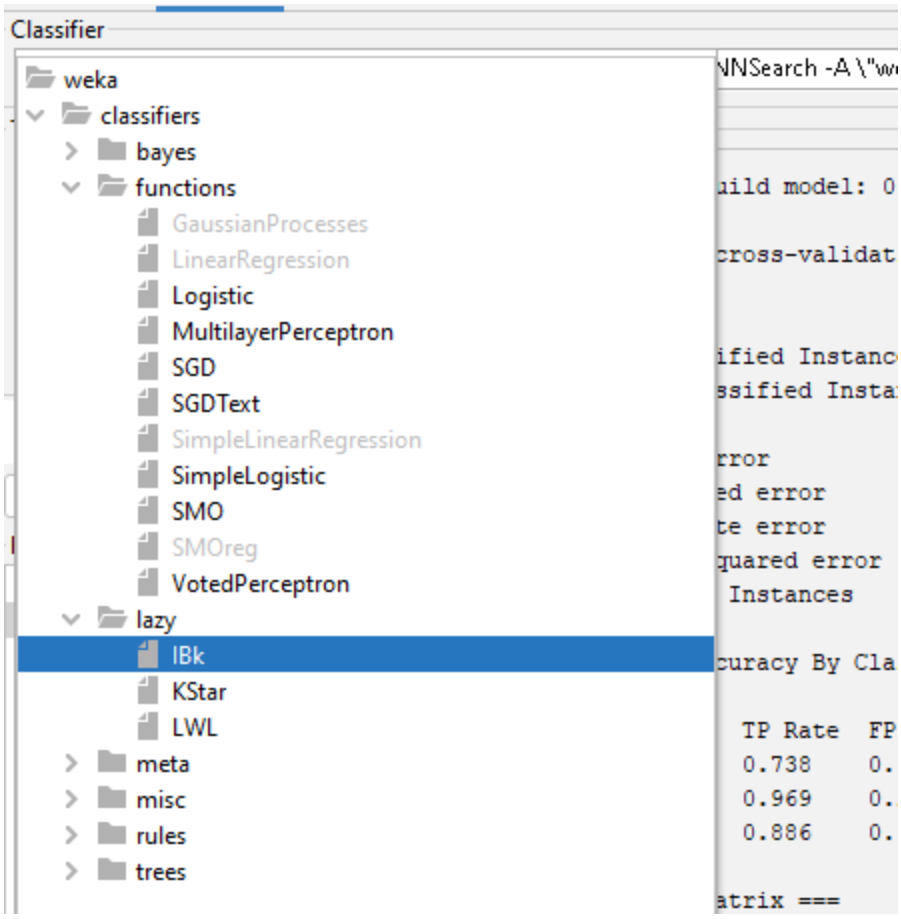
3. **KNN**

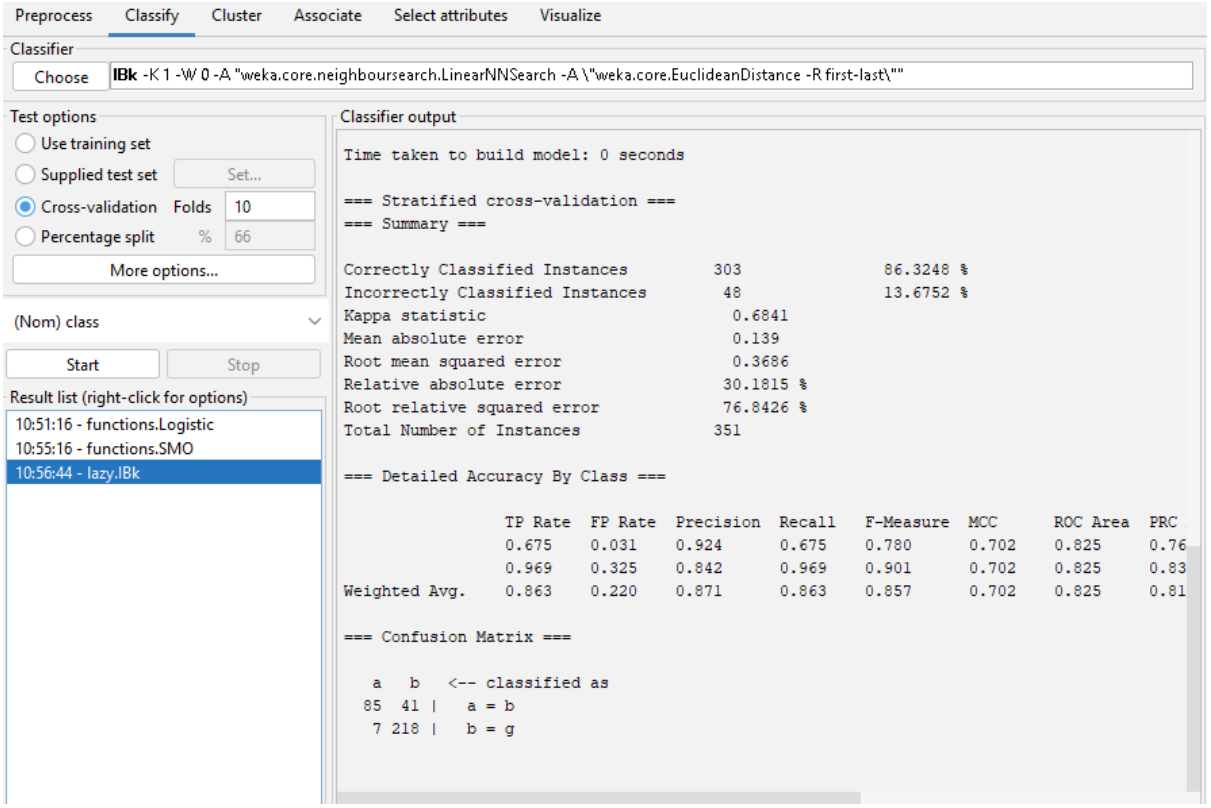
Use the same dataset as used for Logistic Regression
Click on Choose and Select SMO and click on start

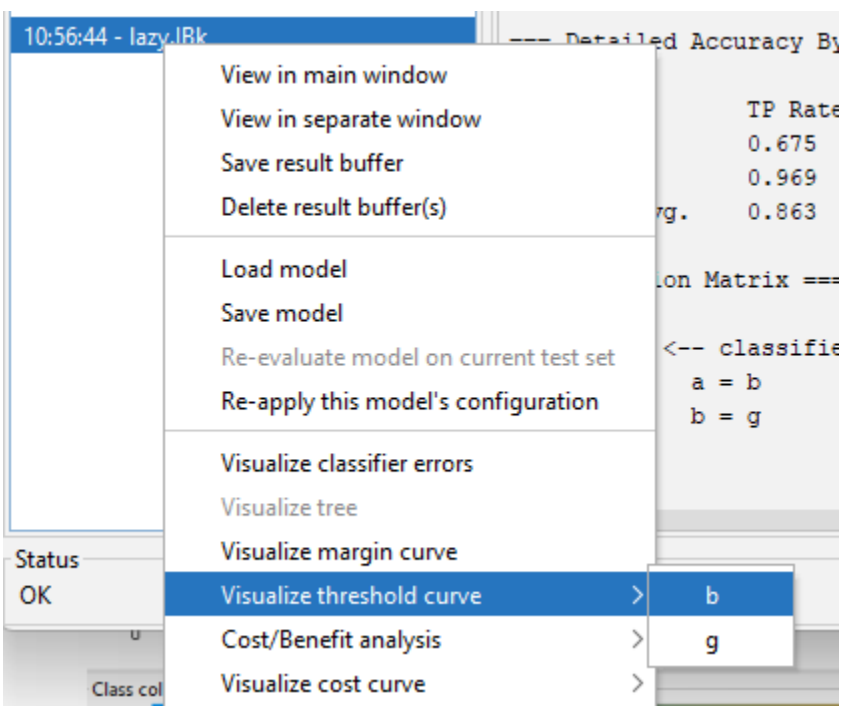

Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
weka
classifiers
bayes
functions
GaussianProcesses
LinearRegression
Logistic
MultilayerPerceptron
SGD
SGDText
SimpleLinearRegression
SimpleLogistic
SMO
SMOreg
VotedPerceptron
lazy
meta
misc
rules
trees
eka.classifiers.functio
uild model: 0.05
cross-validation
ified Instances
ssified Instance
rror
ed error
te error
quared error
Instances
curacy By Class
TP Rate FP Ra
0.794 0.058
0.942 0.206
0.889 0.153

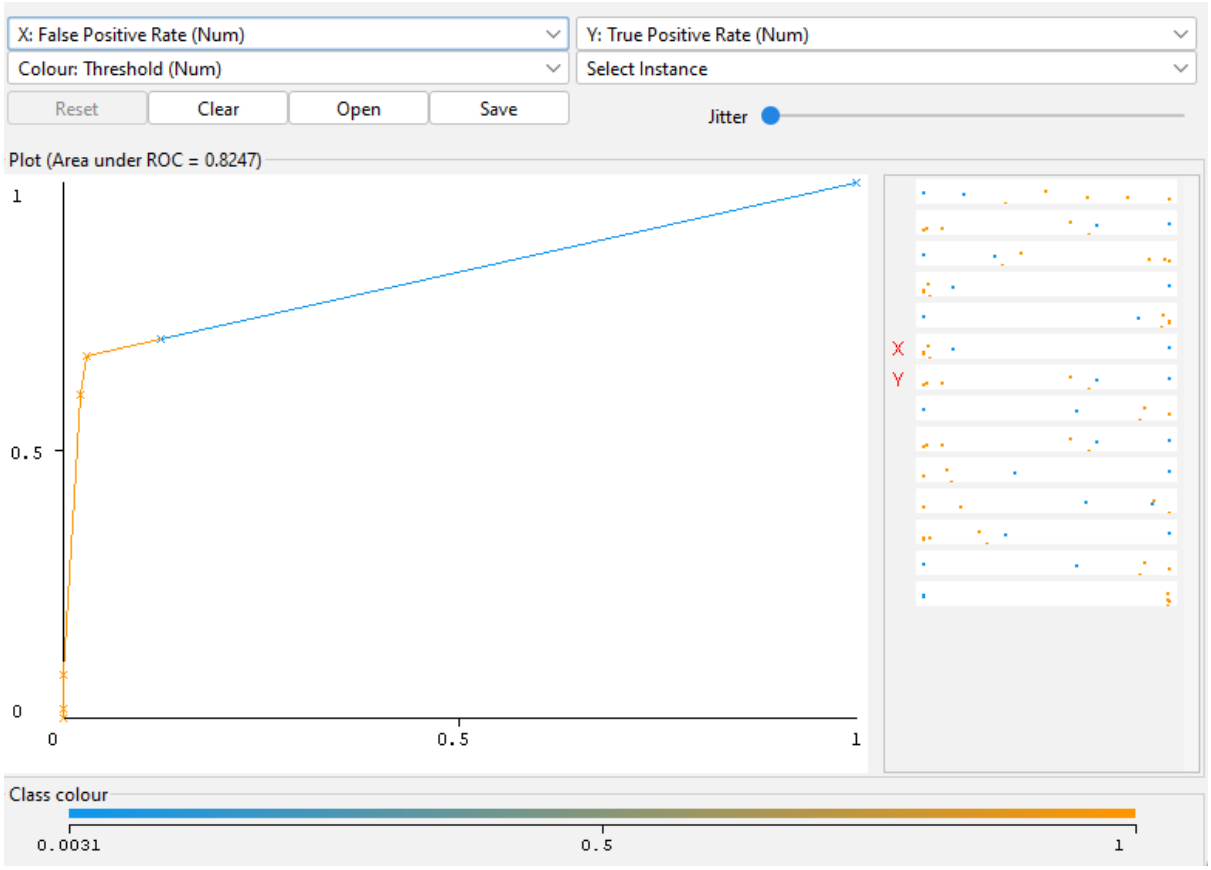

Classifier
Choose
SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.fun
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
10:51:16 - functions.Logistic
10:55:16 - functions.SMO
Classifier output
Time taken to build model: 0.03 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 311 88.604 %
Incorrectly Classified Instances 40 11.396 %
Kappa statistic 0.7406
Mean absolute error 0.114
Root mean squared error 0.3376
Relative absolute error 24.7463 %
Root relative squared error 70.3666 %
Total Number of Instances 351
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC
0.738 0.031 0.930 0.738 0.823 0.751 0.853 0.78
0.969 0.262 0.869 0.969 0.916 0.751 0.853 0.86
Weighted Avg. 0.886 0.179 0.891 0.886 0.883 0.751 0.853 0.83
=== Confusion Matrix ===
a b <-- classified as
93 33 | a = b
7 218 | b = g

4. IBK

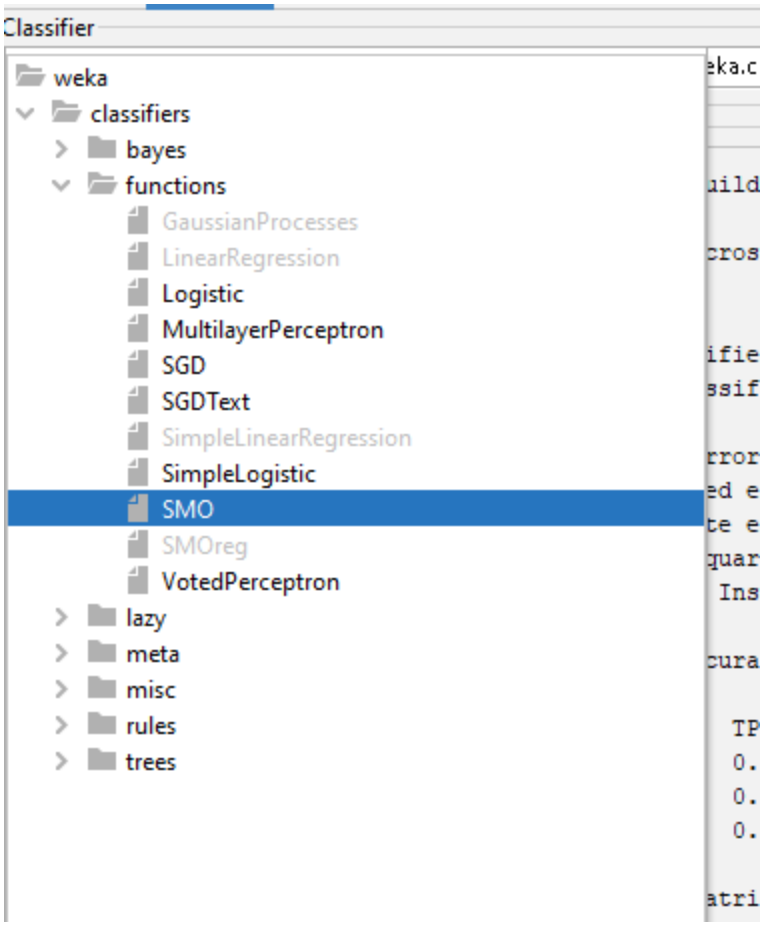

Classifier
weka
classifiers
bayes
functions
GaussianProcesses
LinearRegression
Logistic
MultilayerPerceptron
SGD
SGDText
SimpleLinearRegression
SimpleLogistic
SMO
SMOreg
VotedPerceptron
lazy
IBk
KStar
LWL
meta
misc
rules
trees
NNSearch -A \"w
uild model: 0
cross-validat
ified Instanc
ssified Insta
rror
ed error
te error
quared error
Instances
curacy By Cla
TP Rate FP
0.738 0.
0.969 0.
0.886 0.
atrix ===

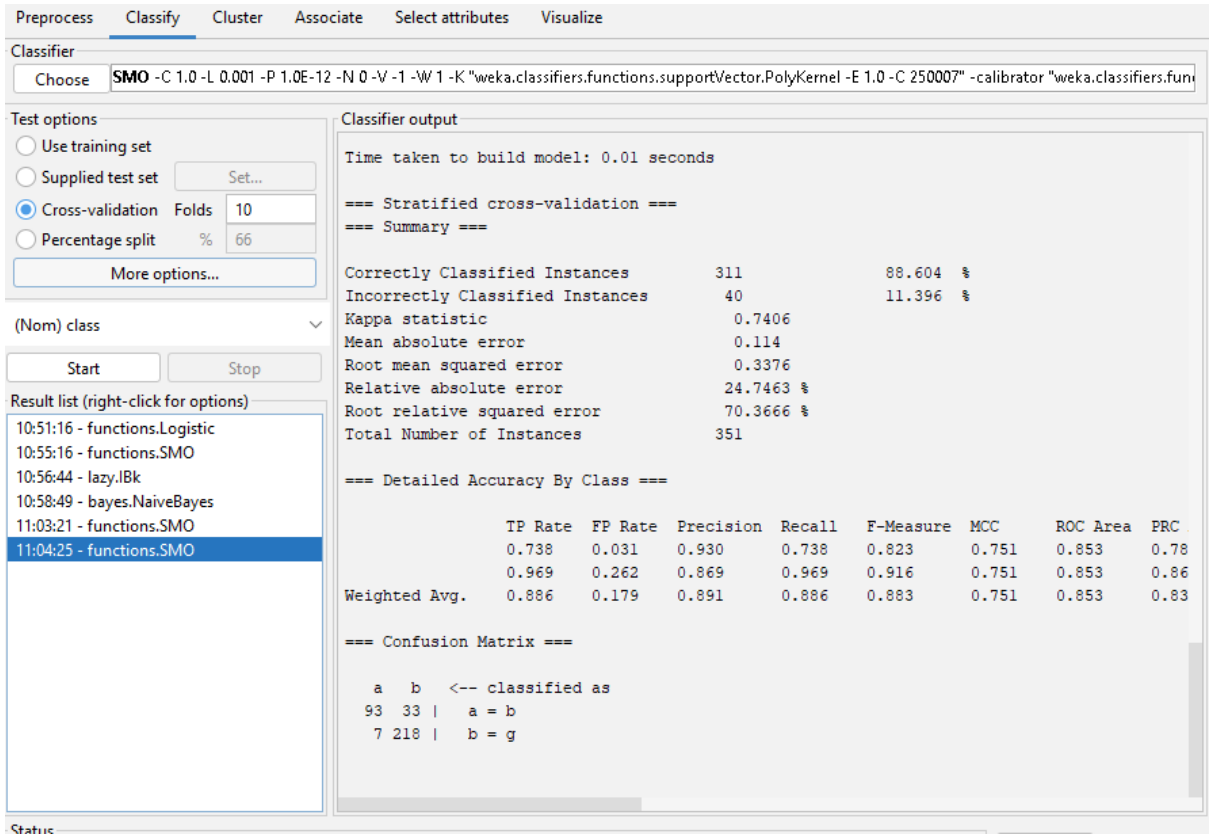

Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
10:51:16 - functions.Logistic
10:55:16 - functions.SMO
10:56:44 - lazy.IBk
Classifier output
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 303 86.3248 %
Incorrectly Classified Instances 48 13.6752 %
Kappa statistic 0.6841
Mean absolute error 0.139
Root mean squared error 0.3686
Relative absolute error 30.1815 %
Root relative squared error 76.8426 %
Total Number of Instances 351
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC
0.675 0.031 0.924 0.675 0.780 0.702 0.825 0.76
0.969 0.325 0.842 0.969 0.901 0.702 0.825 0.83
Weighted Avg. 0.863 0.220 0.871 0.863 0.857 0.702 0.825 0.81
=== Confusion Matrix ===
a b <-- classified as
85 41 | a = b
7 218 | b = g


10:56:44 - lazy.IBk
View in main window
View in separate window
Save result buffer
Delete result buffer(s)
Load model
Save model
Re-evaluate model on current test set
Re-apply this model's configuration
Visualize classifier errors
Visualize tree
Visualize margin curve
Visualize threshold curve
Cost/Benefit analysis
Visualize cost curve
b
g
Status
OK

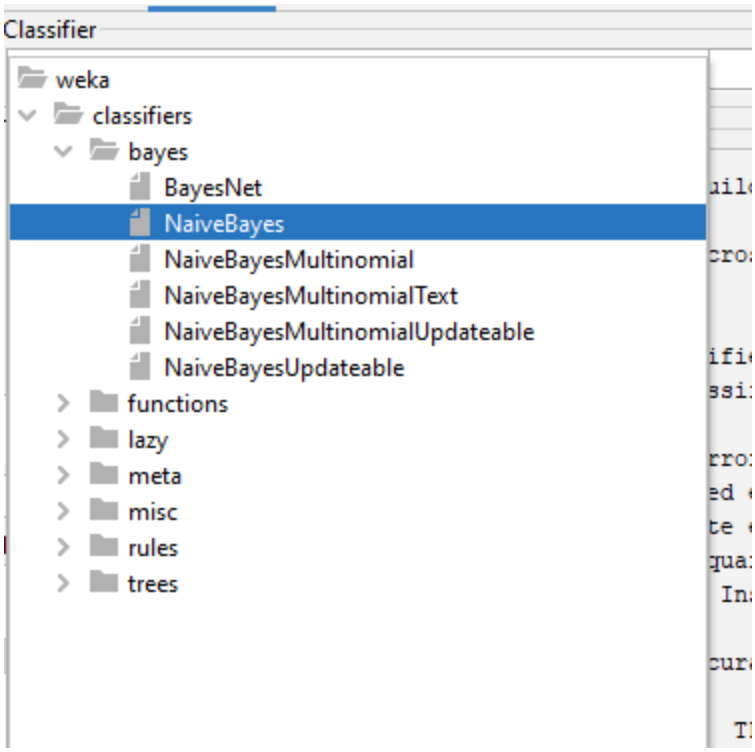

X: False Positive Rate (Num)
Y: True Positive Rate (Num)
Colour: Threshold (Num)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot (Area under ROC = 0.8247)
1
0.5
0
0
0.5
1
X
Y
Class colour
0.0031
0.5
1

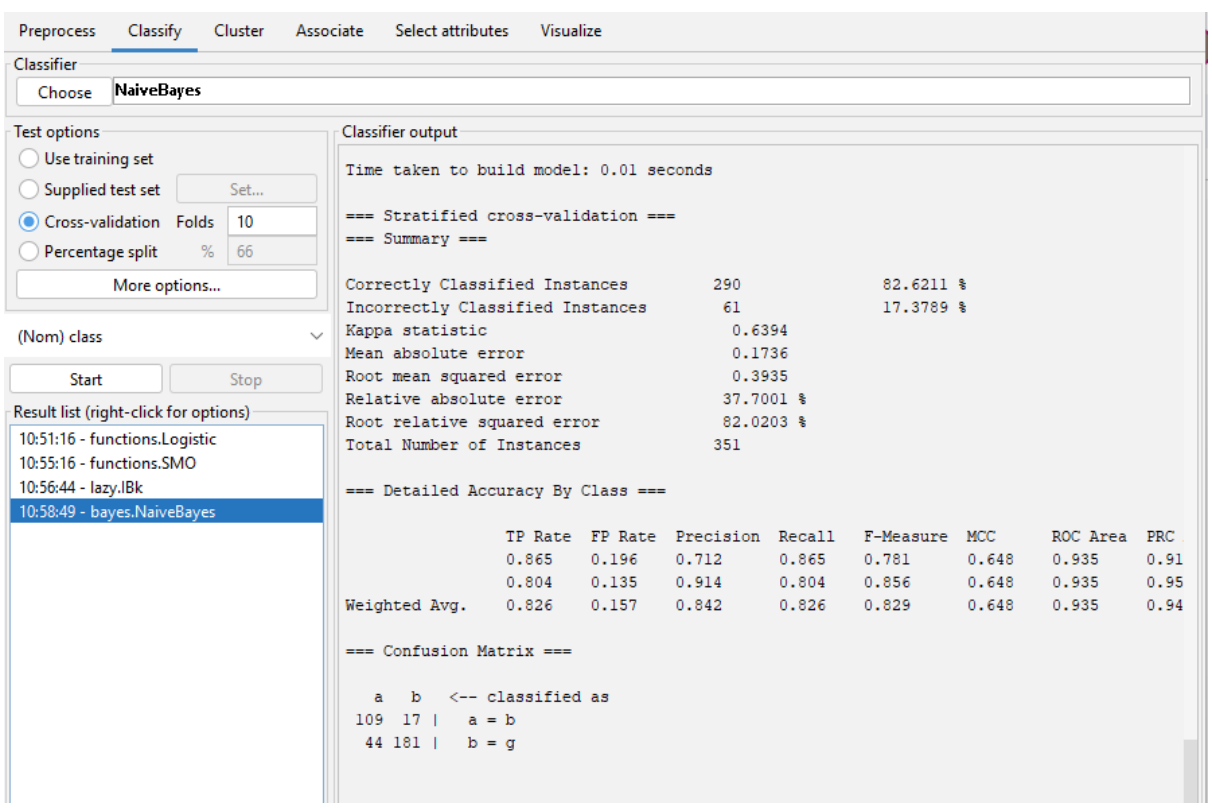
5. SMO

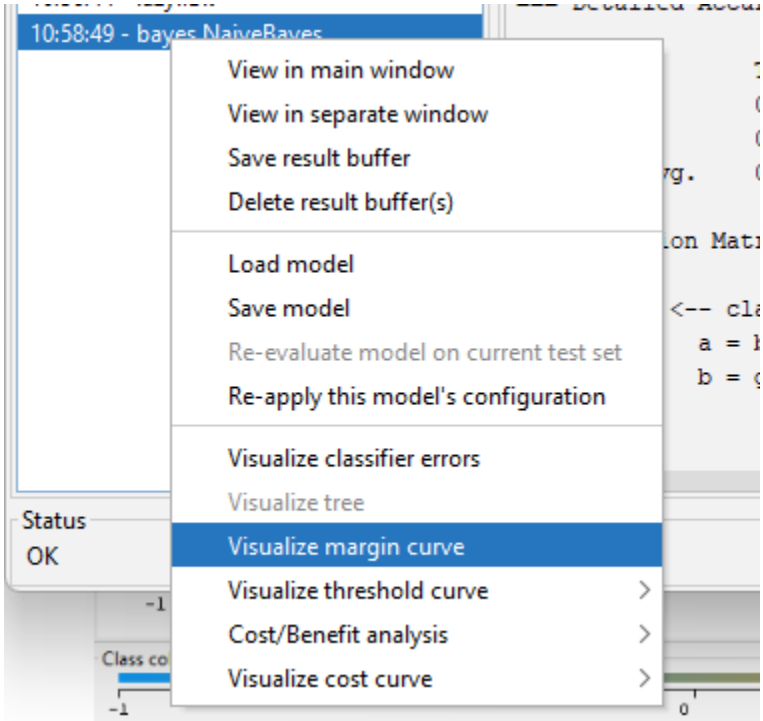

Classifier
weka
classifiers
bayes
functions
GaussianProcesses
LinearRegression
Logistic
MultilayerPerceptron
SGD
SGDText
SimpleLinearRegression
SimpleLogistic
SMO
SMOreg
VotedPerceptron
lazy
meta
misc
rules
trees

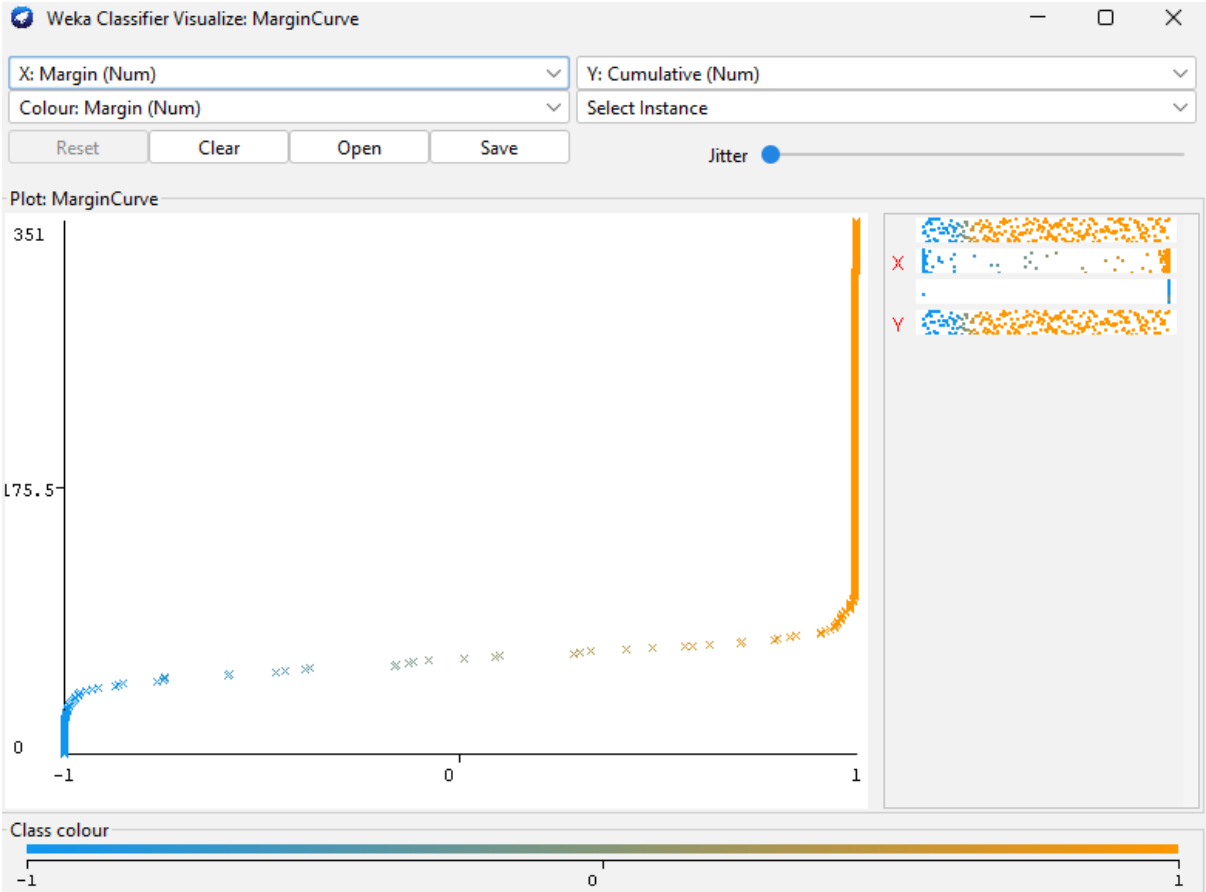


6. Naive Bayes


Classifier
weka
classifiers
bayes
BayesNet
NaiveBayes
NaiveBayesMultinomial
NaiveBayesMultinomialText
NaiveBayesMultinomialUpdateable
NaiveBayesUpdateable
functions
lazy
meta
misc
rules
trees

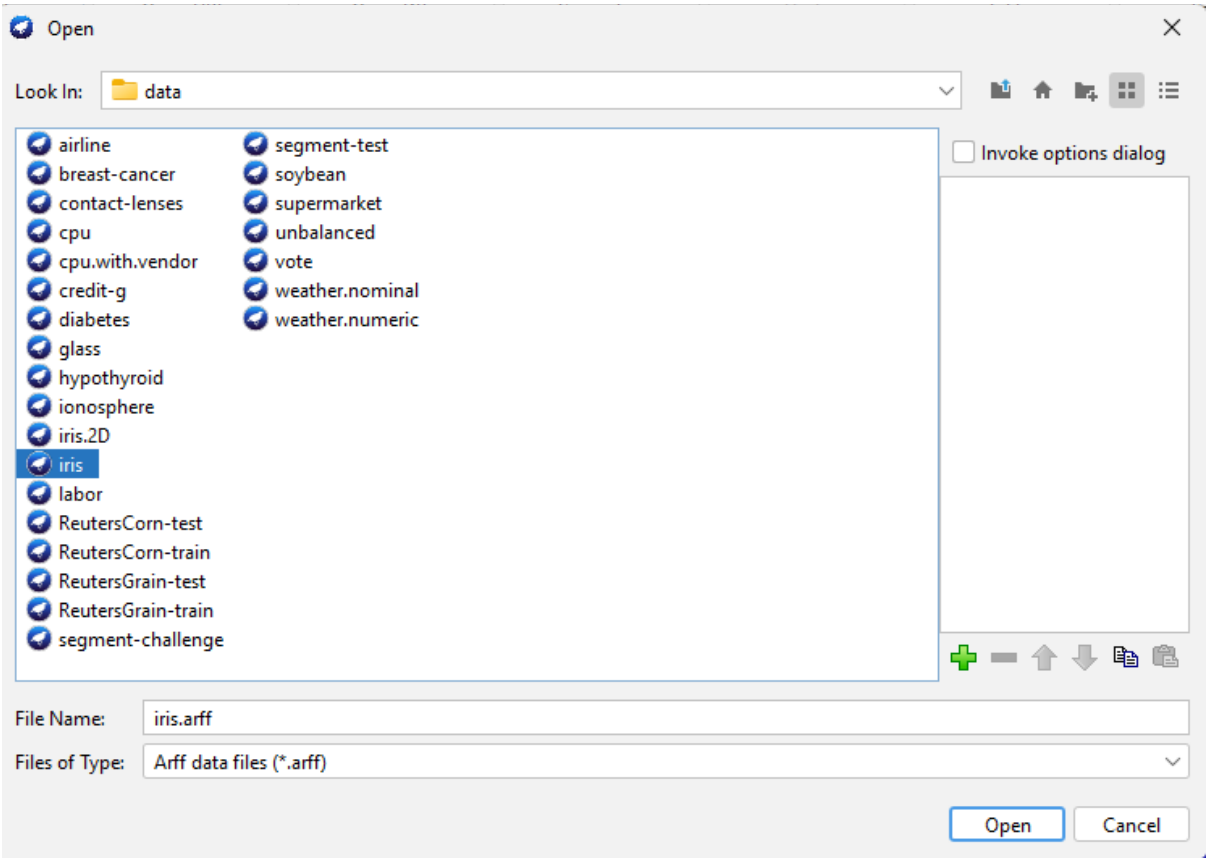

Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
10:51:16 - functions.Logistic
10:55:16 - functions.SMO
10:56:44 - lazy.IBk
10:58:49 - bayes.NaiveBayes
Classifier output
Time taken to build model: 0.01 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 290 82.6211 %
Incorrectly Classified Instances 61 17.3789 %
Kappa statistic 0.6394
Mean absolute error 0.1736
Root mean squared error 0.3935
Relative absolute error 37.7001 %
Root relative squared error 82.0203 %
Total Number of Instances 351
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC
0.865 0.196 0.712 0.865 0.781 0.648 0.935 0.91
0.804 0.135 0.914 0.804 0.856 0.648 0.935 0.95
Weighted Avg. 0.826 0.157 0.842 0.826 0.829 0.648 0.935 0.94
=== Confusion Matrix ===
a b <-- classified as
109 17 | a = b
44 181 | b = g

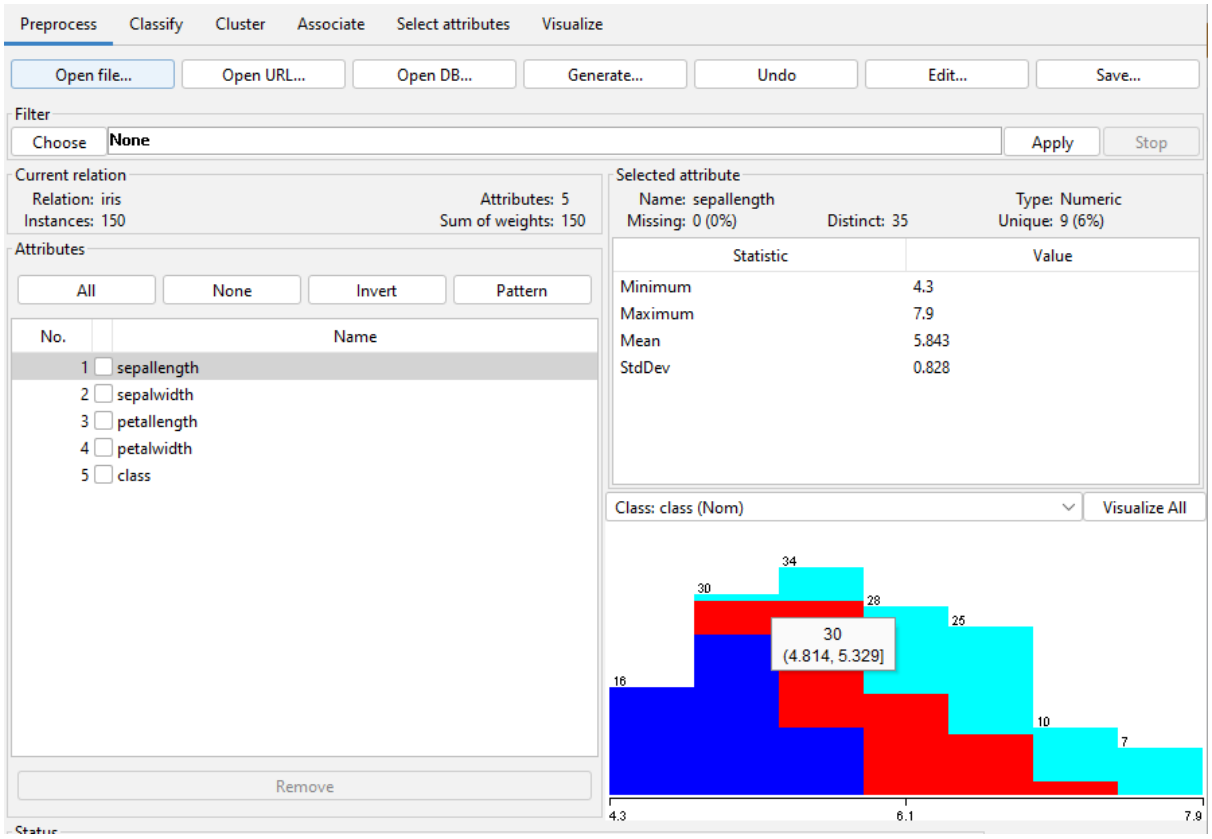

10:58:49 - bayes.NaiveBayes
View in main window
View in separate window
Save result buffer
Delete result buffer(s)
Load model
Save model
Re-evaluate model on current test set
Re-apply this model's configuration
Visualize classifier errors
Visualize tree
Visualize margin curve
Visualize threshold curve
Cost/Benefit analysis
Visualize cost curve
Status
OK

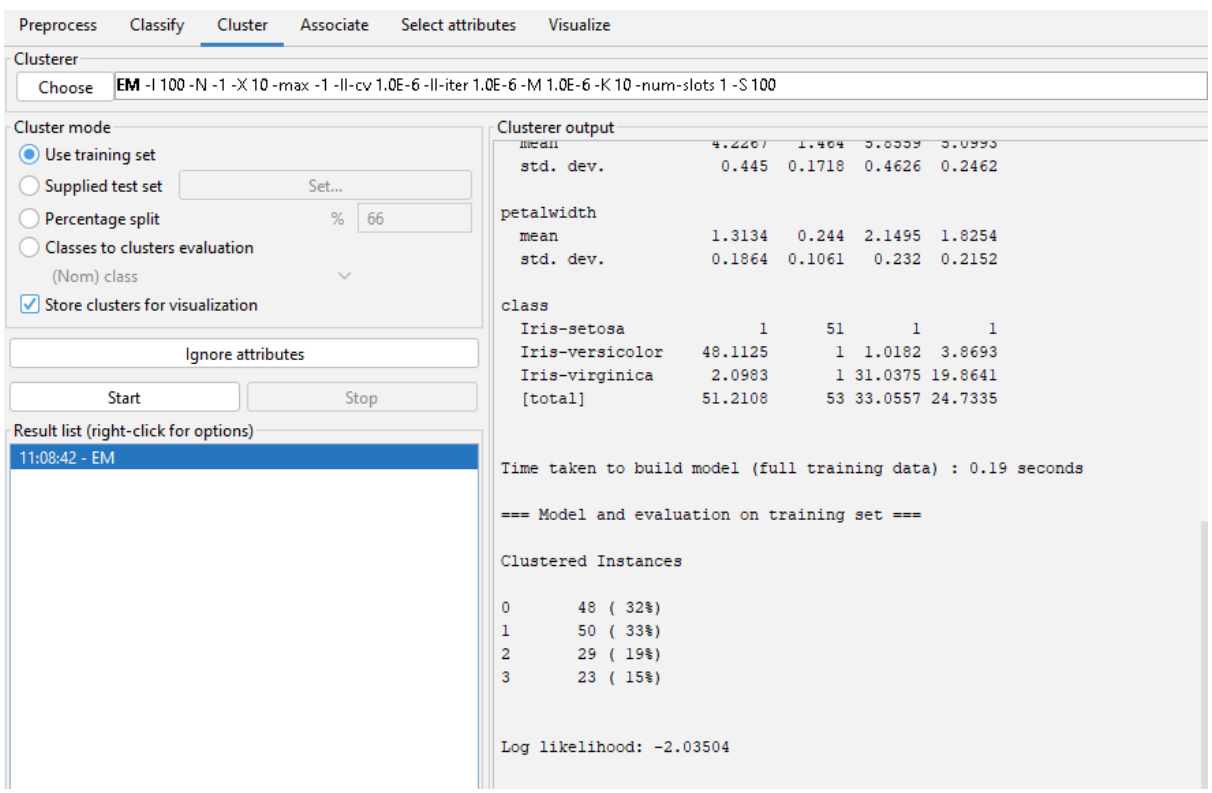


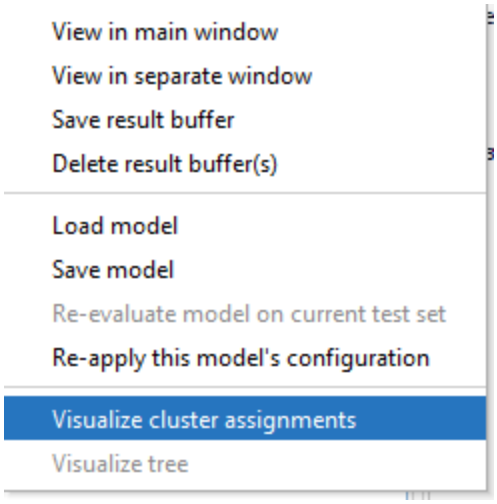
- **Clustering**

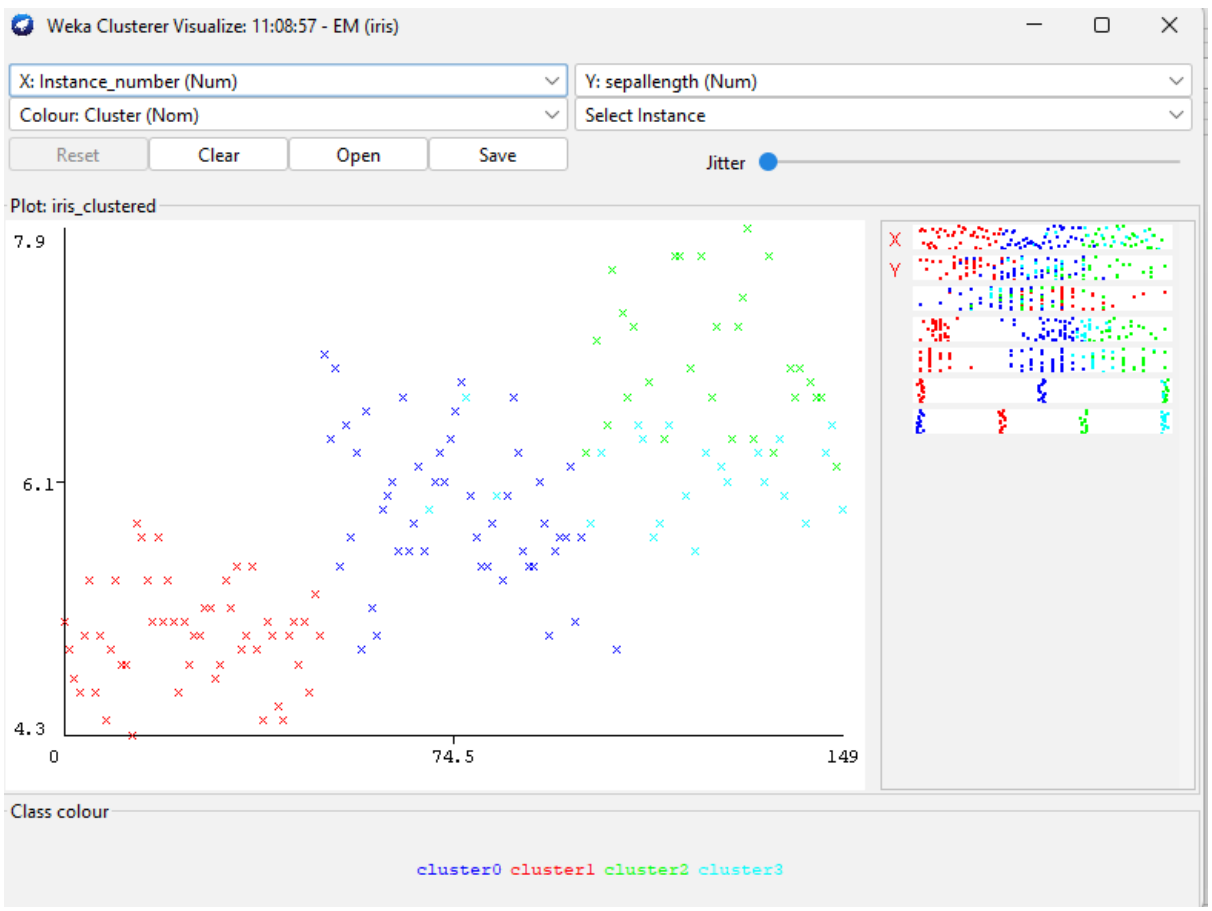
1. **EM**


Open
Look In:
data
airline
breast-cancer
contact-lenses
cpu
cpu.with.vendor
credit-g
diabetes
glass
hypothyroid
ionosphere
iris.2D
iris
labor
ReutersCorn-test
ReutersCorn-train
ReutersGrain-test
ReutersGrain-train
segment-challenge
segment-test
soybean
supermarket
unbalanced
vote
weather.nominal
weather.numeric
Invoke options dialog
File Name:
iris.arff
Files of Type:
Arff data files (*.arff)
Open
Cancel

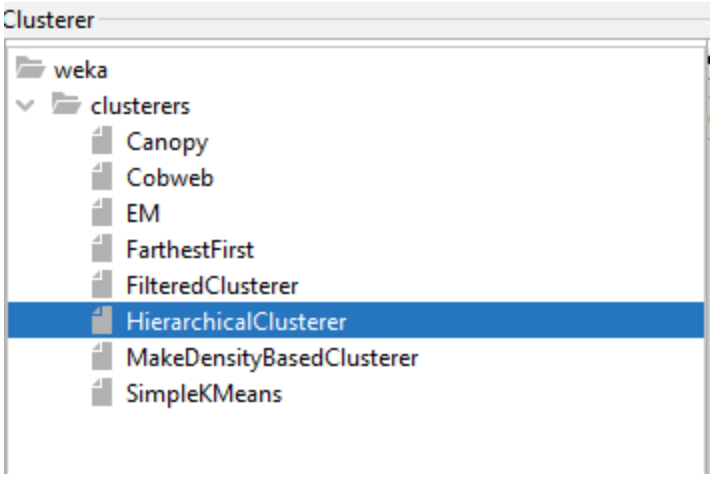

Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: iris
Instances: 150
Attributes: 5
Sum of weights: 150
Selected attribute
Name: sepallength
Missing: 0 (0%)
Distinct: 35
Type: Numeric
Unique: 9 (6%)
Statistic
Value
Minimum 4.3
Maximum 7.9
Mean 5.843
StdDev 0.828
Attributes
All
None
Invert
Pattern
No.
Name
1 sepallength
2 sepalwidth
3 petallength
4 petalwidth
5 class
Remove
Class: class (Nom)
Visualize All
34
30
28
25
30
(4.814, 5.329]
16
10
7
4.3
6.1
7.9

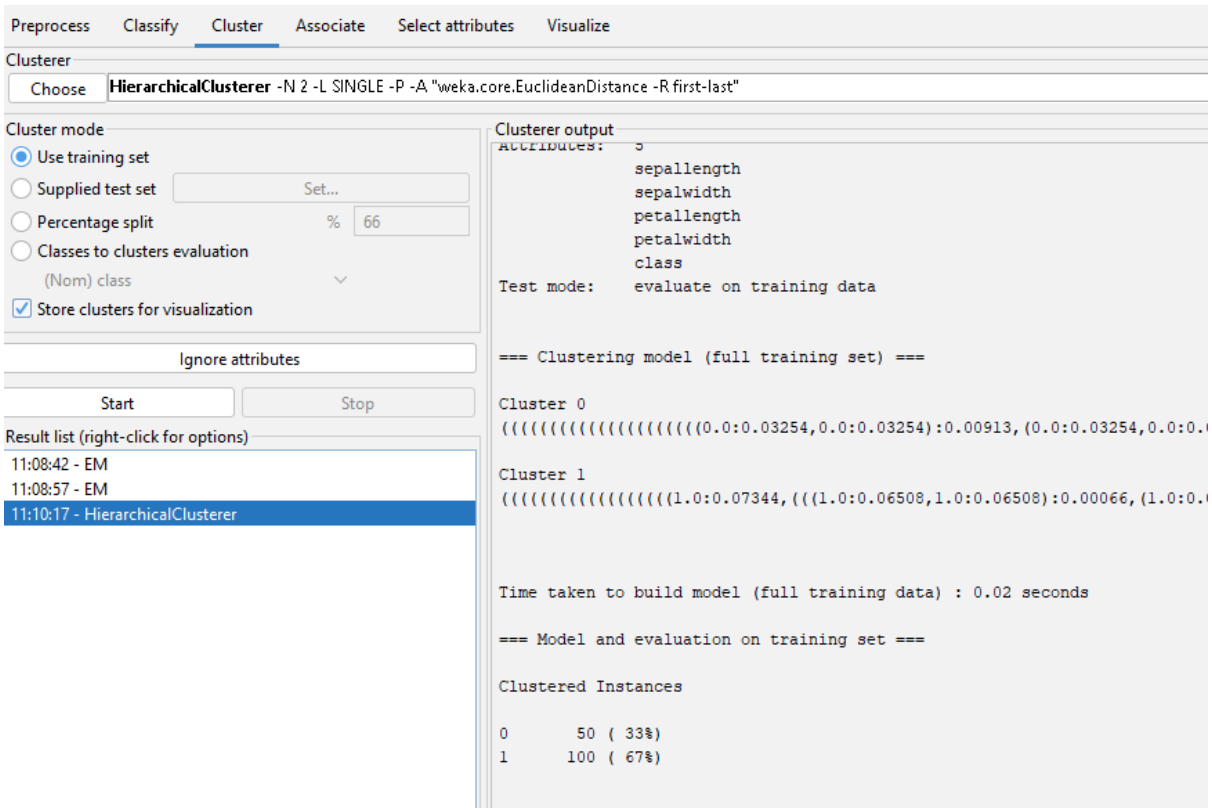

Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
EM -I 100 -N -1 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -num-slots 1 -S 100
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
11:08:42 - EM
Clusterer output
std. dev. 0.445 0.1718 0.4626 0.2462
petalwidth
mean 1.3134 0.244 2.1495 1.8254
std. dev. 0.1864 0.1061 0.232 0.2152
class
Iris-setosa 1 51 1 1
Iris-versicolor 48.1125 1 1.0182 3.8693
Iris-virginica 2.0983 1 31.0375 19.8641
[total] 51.2108 53 33.0557 24.7335
Time taken to build model (full training data) : 0.19 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 48 (32%)
1 50 (33%)
2 29 (19%)
3 23 (15%)
Log likelihood: -2.03504

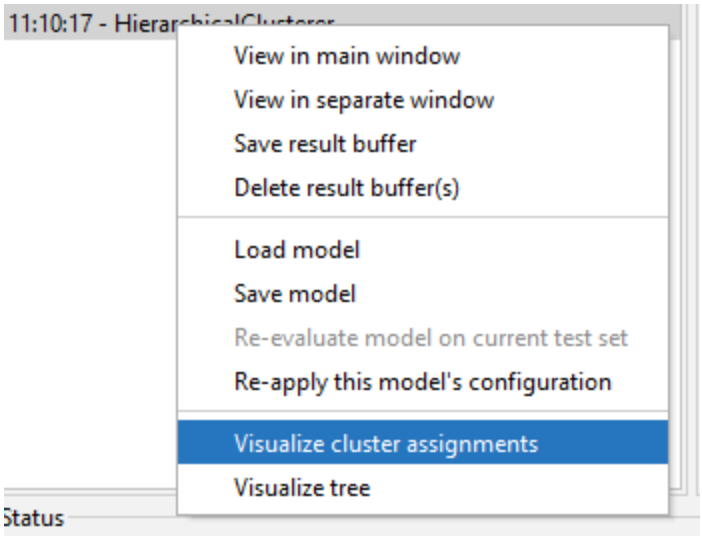


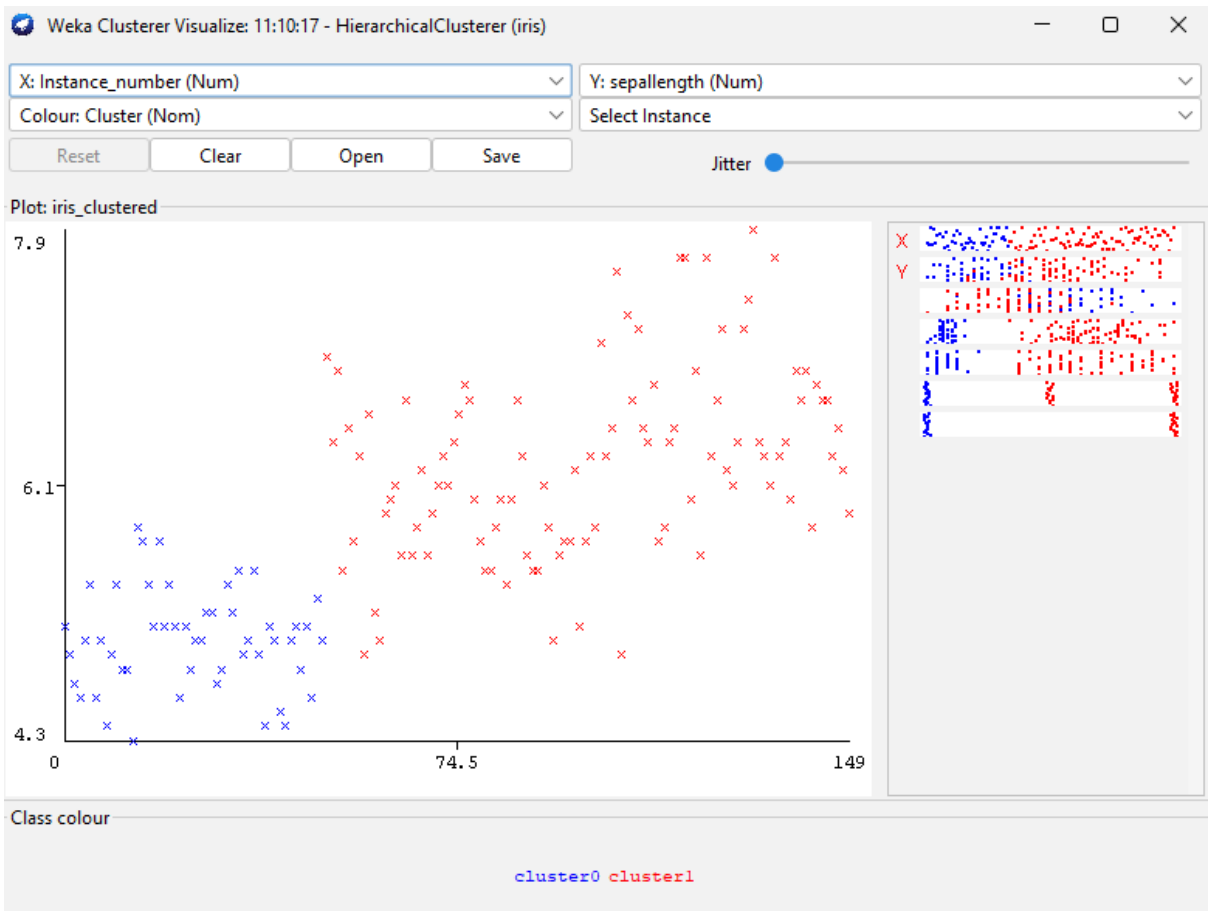


2. Hierarchical

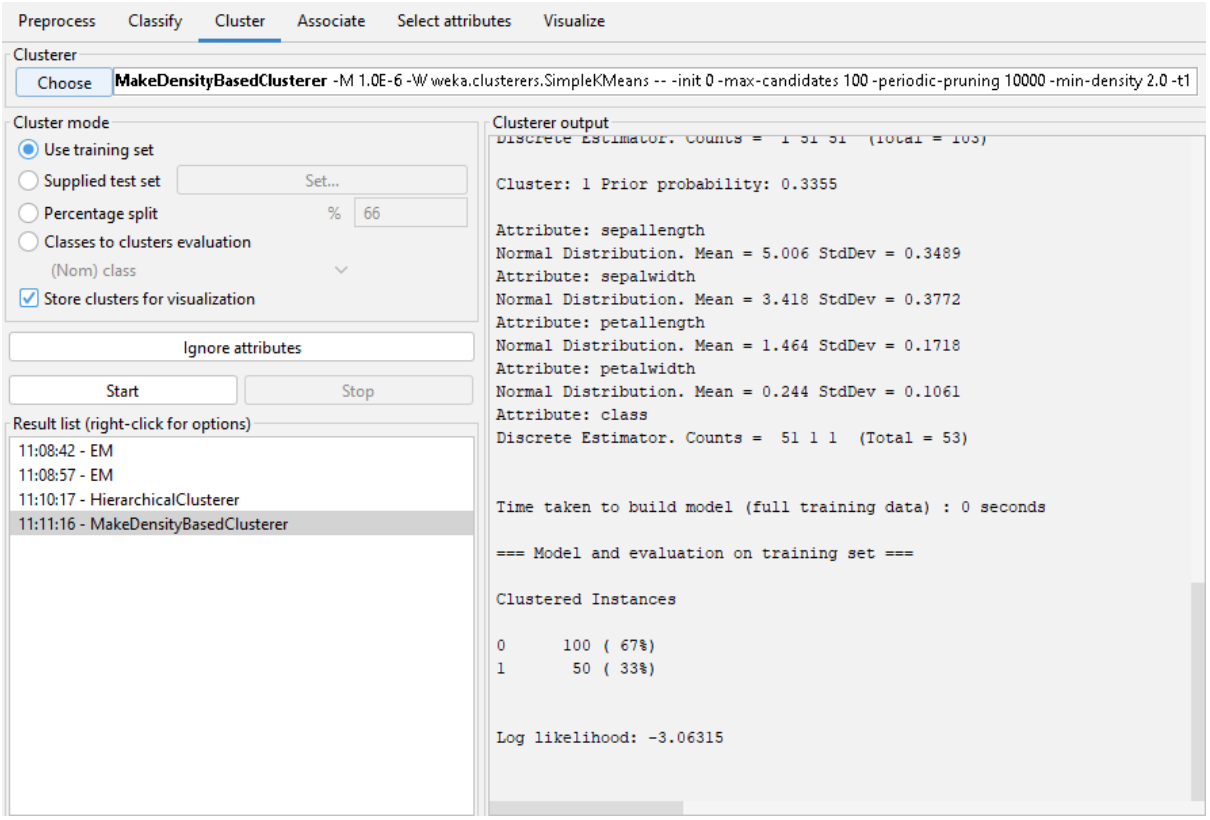

Clusterer
weka
clusterers
Canopy
Cobweb
EM
FarthestFirst
FilteredClusterer
HierarchicalClusterer
MakeDensityBasedClusterer
SimpleKMeans

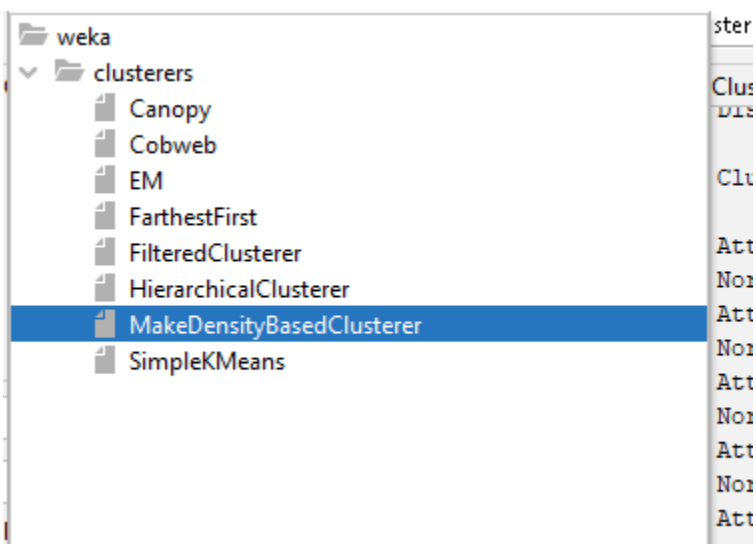

Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
11:08:42 - EM
11:08:57 - EM
11:10:17 - HierarchicalClusterer
Clusterer output
sepallength
sepalwidth
petallength
petalwidth
class
Test mode: evaluate on training data
=== Clustering model (full training set) ===
Cluster 0
Cluster 1
Time taken to build model (full training data) : 0.02 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 50 (33%)
1 100 (67%)

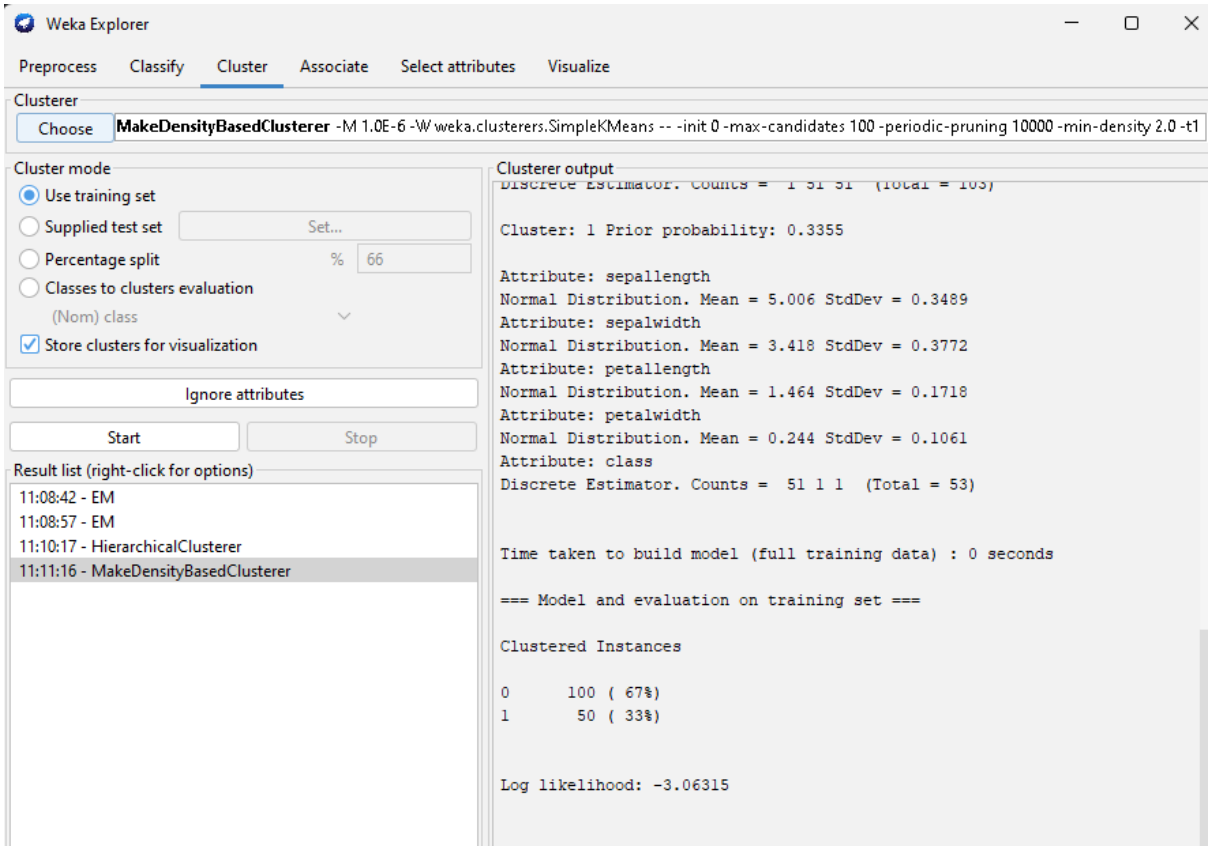


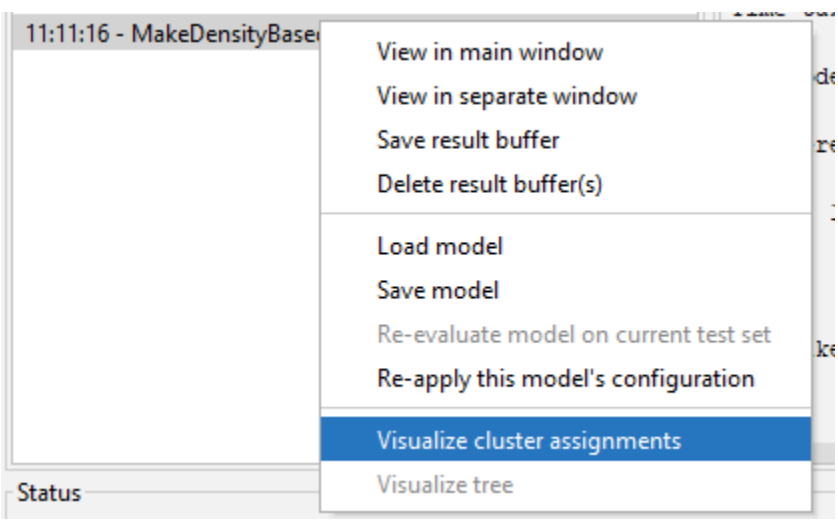


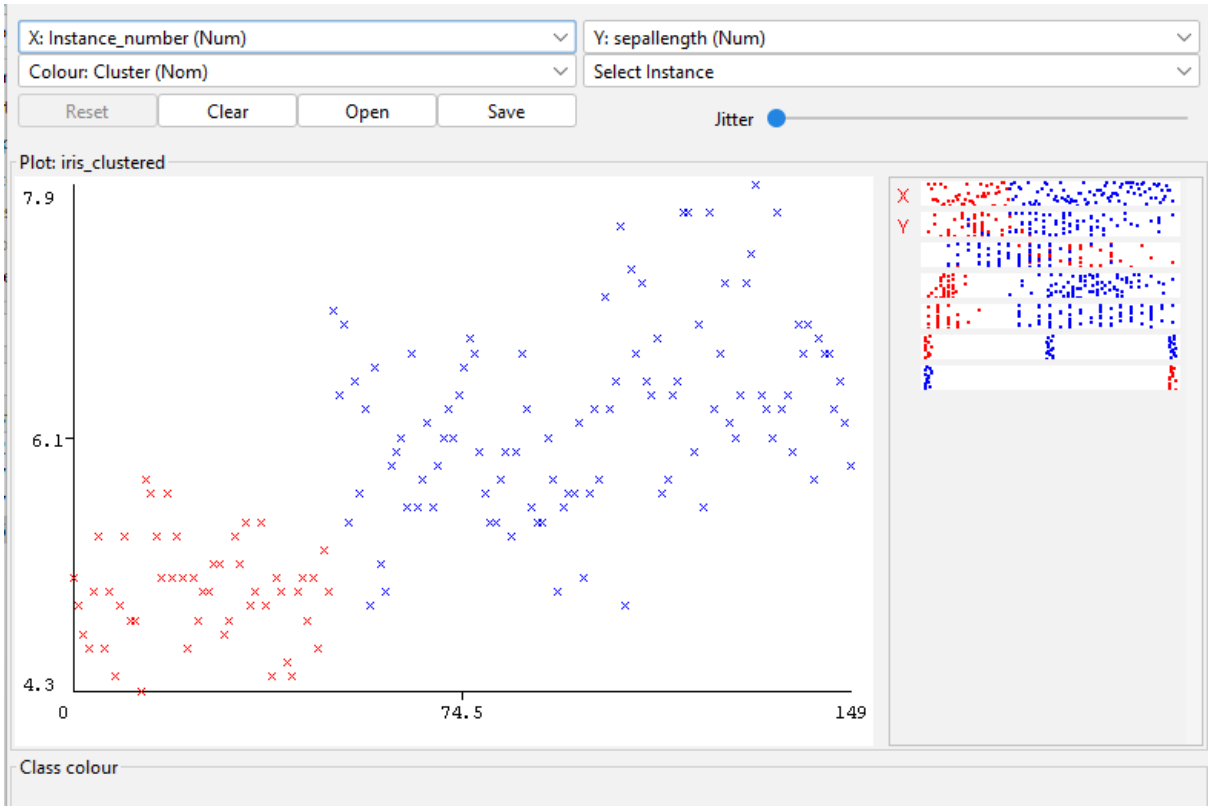
3. Make Density Based Cluster


Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
MakeDensityBasedClusterer -M 1.0E-6 -W weka.clusterers.SimpleKMeans -- -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
11:08:42 - EM
11:08:57 - EM
11:10:17 - HierarchicalClusterer
11:11:16 - MakeDensityBasedClusterer
Clusterer output
Cluster: 1 Prior probability: 0.3355
Attribute: sepallength
Normal Distribution. Mean = 5.006 StdDev = 0.3489
Attribute: sepalwidth
Normal Distribution. Mean = 3.418 StdDev = 0.3772
Attribute: petallength
Normal Distribution. Mean = 1.464 StdDev = 0.1718
Attribute: petalwidth
Normal Distribution. Mean = 0.244 StdDev = 0.1061
Attribute: class
Discrete Estimator. Counts = 51 1 1 (Total = 53)
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 100 (67%)
1 50 (33%)
Log likelihood: -3.06315

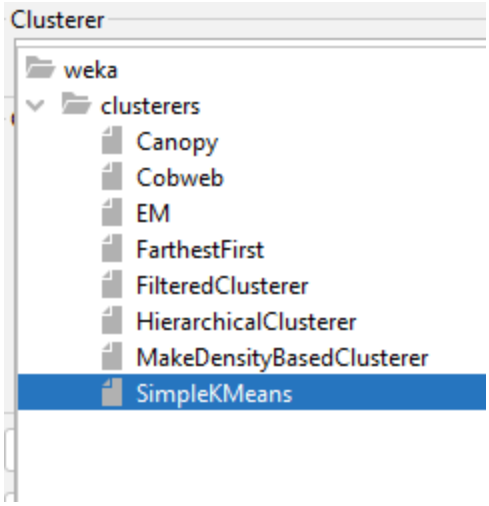

weka
clusterers
Canopy
Cobweb
EM
FarthestFirst
FilteredClusterer
HierarchicalClusterer
MakeDensityBasedClusterer
SimpleKMeans

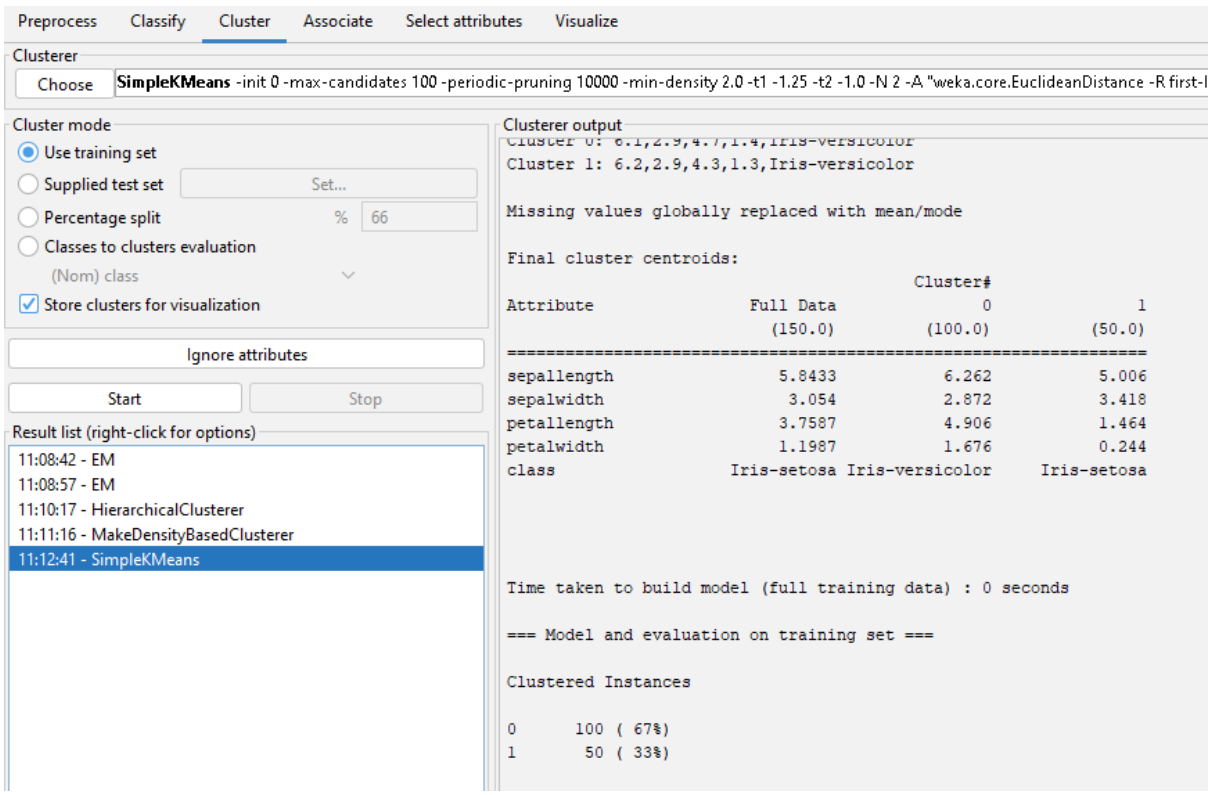

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
MakeDensityBasedClusterer -M 1.0E-6 -W weka.clusterers.SimpleKMeans -- -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
11:08:42 - EM
11:08:57 - EM
11:10:17 - HierarchicalClusterer
11:11:16 - MakeDensityBasedClusterer
Clusterer output
Cluster: 1 Prior probability: 0.3355
Attribute: sepallength
Normal Distribution. Mean = 5.006 StdDev = 0.3489
Attribute: sepalwidth
Normal Distribution. Mean = 3.418 StdDev = 0.3772
Attribute: petallength
Normal Distribution. Mean = 1.464 StdDev = 0.1718
Attribute: petalwidth
Normal Distribution. Mean = 0.244 StdDev = 0.1061
Attribute: class
Discrete Estimator. Counts = 51 1 1 (Total = 53)
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 100 (67%)
1 50 (33%)
Log likelihood: -3.06315

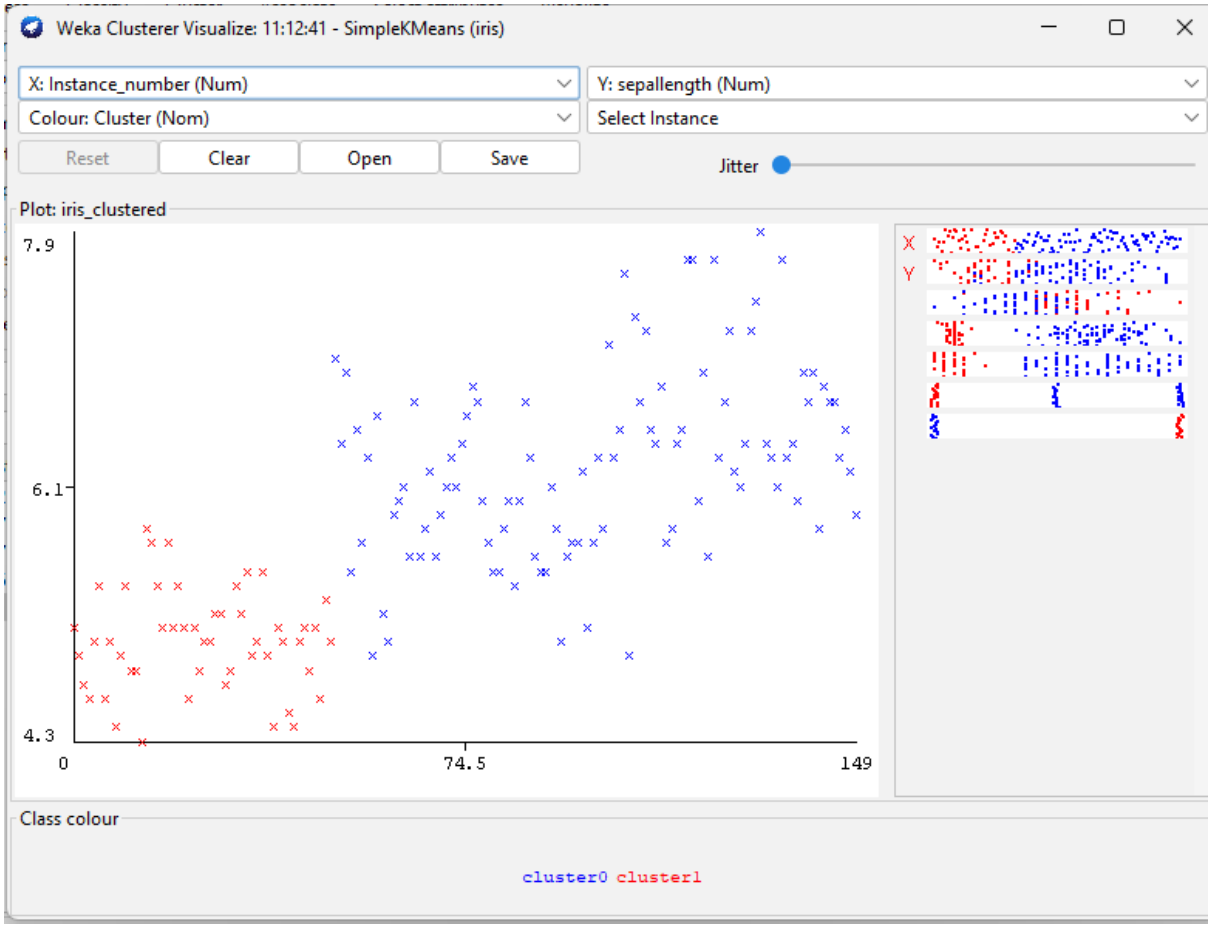

11:11:16 - MakeDensityBase
View in main window
View in separate window
Save result buffer
Delete result buffer(s)
Load model
Save model
Re-evaluate model on current test set
Re-apply this model's configuration
Visualize cluster assignments
Visualize tree
Status



4. Simple K Means

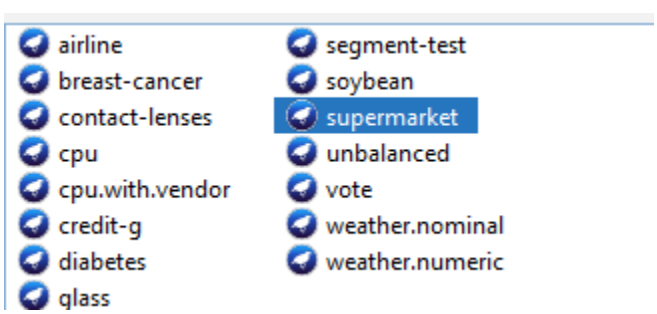

Clusterer
weka
clusterers
Canopy
Cobweb
EM
FarthestFirst
FilteredClusterer
HierarchicalClusterer
MakeDensityBasedClusterer
SimpleKMeans

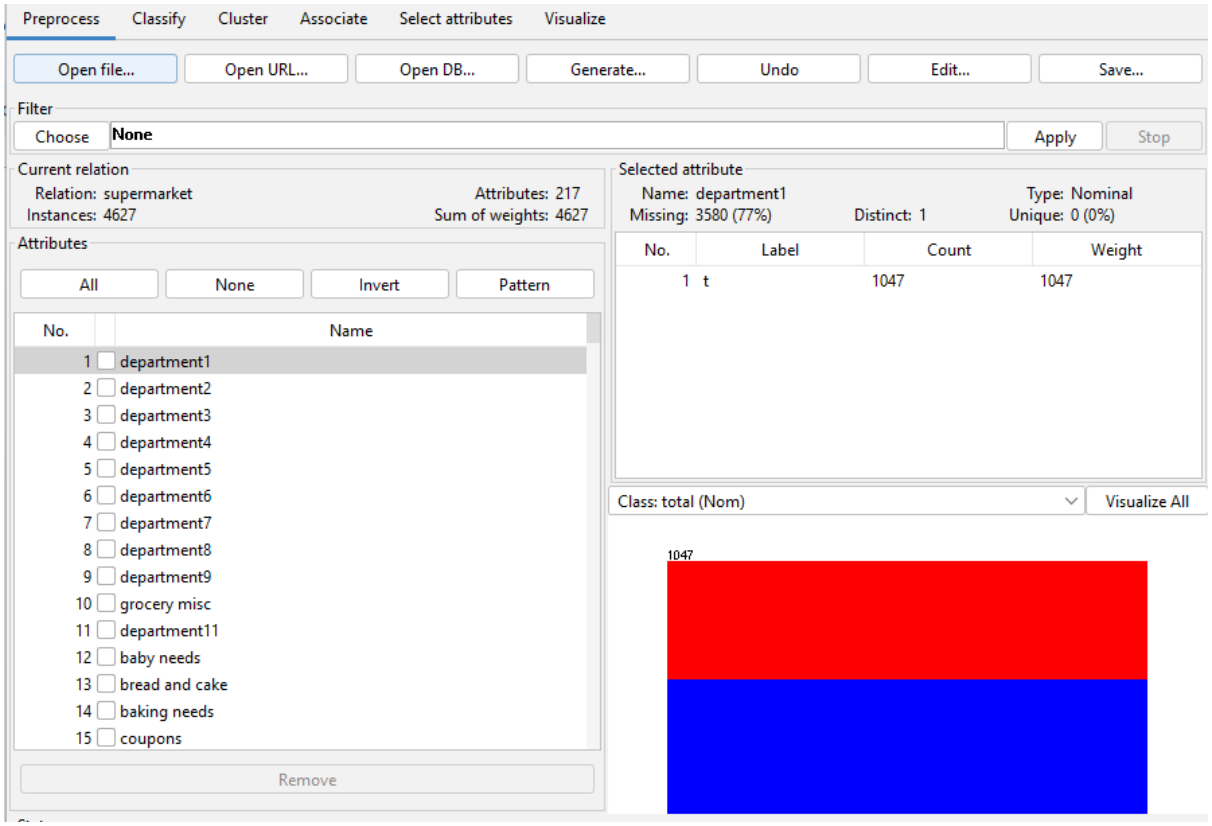

Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-l
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
11:08:42 - EM
11:08:57 - EM
11:10:17 - HierarchicalClusterer
11:11:16 - MakeDensityBasedClusterer
11:12:41 - SimpleKMeans
Clusterer output
Cluster 1: 6.2,2.9,4.3,1.3,Iris-versicolor
Missing values globally replaced with mean/mode
Final cluster centroids:
Cluster#
Attribute Full Data 0 1
(150.0) (100.0) (50.0)
sepallength 5.8433 6.262 5.006
sepalwidth 3.054 2.872 3.418
petallength 3.7587 4.906 1.464
petalwidth 1.1987 1.676 0.244
class Iris-setosa Iris-versicolor Iris-setosa
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 100 (67%)
1 50 (33%)

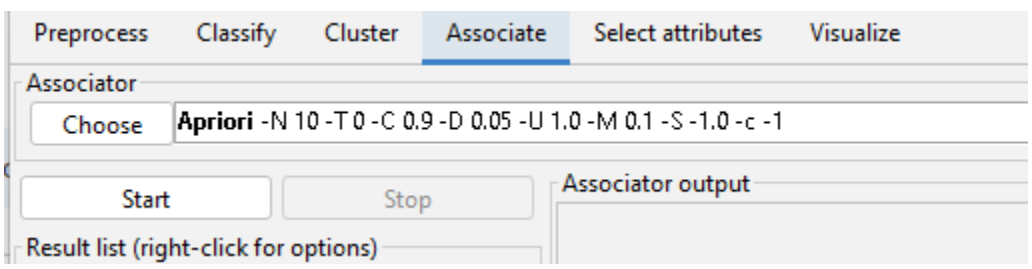


- **Association**

Apriori




Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: supermarket
Instances: 4627
Attributes: 217
Sum of weights: 4627
Selected attribute
Name: department1
Missing: 3580 (77%)
Distinct: 1
Type: Nominal
Unique: 0 (0%)
No. Label Count Weight
1 t 1047 1047
Attributes
All
None
Invert
Pattern
No. Name
1 department1
2 department2
3 department3
4 department4
5 department5
6 department6
7 department7
8 department8
9 department9
10 grocery misc
11 department11
12 baby needs
13 bread and cake
14 baking needs
15 coupons
Remove
Class: total (Nom)
Visualize All
1047


Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Start
Stop
Associator output
Result list (right-click for options)

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Associator

Choose **Apriori** -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

Start | Stop

Result list (right-click for ...

11:14:49 - Apriori

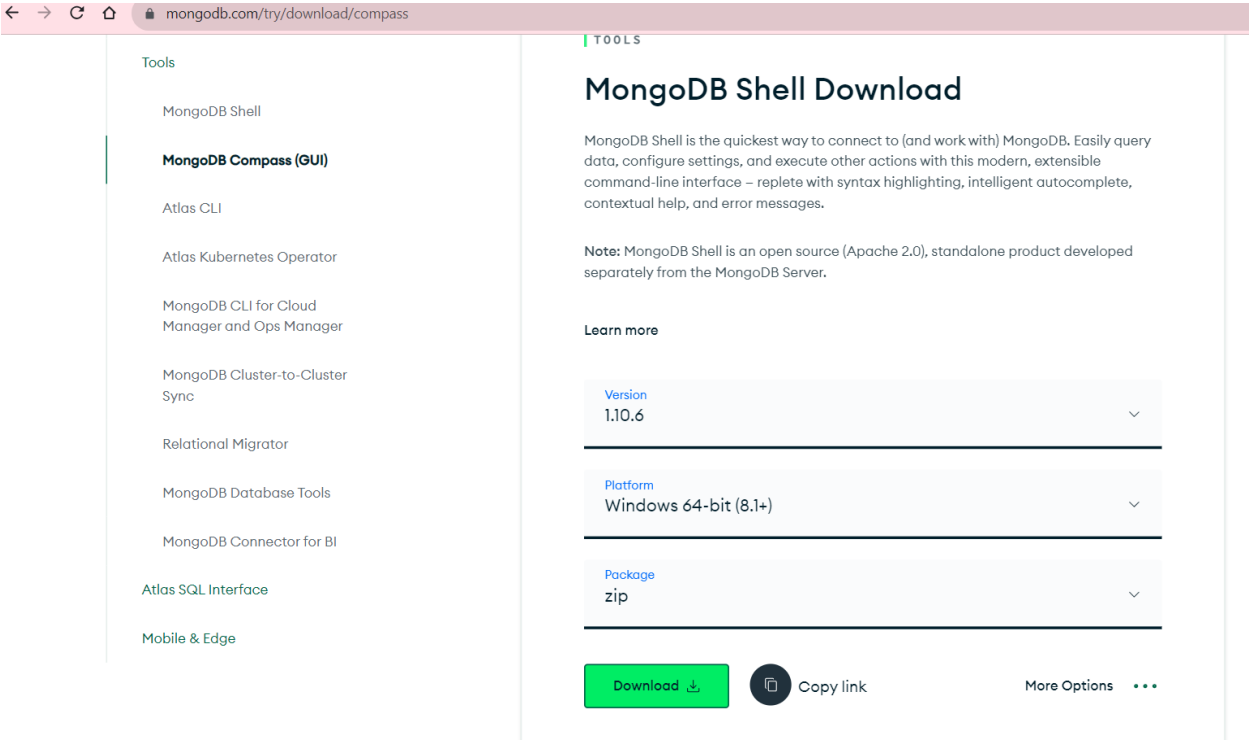
Associator output

```
=== Run information ===

Scheme:       weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation:     supermarket
Instances:    4627
Attributes:   217
              [list of attributes omitted]
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.15 (694 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 44

Size of set of large itemsets L(2): 380

Size of set of large itemsets L(3): 910

Size of set of large itemsets L(4): 633

Size of set of large itemsets L(5): 105
```

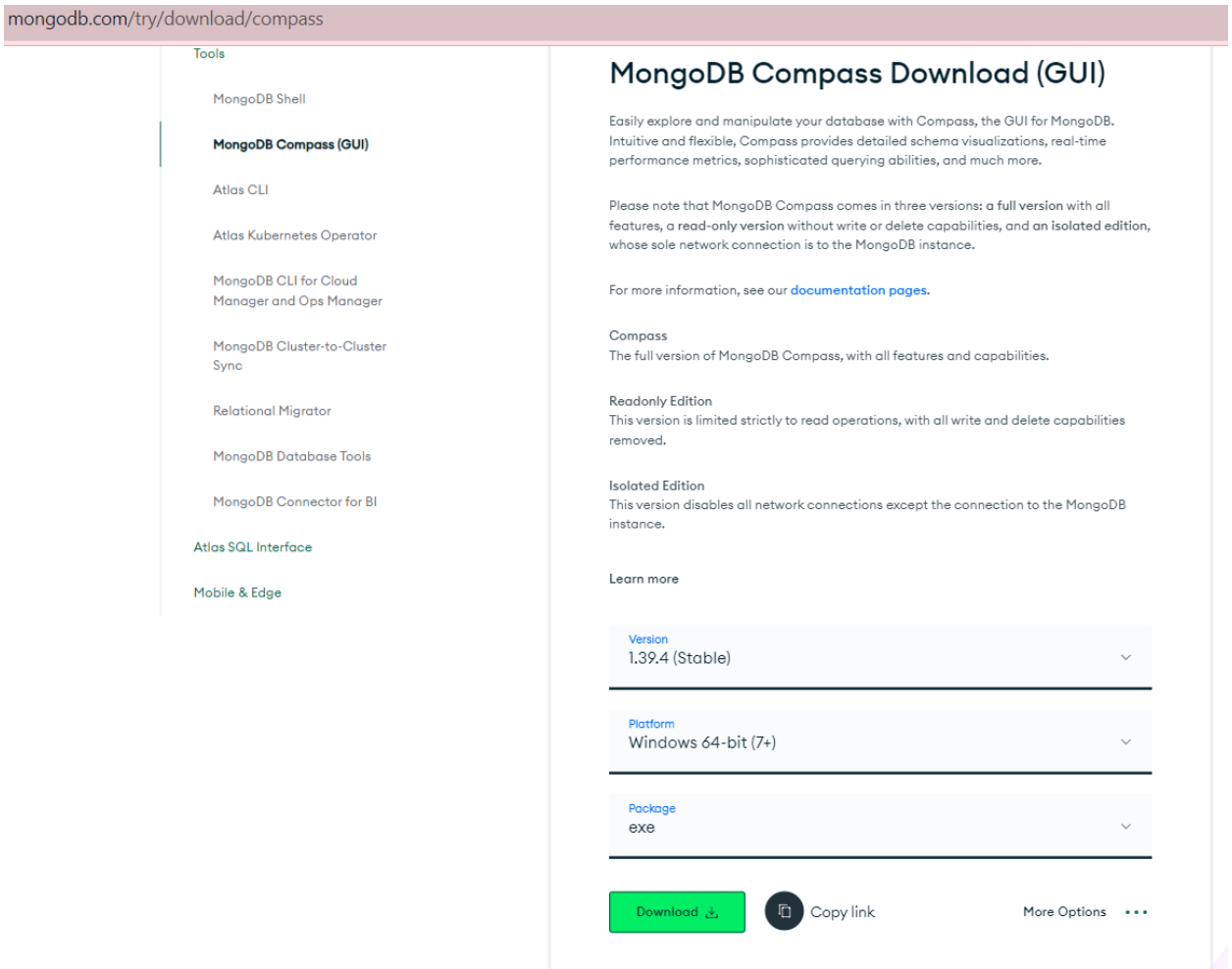
Practical No. 03 : - MongoDB Installation & Configuration

Go to :- https://www.mongodb.com/try/download/shell

Download MongoDB shell -



Download mongoDb compass -



 # After downloading install on your machine and configure it

Practical No. 04 : - <u>MongoDB CRUD Operations</u>

1) Create Database :- Let us create a database name userdb

```
> use userdb
< switched to db userdb
```

2) Creating a New Collection

```
> db.createCollection("users31")
< { ok: 1 }
```

3) Create Operation : There are 2 ways to create new documents to a collection in MongoDB:

1. insertOne():

Syntax:- db.collectionName.insertOne()

```
> db.users31.insertOne({Name:"Ankita",Age:21,Address:"Thane"})
< {
    acknowledged: true,
    insertedId: ObjectId("64d48bded302cc29d0b4f2e9")
  }
> db.users31.find()
< {
    _id: ObjectId("64d47f63d302cc29d0b4f2e6"),
    Name: 'Ankita',
    Age: 21,
    Address: 'Thane'
  }
```

2. insertMany()

Syntax:

db.collectionName.insertMany();

```
> db.users31.insertMany([{Name:"Ankita",Age:21,Address:"Thane"},{Name:"Aashu",Age:47,Address:"Vikhroli"},{Name:"Ram",Age:46,Address:"Ghatkopar"}])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("64d47f63d302cc29d0b4f2e6"),
      '1': ObjectId("64d47f63d302cc29d0b4f2e7"),
      '2': ObjectId("64d47f63d302cc29d0b4f2e8")
    }
  }
```

4) **Read Operations:** In MongoDB, read operations are used to retrieve data from the database.

1. find()

Syntax:- db.collectionName.find(query, projection)

```
> db.users31.find()
< {
    _id: ObjectId("64d47f63d302cc29d0b4f2e6"),
    Name: 'Ankita',
    Age: 21,
    Address: 'Thane'
  }
  {
    _id: ObjectId("64d47f63d302cc29d0b4f2e7"),
    Name: 'Aashu',
    Age: 47,
    Address: 'Vikhroli'
  }
  {
    _id: ObjectId("64d47f63d302cc29d0b4f2e8"),
    Name: 'Ram',
    Age: 46,
    Address: 'Ghatkopar'
  }
```

```
> db.users31.find({Age:{$gt:29}},{Name:1,Age:1})
< {
    _id: ObjectId("64d47f63d302cc29d0b4f2e7"),
    Name: 'Aashu',
    Age: 47
  }
  {
    _id: ObjectId("64d47f63d302cc29d0b4f2e8"),
    Name: 'Ram',
    Age: 46
  }
```

```
> db.users31.find({Age:{$gt:29}},{Name:1,Age:1,Address:1})
< {
    _id: ObjectId("64d47f63d302cc29d0b4f2e7"),
    Name: 'Aashu',
    Age: 47,
    Address: 'Vikhroli'
  }
  {
    _id: ObjectId("64d47f63d302cc29d0b4f2e8"),
    Name: 'Ram',
    Age: 46,
    Address: 'Ghatkopar'
  }
```

```
> db.users31.find({Age:{$lt:29}},{Name:1,Age:1,Address:1})
< {
    _id: ObjectId("64d47f63d302cc29d0b4f2e6"),
    Name: 'Ankita',
    Age: 21,
    Address: 'Thane'
  }
```

2. findOne()

The findOne() method returns a single document object, or null if no document is found. You can pass a query object to this method to filter the results.

Syntax:

db.collectionName.findOne()

```
> db.part_2.findOne({ name: "Jim" })
< {
    _id: ObjectId("64d5ad88d47c56bff07a5800"),
    name: 'Jim',
    age: 29,
    status: 'active'
  }
```

5) Update Operations

In MongoDB, the "update" operation is used to modify existing documents in a collection.

Methods:

There are several ways to perform an update operation, including the following:

1. updateOne()

The updateOne() method is used to update a single document that matches a specified filter.

Syntax:

db.collectionName.updateOne(filter, update, options)

```
> db.part_2.updateOne({ name: "Angela" }, { $set: { email: "angela@gmail.com" }})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 0,
    upsertedCount: 0
  }
> db.part_2.find()
< {
    _id: ObjectId("64d59fb9d47c56bff07a57e2"),
    name: 'Angela',
    age: 27,
    email: 'angela@gmail.com'
  }
  {
    _id: ObjectId("64d59fb9d47c56bff07a57e3"),
    name: 'Dwight',
    age: 30
  }
  {
    _id: ObjectId("64d59fb9d47c56bff07a57e4"),
    name: 'Jim',
    age: 29
  }
```

2. updateMany

The updateMany() method is used to update multiple documents that match a specified filter.

Syntax:

db.collectionName.updateMany(filter, update, options)

```
> db.part_2.updateMany({ age: { $lt: 30 } }, { $set: { status: "active" }})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 2,
    modifiedCount: 2,
    upsertedCount: 0
  }
> db.part_2.find()
< {
    _id: ObjectId("64d59fb9d47c56bff07a57e3"),
    name: 'Dwight',
    age: 30
  }
  {
    _id: ObjectId("64d5ad88d47c56bff07a57fe"),
    name: 'Angela',
    age: 27,
    status: 'active'
  }
  {
    _id: ObjectId("64d5ad88d47c56bff07a57ff"),
    name: 'Dwight',
    age: 30
  }
  {
    _id: ObjectId("64d5ad88d47c56bff07a5800"),
    name: 'Jim',
    age: 29,
    status: 'active'
  }
```

6) Delete Operations

In MongoDB, the "delete" operation is used to remove documents from a collection.

There are several ways to perform a delete operation, including the following:

1. deleteOne()

The deleteOne() method is used to remove a single document that matches a specified filter.

Syntax:

db.collectionName.deleteOne(filter, options)

```
> db.part_2.deleteOne({name:"Angela"})
< {
    acknowledged: true,
    deletedCount: 1
  }
> db.part_2.find()
< {
    _id: ObjectId("64d59fb9d47c56bff07a57e3"),
    name: 'Dwight',
    age: 30
  }
  {
    _id: ObjectId("64d59fb9d47c56bff07a57e4"),
    name: 'Jim',
    age: 29
  }
```

2. deleteMany()

The deleteMany() method is used to remove multiple documents that match a specified filter.

Syntax:

db.collectionName.deleteMany(filter, options)

```
> db.part_2.find()
< {
    _id: ObjectId("64d59fb9d47c56bff07a57e3"),
    name: 'Dwight',
    age: 30
  }
  {
    _id: ObjectId("64d59fb9d47c56bff07a57e4"),
    name: 'Jim',
    age: 29
  }
> db.part_2.deleteMany({age:{$lt:30}})
< {
    acknowledged: true,
    deletedCount: 1
  }
> db.part_2.find()
< {
    _id: ObjectId("64d59fb9d47c56bff07a57e3"),
    name: 'Dwight',
    age: 30
  }
```

3. drop()

The drop() method is used to remove an entire collection.

Syntax:

db.collectionName.drop()

```
> db.sales.drop()
< true
```

Practical 5: MongoDB aggregation operations

The operations on each stage can be one of the following:

- $project – select fields for the output documents.
- $match – select documents to be processed.
- $limit – limit the number of documents to be passed to the next stage.
- $skip – skip a specified number of documents.
- $sort – sort documents.
- $group – group documents by a specified key.

The following shows the syntax for defining an aggregation pipeline:

db.collection.aggregate([{ $match:...},{$group:...},{$sort:...}]);

In this syntax:

- First, call the aggregate() method on the collection.
- Second, pass an array of documents, where each document describes a stage in the pipeline.

MongoDB aggregation example

First, switch to the coffeeshop database that stores the coffee sales:

– use coffeeshop

Second, insert documents into the sales collection:

– db.sales.insertMany([

{ "_id" : 1, "item" : "Americanos", "price" : 5, "size": "Short", "quantity" : 22, "date" : ISODate("2022-01-15T08:00:00Z") },

{ "_id" : 2, "item" : "Cappuccino", "price" : 6, "size": "Short","quantity" : 12, "date" : ISODate("2022-01-16T09:00:00Z") },

{ "_id" : 3, "item" : "Lattes", "price" : 15, "size": "Grande","quantity" : 25, "date" : ISODate("2022-01-16T09:05:00Z") },

```
{ "_id" : 4, "item" : "Mochas", "price" : 25,"size": "Tall", "quantity" : 11, "date" : ISODate("2022-02-17T08:00:00Z") },

{ "_id" : 5, "item" : "Americanos", "price" : 10, "size": "Grande","quantity" : 12, "date" : ISODate("2022-02-18T21:06:00Z") },

{ "_id" : 6, "item" : "Cappuccino", "price" : 7, "size": "Tall","quantity" : 20, "date" : ISODate("2022-02-20T10:07:00Z") },

{ "_id" : 7, "item" : "Lattes", "price" : 25,"size": "Tall", "quantity" : 30, "date" : ISODate("2022-02-21T10:08:00Z") },

{ "_id" : 8, "item" : "Americanos", "price" : 10, "size": "Grande","quantity" : 21, "date" : ISODate("2022-02-22T14:09:00Z") },

{ "_id" : 9, "item" : "Cappuccino", "price" : 10, "size": "Grande","quantity" : 17, "date" : ISODate("2022-02-23T14:09:00Z") },

{ "_id" : 10, "item" : "Americanos", "price" : 8, "size": "Tall","quantity" : 15, "date" : ISODate("2022-02-25T14:09:00Z")}

]);
```

Third, use an aggregation pipeline to filter the sales by the Americanos, calculate the sum of quantity grouped by sizes, and sort the result document by the total quantity in descending order.

```
db.sales.aggregate([
        {
                $match: { item: "Americanos" }
        },
        {
                $group: {
                        _id: "$size",
                        totalQty: {$sum: "$quantity"}
                }
```

```
        },
        {
                $sort: { totalQty : -1}
        }
]);
[
  { _id: 'Grande', totalQty: 33 },
  { _id: 'Short', totalQty: 22 },
  { _id: 'Tall', totalQty: 15 }
]
```

```
> db.sales.aggregate([{$match:{item:"Americanos"}},{$group:{_id:"$size",totalQty:{$sum:"$quantity"}}},{$sort:{totalQty:-1}}]);
< {
    _id: 'short',
    totalQty: 110
  }
```

```
> db.sales.aggregate([{$match:{item:"Americanos"}},{$group:{_id:"$size",totalQty:{$sum:"$quantity"}}},{$sort:{totalQty:1}}]);
< {
    _id: 'short',
    totalQty: 44
  }
  {
    _id: 'tall',
    totalQty: 44
  }
```

Practical No. 06 : - Sort, Limit, Skip operation in MONGODb

```
> db.cars.insertMany([{"make":"Nissan","model":"GTR","year":2016,"type":"Sports","reg_no":"EDS 5578"},
  {"make":"BMW","model":"X5","year":2020,"type":"SUV","reg_no":"LLS 6899"},
  {"make":"Toyata","model":"Yaris","year":2019,"type":"Compact","reg_no":"HXE 0153"},
  {"make":"Audi","model":"RS3","year":2018,"type":"Sports","reg_no":"RFD 7866"},
  {"make":"Ford","model":"Transit","year":2017,"type":"Van","reg_no":"TST 9880"},
  {"make":"Honda","model":"Gold Wing","year":2018,"type":"Bike","reg_no":"LKS 2477"}])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("64dd86755db01734109aeceb"),
      '1': ObjectId("64dd86755db01734109aecec"),
      '2': ObjectId("64dd86755db01734109aeced"),
      '3': ObjectId("64dd86755db01734109aecee"),
      '4': ObjectId("64dd86755db01734109aecef"),
      '5': ObjectId("64dd86755db01734109aecf0")
    }
  }
```

```
> db.cars.find()
< {
    _id: ObjectId("64d5b405a0a1a00ededbcc4d"),
    make: 'Mahindra',
    model: 'XUV',
    year: 2019,
    type: 'classic',
    reg_no: 'xuv700'
  }
  {
    _id: ObjectId("64d5b405a0a1a00ededbcc4e"),
    make: 'BMW',
    model: 'X5',
    year: 2020,
    type: 'SUV',
    reg_no: 'x123'
  }
  {
    _id: ObjectId("64d5b44ea0a1a00ededbcc4f"),
    make: 'Nissan',
    model: 'GTR',
    year: 2021,
    type: 'Sports',
    reg_no: 'gtr100'
  }
```

<u>sort</u>({year:1}) : Sorts the year in ascending order, -1 sorts in descending order

```
> db.cars.find({},{_id:0}).sort({year:1})
< {
    make: 'Ford',
    model: 'Transit',
    year: 2011,
    type: 'Van',
    reg_no: 'for12'
  }
  {
    make: 'Mahindra',
    model: 'XUV',
    year: 2019,
    type: 'classic',
    reg_no: 'xuv700'
  }
```

```
> db.cars.find({},{make:1, _id:0}).sort({make:1})
< {
    make: 'BMW'
  }
  {
    make: 'Ford'
  }
  {
    make: 'Honda'
  }
  {
    make: 'Mahindra'
  }
  {
    make: 'Nissan'
  }
  {
    make: 'Toyota'
  }
```

Limit

pretty() : to display it proper

```
> db.cars.find({},{_id:0}).sort({make:1,year:1}).limit(2).pretty()
< {
    make: 'BMW',
    model: 'X5',
    year: 2020,
    type: 'SUV',
    reg_no: 'x123'
  }
  {
    make: 'Ford',
    model: 'Transit',
    year: 2011,
    type: 'Van',
    reg_no: 'for12'
  }
```

```
> db.cars.find({},{_id:0}).sort({make:1,year:1}).skip(4).limit(2).pretty()
< {
    make: 'Nissan',
    model: 'GTR',
    year: 2021,
    type: 'Sports',
    reg_no: 'gtr100'
  }
  {
    make: 'Toyota',
    model: 'Yaris',
    year: 2021,
    type: 'Compact',
    reg_no: 'hxe153'
  }
```

skip(4) : skip start 4 columns and displays rest.

```
> db.cars.find({},{_id:0}).sort({make:1,year:1}).skip(4).limit(2).pretty()
< {
    make: 'Nissan',
    model: 'GTR',
    year: 2021,
    type: 'Sports',
    reg_no: 'gtr100'
  }
  {
    make: 'Toyota',
    model: 'Yaris',
    year: 2021,
    type: 'Compact',
    reg_no: 'hxe153'
  }
```

Practical 07 : Comparison operators

```
> use supermarket;
< switched to db supermarket
> db.employee.insertMany([{"_id":001,"emp_name":"Siddhesh", "emp_age":22,"job_role":"Data Analyst","sal":200000}])
< {
    acknowledged: true,
    insertedIds: {
      '0': 1
    }
  }
```

Similarly create more 5 records.

```
> db.employee.find()
< {
    _id: 1,
    emp_name: 'Siddhesh',
    emp_age: 22,
    job_role: 'Senior Manager',
    sal: 200000
  }
  {
    _id: 2,
    emp_name: 'Gautham',
    emp_age: 23,
    job_role: 'Cashier',
    sal: 150000
  }
  {
    _id: 3,
    emp_name: 'Jayesh',
    emp_age: 21,
    job_role: 'Store Associate',
    sal: 250000
  }
```

```
> db.createCollection("inventory")
< { ok: 1 }
> db.inventory.insertMany([{"_id":"SM01", "name":"Chocolate Bar - 100 g", "price":5.23, "quantity":25000, "category":["chocolate","sweets"]}])
< {
    acknowledged: true,
    insertedIds: {
      '0': 'SM01'
    }
  }
> db.inventory.insertMany([{"_id":"SM02", "name":"Milk 1Lt", "price":3, "quantity":1000, "category":["dairy","healthy"]},{"_id":"SM03", "name":
  {"_id":"SM07","name": "ZZ Butter 500g", "price": 25, "quantity": 500, "category" : ["dairy", "healthy","premium"]},
  {"_id":"SM08","name": "Beans (Packed) - 250g", "price": 6.75, "quantity" : 6000, "category": ["vegetables", "healthy", "organic"]}])
< {
    acknowledged: true,
    insertedIds: {
      '0': 'SM02',
      '1': 'SM03',
      '2': 'SM04',
      '3': 'SM05',
      '4': 'SM06',
      '5': 'SM07',
      '6': 'SM08'
    }
  }
```

```
> db.inventory.find()
< {
    _id: 'SM01',
    name: 'Chocolate Bar - 100 g',
    price: 5.23,
    quantity: 25000,
    category: [
      'chocolate',
      'sweets'
    ]
  }
  {
    _id: 'SM02',
    name: 'Milk 1Lt',
    price: 3,
    quantity: 1000,
    category: [
      'dairy',
      'healthy'
    ]
  }
```

```
> db.payments.insertMany([
 {"_id": "BL2021005", "gross_amount": 105.65, "discounts": 10, "net_amount": 95.65, "date_time":ISODate("2021-01-01T16:15:55Z")},
 {"_id": "BL2021006", "gross_amount": 45.25, "discounts" : 0, "net_amount": 45.25, "date_time":ISODate("2021-01-01T16:00:00Z")},
 {"_id":"BL2021007", "gross_amount": 153.33, "discounts" : 20.33, "net_amount": 133, "date_time":ISODate("2021-01-01T16:31:08Z")},
 {"_id": "BL2021008", "gross amount": 21, "discounts" : 0, "net_amount": 21, "date_time" : ISODate("2021-01-01T20:25:52Z")},
 {"_id": "BL2021009", "gross amount": 89.72, "discounts": 0.72, "net_amount": 89, "date_time":ISODate("2021-01-02T08:45:12Z")},
 {"_id": "BL2021010", "gross_amount": 33.5, "discounts": 20.5, "net_amount": 13, "date_time":ISODate("2021-01-02T11:02:35Z")}
 ])
< {
   acknowledged: true,
   insertedIds: {
     '0': 'BL2021005',
     '1': 'BL2021006',
     '2': 'BL2021007',
     '3': 'BL2021008',
     '4': 'BL2021009',
     '5': 'BL2021010'
   }
 }
```

```
> db.payments.find()
< {
    _id: 'BL2021005',
    gross_amount: 105.65,
    discounts: 10,
    net_amount: 95.65,
    date_time: 2021-01-01T16:15:55.000Z
  }
  {
    _id: 'BL2021006',
    gross_amount: 45.25,
    discounts: 0,
    net_amount: 45.25,
    date_time: 2021-01-01T16:00:00.000Z
  }
```

```
> db.createCollection("promo")
< { ok: 1 }
> db.promo.insertMany([
  {"_id": "PROMO01", "name": "Sales Promo", "period": 7, "daily sales" : [ 20, 50, 12, 30, 45, 15,60]},
  {"_id": "PROMO02", "name": "Milk Promo", "period": 2, "daily sales" : [ 120, 200 ]},
  {"_id": "PROMO03", "name": "Meat Promo", "period": 3, "daily sales": [ 101, 250 ]},
  {"_id": "PROMO04", "name": "New Year Promo", "period": 7, "daily sales": [ 65, 88, 105, 188, 74, 278, 350 ]}
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': 'PROMO01',
      '1': 'PROMO02',
      '2': 'PROMO03',
      '3': 'PROMO04'
    }
  }
```

```
> db.promo.find()
< {
    _id: 'PROMO01',
    name: 'Sales Promo',
    period: 7,
    'daily sales': [
      20,
      50,
      12,
      30,
      45,
      15,
      60
    ]
  }
  {
```

Queries
Equal operator

```
> db.inventory.find({"_id":{ $eq:"SM08"}}).pretty()
< {
    _id: 'SM08',
    name: 'Beans (Packed) - 250g',
    price: 6.75,
    quantity: 6000,
    category: [
      'vegetables',
      'healthy',
      'organic'
    ]
  }
```

```
> db.inventory.find({"_id":{ $eq:"SM08"}})
< {
    _id: 'SM08',
    name: 'Beans (Packed) - 250g',
    price: 6.75,
    quantity: 6000,
    category: [
      'vegetables',
      'healthy',
      'organic'
    ]
  }
```

Greater operator

```
> db.inventory.find({"quantity":{ $gt:12000}}).pretty()
< {
    _id: 'SM01',
    name: 'Chocolate Bar - 100 g',
    price: 5.23,
    quantity: 25000,
    category: [
      'chocolate',
      'sweets'
    ]
  }
```

Greater than equal

```
> db.inventory.find({"quantity":{ $gte:12000}}).pretty()
< {
    _id: 'SM01',
    name: 'Chocolate Bar - 100 g',
    price: 5.23,
    quantity: 25000,
    category: [
      'chocolate',
      'sweets'
    ]
  }
  {
    _id: 'SM06',
    name: 'Bell Pepper (Packed) - 250g',
    price: 4.95,
    quantity: 12000,
    category: [
      'vegetables',
      'healthy',
      'organic'
    ]
  }
```

Less than equal

```
> db.inventory.find({"quantity":{ $lte:1000}}).pretty()
< {
    _id: 'SM02',
    name: 'Milk 1Lt',
    price: 3,
    quantity: 1000,
    category: [
      'dairy',
      'healthy'
    ]
  }
  {
    _id: 'SM07',
    name: 'ZZ Butter 500g',
    price: 25,
    quantity: 500,
    category: [
      'dairy',
      'healthy',
      'premium'
    ]
  }
```

Not equal

```
> db.employee.find({ $nor: [{"job_role":"Store Associate"}, {"emp_age": {$gte:21, $lte:22}}]}).pretty()
< {
    _id: 2,
    emp_name: 'Gautham',
    emp_age: 23,
    job_role: 'Cashier',
    sal: 150000
  }
  {
    _id: 5,
    emp_name: 'Aarti',
    emp_age: 26,
    job_role: 'Senior Cashier',
    sal: 50000
  }
```

```
> db.promo.find({"period":{ $ne:7}}).pretty()
< {
    _id: 'PROMO02',
    name: 'Milk Promo',
    period: 2,
    'daily sales': [
      120,
      200
    ]
  }
  {
    _id: 'PROMO03',
    name: 'Meat Promo',
    period: 3,
    'daily sales': [
      101,
      250
    ]
  }
```

Practical 8 :- Logical Operators

And

```
> db.employee.find({ $and: [{"job_role":"Store Associate"}, {"emp_age": {$gte:21, $lte:23}}]}).pretty()
< {
    _id: 3,
    emp_name: 'Jayesh',
    emp_age: 21,
    job_role: 'Store Associate',
    sal: 250000
  }
```

Or (If any condition matches, it displays the result)

```
> db.employee.find({ $or: [{"job_role":"Store Associate"}, {"emp_age": {$gte:21, $lte:22}}]}).pretty()
< {
    _id: 1,
    emp_name: 'Siddhesh',
    emp_age: 22,
    job_role: 'Senior Manager',
    sal: 200000
  }
  {
    _id: 3,
    emp_name: 'Jayesh',
    emp_age: 21,
    job_role: 'Store Associate',
    sal: 250000
  }
  {
    _id: 4,
    emp_name: 'Pratik',
    emp_age: 25,
    job_role: 'Store Associate',
    sal: 100000
  }
```

Nor (Opposite of OR)

```
> db.employee.find({ $nor: [{"job_role":"Store Associate"}, {"emp_age": {$gte:21, $lte:22}}]}).pretty()
< {
    _id: 2,
    emp_name: 'Gautham',
    emp_age: 23,
    job_role: 'Cashier',
    sal: 150000
  }
  {
    _id: 5,
    emp_name: 'Aarti',
    emp_age: 26,
    job_role: 'Senior Cashier',
    sal: 50000
  }
```

Practical 9: MongoDB $abs, $floor, $ceil Operator

1. Database: userdb
2. Collection: student
3. Document: Six documents that contain the details of the students


```
{
    {
        "_id" : ObjectId("56254d4fdf2222265r4g12ds3d65f"),
        "std_name" : "Micky",
        "gender" : "Female",
        "class" : "X",
        "fees" : 5000,
        "exam_fees" : 500,
        "age" : 16,
        "Total_marks" : 405
        "Result" : "Pass"
    },
    {
        "_id" : ObjectId("56254d4fdf2222265r4g12ds34563"),
        "std_name" : "Moty",
        "gender" : "Male",
        "fees" : 4000,
        "exam_fees" : 500,
        "class" : "VII",
        "age" : 15,
        "Total_marks" : 705
```

"Result" : "Pass"
},
{
"_id" : ObjectId("56254d4fdf2222265r4g12ds31478"),
"std_name" : "Thomas",
"gender" : "Male",
"fees" : 3000,
"exam_fees" : 500,
"class" : "V",
"age" : 12,
" Total_marks" : 450
" Result" : "pass"
},
{
"_id" : ObjectId("56254d4fdf2222265r4g12ds37832"),
"std_name" : "Jin",
"gender" : "Female",
"fees" : 5000,
"exam_fees" : 500,
"class" : "X",
"age" : 16,
"Total_marks" : 750
"Result" : "Pass"
},
{
"_id" : ObjectId("56254d4fdf2222265r4g12ds1c46"),
"std_name" : "Mia",

"gender" : "Female",
"fees" : 6000,
"exam_fees" : 500,
"class" : "XI",
"age" : 17,
" Total_marks" : 450
"Result" : "Pass"
},
{
"_id" : ObjectId("56254d4fdf2222265r4g12ds315hj"),
"std_name" : "Mike,
"gender" : "Male",
"fees" : {
"school_fees" :4000,
"exam_fees" : 500,
"pending_fees" : 950,
}
"class" : "V",
"age" : 15,
"Total_marks" : 450
"Result" : "Pass"
}
}

```
> db.student.aggregate([{$match: {gender:"Female"}},{$project:{name:1,class:1,age:1,Result:1,Total_fees:{$abs:{$add:["$fees","$exam_fees"]}}}}])
< {
    _id: ObjectId("64deeda57a442f98a5c0f61a"),
    class: 'X',
    age: 16,
    Result: 'Pass',
    Total_fees: 5500
  }
  {
    _id: ObjectId("64deeda57a442f98a5c0f61d"),
    class: 'X',
    age: 16,
    Result: 'Pass',
    Total_fees: 5500
  }
  {
    _id: ObjectId("64deeda57a442f98a5c0f61e"),
    class: 'XI',
    age: 17,
    Result: 'Pass',
    Total_fees: 6500
  }
```

```
> db.student.aggregate([{$match: {gender:"Female"}},{$project:{std_name:1,class:1,age:1,floor_grade:{$floor:"$grade"}}}])
< {
    _id: ObjectId("64deeda57a442f98a5c0f61a"),
    std_name: 'Micky',
    class: 'X',
    age: 16,
    floor_grade: null
  }
  {
    _id: ObjectId("64deeda57a442f98a5c0f61d"),
    std_name: 'Jin',
    class: 'X',
    age: 16,
    floor_grade: null
  }
  {
    _id: ObjectId("64deeda57a442f98a5c0f61e"),
    std_name: 'Mia',
    class: 'XI',
    age: 17,
    floor_grade: null
  }
```

```
> db.student.aggregate([{$match: {gender:"Female"}},{$project:{std_name:1,class:1,age:1,ceil_grade:{$ceil:"$grade"}}}])
< {
    _id: ObjectId("64deeda57a442f98a5c0f61a"),
    std_name: 'Micky',
    class: 'X',
    age: 16,
    ceil_grade: null
  }
  {
    _id: ObjectId("64deeda57a442f98a5c0f61d"),
    std_name: 'Jin',
    class: 'X',
    age: 16,
    ceil_grade: null
  }
  {
    _id: ObjectId("64deeda57a442f98a5c0f61e"),
    std_name: 'Mia',
    class: 'XI',
    age: 17,
    ceil_grade: null
  }
```

Practical 10: MongoDB $log, $mod, $divide, $multiply Operator

1. Database: userdb
2. Collection: shapes
3. Document: Six documents that contain the details of the shapes

```
>db.example1.find().pretty()
{
    {
     "_id" : ObjectId("56254d4fdf2222265r4g1hb78452"),
     "name" : "rectangle",
     "area" : 16
    }
    {
     "_id" : ObjectId("56254d4fdf2222265r4g1hb71478"),
     "name" : "rectangle",
     "area" : 6
    }
    {
     "_id" : ObjectId("56254d4fdf2222265r4g1789654"),
     "name" : "circle",
     "area" : 19,
     "unit" : {
                "diameter" : 6,
                "radius" : 3
               }
```

```
    }
    {
     "_id" : ObjectId("56254d4fdf2222265r4g1987412"),
     "name" : "rectangle",
     "area" : 20
    }
    {
     "_id" : ObjectId("56254d4fdf2222265r4g1987412"),
     "name" : "square",
     "area" : 20
    }
    {
     "_id" : ObjectId("56254d4fdf2222265r4g1987f15"),
     "name" : "triangle",
     "area" : null
    }
}
```

```
> db.shapes.aggregate([{$match:{name:"rectangle"}},{$project:{name:1,area:1,logArea:{$log:["$area",10]}}}])
< {
    _id: ObjectId("64dedd257a442f98a5c0f614"),
    name: 'rectangle',
    area: 16,
    logArea: 1.2041199826559246
  }
  {
    _id: ObjectId("64dedd257a442f98a5c0f615"),
    name: 'rectangle',
    area: 6,
    logArea: 0.7781512503836435
  }
  {
    _id: ObjectId("64dedd257a442f98a5c0f617"),
    name: 'rectangle',
    area: 20,
    logArea: 1.301029995663981
  }
```

MongoDB $mod Operator

1. Database: userdb
2. Collection: items
3. Document: Ten documents that contain the details of the items

```
>db.items.find().pretty()
{
  {
    "_id" : 1,
    "item_name" : "Apple",
    "total_Price" : null,
    "quantity" : 40,
  }
  {
    "_id" : 2,
```

"item_name" : "Banana",
"total_Price" : 1000,
"quantity" : 72,
}
{
"_id" : 3,
"item_name" : "Cherry",
"total_Price" : 215,
"quantity" : 25,
}
{
"_id" : 4,
"item_name" : "Apple",
"total_Price" : null,
"quantity" : 25,
}
{
"_id" : 5,
"item_name" : "Banana",
"total_Price" : 400,
"quantity" : 35,
}
{
"_id" : 6,
"item_name" : "Banana",
"total_Price" : 510,
"quantity" : 100,

```
  }
  {
    "_id" : 7,
    "item_name" : "Cherry",
    "total_Price" : 500,
    "quantity" : 41,
  }
  {
    "_id" : 8,
    "item_name" : "Rasbhari",
    "total_Price" : 80,
    "quantity" : "Ten",
  }
  {
    "_id" : 9,
    "item_name" : "Banana",
    "total_Price" : 205,
    "quantity" : 10,
  }
  {
    "_id" : 10,
    "item_name" : "Apple",
    "total_Price" : 95,
    "quantity" : null,
  }
}
```

```
> db.items.aggregate([{$match:{item_name:"Banana"}},{$project:{item_name:1,total_Price:1,quantity:1,remainderValue:{$mod:["$total_Price","$quantity"]}}}])
< {
    _id: 2,
    item_name: 'Banana',
    total_Price: 1000,
    quantity: 72,
    remainderValue: 64
  }
  {
    _id: 5,
    item_name: 'Banana',
    total_Price: 400,
    quantity: 35,
    remainderValue: 15
  }
  {
    _id: 6,
    item_name: 'Banana',
    total_Price: 510,
    quantity: 100,
    remainderValue: 10
  }
  {
    _id: 9,
    item_name: 'Banana',
    total_Price: 205,
    quantity: 10,
    remainderValue: 5
  }
```

MongoDB $divide Operator

1. Database: userdb
2. Collection: products
3. Document: Ten documents that contain the details of each product

```
>db.products.find().pretty()
{
  {
    "_id" : 1,
    "name" : "Mobile",
    "totalPrice" : 100000,
    "totalQuantity" : 50,
    "billYear" : 2018
  }
```

{
"_id" : 2,
"name" : "Keyboard",
"totalPrice" : 5000,
"totalQuantity" : 10,
"billYear" : 2017
}
{
"_id" : 3,
"name" : "Mouse",
"totalPrice" : 2000,
"totalQuantity" : 5,
"billYear" : 2018
}
{
"_id" : 4,
"name" : "Memory Card",
"totalPrice" : 2500,
"totalQuantity" : 25,
"billYear" : 2019
}
{
"_id" : 5,
"name" : "Mobile",
"totalPrice" : 20000,
"totalQuantity" : 4,
"billYear" : 2020

}
{
"_id" : 6,
"name" : "Mobile",
"totalPrice" : 25000,
"totalQuantity" : 2,
"billYear" : 2021
}
{
"_id" : 7,
"name" : "Memory Card",
"totalPrice" : 1000,
"totalQuantity" : 10,
"billYear" : 2019
}
{
"_id" : 8,
"name" : "Pen drive",
"totalPrice" : 15000,
"totalQuantity" : "Two",
"billYear" : 2018
}
{
"_id" : 9,
"name" : "Laptop",
"billDetail" : {
"totalPrice" : 45000,

"totalQuantity" : 1,
}
"billYear" : 2021
}
{
"_id" : 10,
"name" : "Memory Carde",
"totalPrice" : 4000,
"totalQuantity" : 50,
"billYear" : 2020
}
}

```
> db.hwproducts.aggregate([{$match:{name:"Mobile"}},{$project:{name:1,totalPrice:1,totalQuantity:1,mobilePrice:{$divide:["$totalPrice","$totalQuantity"]}}}])
< {
    _id: 1,
    name: 'Mobile',
    totalPrice: 100000,
    totalQuantity: 50,
    mobilePrice: 2000
  }
  {
    _id: 5,
    name: 'Mobile',
    totalPrice: 20000,
    totalQuantity: 4,
    mobilePrice: 5000
  }
  {
    _id: 6,
    name: 'Mobile',
    totalPrice: 25000,
    totalQuantity: 2,
    mobilePrice: 12500
  }
```

MongoDB $multiply Operator

1. Database: userdb
2. Collection: products

3. Document: Ten documents that contain the details of each product

```
>db.products.find().pretty()
{
  {
    "_id" : 1,
    "name" : "BlueBox",
    "x" : 10,
    "y" : 50,
    "billYear" : 2018
  }
  {
    "_id" : 2,
    "name" : "GreenBox",
    "x" : 10,
    "y" : 6,
    "billYear" : 2017
  }
  {
    "_id" : 3,
    "name" : "RedBox",
    "x" : 7,
    "y" : 9,
    "billYear" : 2018
  }
  {
    "_id" : 4,
```

"name" : "WhiteBox",
"x" : 2,
"y" : 7,
"z" : 4,
"billYear" : 2019
}
{
"_id" : 5,
"name" : "BlueBox",
"x" : 4,
"y" : 12,
"billYear" : 2020
}
{
"_id" : 6,
"name" : "BlueBox",
"x" : 10,
"y" : 5,
"billYear" : 2021
}
{
"_id" : 7,
"name" : "WhiteBox",
"x" : 5,
"y" : 1,
"z" : 45,
"billYear" : 2019

```
  }
  {
    "_id" : 8,
    "name" : "GreenBox",
    "x" : -15,
    "y" : 5,
    "billYear" : 2018
  }
  {
    "_id" : 9,
    "name" : "BlackBox",
    "billDetail" : {
                    "x" : 45,
                    "y" : 56,
                   }
    "billYear" : 2021
  }
  {
    "_id" : 10,
    "name" : "WhiteBox",
    "x" : 4,
    "y" : 5,
    "z" : 6,
    "billYear" : 2020
  }
}
```

```
> db.swproducts.aggregate([{$match:{name:"BlueBox"}},{$project:{name:1,_id:1,Price:{$multiply:["$x","$y"]}}}])
< {
    _id: 1,
    name: 'BlueBox',
    Price: 500
  }
  {
    _id: 5,
    name: 'BlueBox',
    Price: 48
  }
  {
    _id: 6,
    name: 'BlueBox',
    Price: 50
  }
```

Practical 11: MongoDB $pow, $sqrt, $subtract Operator

1. Database: userdb
2. Collection: shapes
3. Document: Six documents that contain the details of the shapes

```
>db.shapes.find().pretty()
{
    {
     "_id" : 1,
     "name" : "rectangle",
     "area" : 16
    }
    {
     "_id" : 2,
     "name" : "square",
     "area" : 10
    }
    {
     "_id" : 3,
     "name" : "circle",
     "perimeter" : 15,
     "area" : 10,
     "details" : {
                 "radius" : 3,
                 "diameter" : 6
                }
```

```
    }
    {
     "_id" : 4,
     "name" : "rectangle",
     "area" : 0
    }
    {
     "_id" : 5,
     "name" : "oval",
     "area" : 20
    }
    {
     "_id" : 6,
     "name" : "triangle",
     "area" : 5
    }
    {
     "_id" : 7,
     "name" : "rectangle",
     "area" : null
    }
}
```

```
> db.shapes1.aggregate([{$match:{name:"rectangle"}},{$project:{name:1,area:1,result:{$pow:["$area",3]}}}])
< {
    _id: 1,
    name: 'rectangle',
    area: 16,
    result: 4096
  }
  {
    _id: 4,
    name: 'rectangle',
    area: 0,
    result: 0
  }
  {
    _id: 7,
    name: 'rectangle',
    area: null,
    result: null
  }
```

MongoDB $sqrt Operator

db.items.find().pretty()

```
{
    {
     "_id" : 1,
     "item_name" : "bat",
     "quantity" : 4
    }
    {
     "_id" : 2,
     "item_name" : "ball",
     "quantity" : null
    }
    {
     "_id" : 3,
```

"item_name" : "box",
"details" : {
"length" : 20,
"width" : 25
}
}
{
"_id" : 4,
"item_name" : "ball",
"quantity" : null
}
{
"_id" : 5,
"item_name" : "bat",
"quantity" : 20
}
{
"_id" : 6,
"item_name" : "toy",
"quantity" : -10
}
{
"_id" : 7,
"item_name" : "bat",
"quantity" : 75
}
{

"_id" : 8,

"item_name" : "bat",

"quantity" : 45

}

}

```
> db.toys.aggregate([{$match:{item_name:"bat"}},{$project:{item_name:1,quantity:1,result:{$sqrt:"$quantity"}}}])
< {
    _id: 1,
    item_name: 'bat',
    quantity: 4,
    result: 2
  }
  {
    _id: 5,
    item_name: 'bat',
    quantity: 20,
    result: 4.47213595499958
  }
  {
    _id: 7,
    item_name: 'bat',
    quantity: 75,
    result: 8.660254037844387
  }
  {
    _id: 8,
    item_name: 'bat',
    quantity: 45,
    result: 6.708203932499369
  }
```

MongoDB $subtract Operator

db.students.find().pretty()

{

{

"_id" : 1,
"std_name" : "John",
"father_name" : "Mick",
"department" : "MCA",
"semester_fee" : 6000,
"annual_fee" : 10000,
"start_date" : ISODate("2019-07-03T08:00:00Z"),
"end_date" : ISODate("2021-05-26T09:00:00Z")
}
{
"_id" : 2,
"std_name" : "Oliver",
"father_name" : "Thomas",
"department" : "BCA",
"semester_fee" : 4000,
"annual_fee" : 6000,
"start_date" : ISODate("2020-07-03T08:00:00Z"),
"end_date" : ISODate("2023-05-01T09:00:00Z")

}
{
"_id" : 3,
"std_name" : "Jack",
"father_name" : "James",
"department" : "MCA",
"semester_fee" : 7000,
"annual_fee" : 12500,
"start_date" : ISODate("2020-07-11T00:00:00Z"),
"end_date" : ISODate("2022-05-25T09:00:00Z")
}
{
"_id" : 4,
"std_name" : "Robert",
"father_name" : "David",
"department" : "Btech",
"fees" : {
"semester_fee" : 15000,

"annual_fee" : 22500
}
"start_date" : ISODate("2018-07-11T08:00:00Z"),
"end_date" : ISODate("2022-05-25T09:00:00Z")
}
{
"_id" : 5,
"std_name" : "Richard",
"father_name" : "William",
"department" : "BCA",
"semester_fee" : 11500,
"annual_fee" : 20000,
"start_date" : ISODate("2020-07-03T08:00:00Z"),
"end_date" : ISODate("2023-05-01T09:00:00Z")
}
{
"_id" : 6,
"std_name" : "Daniel",

"father_name" : "Paul",

"department" : "MCA",

"semester_fees" : 12500,

"annual_fee" : 25000,

"start_date" : ISODate("2018-07-11T08:00:00Z"),

"end_date" : ISODate("2020-05-25T09:00:00Z")

}

}

```
> db.student1.aggregate([{$match:{department:"MCA"}},{$project:{std_name:1,father_name:1,annual_fee:1,semester_fee:1,result:{$subtract:["$annual_fee","$semester_fee"]}}}])
< {
    _id: 1,
    std_name: 'John',
    father_name: 'Mick',
    semester_fee: 6000,
    annual_fee: 10000,
    result: 4000
  }
  {
    _id: 3,
    std_name: 'Jack',
    father_name: 'James',
    semester_fee: 7000,
    annual_fee: 12500,
    result: 5500
  }
  {
    _id: 6,
    std_name: 'Daniel',
    father_name: 'Paul',
    annual_fee: 25000,
    result: null
  }
```

Practical 12: MongoDB $trunc, $round, $cmp Operator

$trunc

```
db.student.aggregate([
 {$project:{grade:1,value:{$trunc:["$grade"]}}}
]);
```

```
> db.student.aggregate([
    {$project:{grade:1,value:{$trunc:["$grade"]}}}
  ]);
< {
    _id: ObjectId("64de0b5ce572de65422ce337"),
    grade: Decimal128("7.85"),
    value: Decimal128("7")
  }
  {
    _id: ObjectId("64de0b5ce572de65422ce338"),
    grade: Decimal128("8.5"),
    value: Decimal128("8")
  }
  {
    _id: ObjectId("64de0b5ce572de65422ce339"),
    grade: Decimal128("7.1"),
    value: Decimal128("7")
  }
```

$round

```
db.student.aggregate([
 {$project:{grade:1,value:{$round:["$grade"]}}}
]);
```

```
> db.student.aggregate([
    {$project:{grade:1,value:{$round:["$grade"]}}}
  ]);
< {
    _id: ObjectId("64de0b5ce572de65422ce337"),
    grade: Decimal128("7.85"),
    value: Decimal128("8")
  }
  {
    _id: ObjectId("64de0b5ce572de65422ce338"),
    grade: Decimal128("8.5"),
    value: Decimal128("8")
  }
  {
    _id: ObjectId("64de0b5ce572de65422ce339"),
    grade: Decimal128("7.1"),
    value: Decimal128("7")
  }
```

$cmp

Collection = area

```
db.area.insertMany([
 { "_id": 1, "name": "rectangle", "length": 11, "breadth": 10 },
 { "_id": 2, "name": "square", "length": 10, "breadth": 10 },
 { "_id": 3, "name": "rectangle", "length": 14, "breadth": 16 },
 { "_id": 4, "name": "square", "length": 6, "breadth": 6 },
 { "_id": 5, "name": "rectangle", "length": 11, "breadth": 16 }
])
```

```
db.area.aggregate([

{$project:{_id:1,name:1,length:1,breath:1,result:{$cmp:["$length","$breath"
]}}}
 ]);
```

```
> db.area.aggregate([
    {$project:{_id:1,name:1,length:1,breadth:1,result:{$cmp:["$length","$breadth"]}}}
    ]);
< {
    _id: 1,
    name: 'rectangle',
    length: 11,
    breadth: 10,
    result: 1
  }
  {
    _id: 2,
    name: 'square',
    length: 10,
    breadth: 10,
    result: 0
  }
  {
    _id: 3,
    name: 'rectangle',
    length: 14,
    breadth: 16,
    result: -1
  }
```


```
{
    {
     "_id" : 1,
     "name" : "pen",
     "data" : 11.25
    }
    {
     "_id" : 2,
     "name" : "pencil",
     "data" : 10.32
    }
    {
     "_id" : 3,
     "name" : "box",
```

"data" : 15.97
}
{
"_id" : 4,
"name" : "bottle",
"data" : -12.3
}
{
"_id" : 5,
"name" : "oval",
"data" : 20.6
}
{
"_id" : 6,
"name" : "triangle",
"data" : 5
}
{
"_id" : 7,
"name" : "bottle",
"data" : -1
}
}

Practical 13: MongoDB $concat, $size, $rename Operator

$concat

```
db.student1.aggregate([
 {$project:{_id:1,
name:{$concat:["$std_name","$last_name"]},department:1}}
 ]);
```

```
> db.student1.aggregate([
    {$project:{_id:1, name:{$concat:["$std_name","$last_name"]},department:1}}
    ]);
< {
    _id: 1,
    department: 'MCA',
    name: 'SidMane'
  }
```

$size

Collection : marks

```
db.marks.insertMany([
{"_id": 1,"name": "Jonny","class": "X","rollNo": 401,"age": 18,"marks":
[55, 60, 70, 45, 95, 68],
"extraMarks": {"practical": [21, 18, 25, 30],"attendance": [5, 9]},"gender":
"Male","bloodgroup": "A+" },
{"_id": 2,"name": "Carry","class": "IX","rollNo": 35,"age": 17,"marks": [85,
40, 90, 75, 85, 77],"gender": "Male","bloodgroup": "B+" },
{"_id": 3,"name": "Jin","class": "IX","rollNo": 49,"age": 17,"marks": [85,
70, 80, 95, 94, 81],"gender": "Female","bloodgroup": "O+" },
{"_id": 4,"name": "Thomas","class": "X","rollNo": 61,"age": 18,"marks":
[91, 65, 71, 63, 98, 76],
"extraMarks": {"practical": [26, 28, 25, 29],"attendance": [8, 8]},"gender":
"Male","bloodgroup": "A+"},
{"_id": 5,"name": "Mia","class": "IX","rollNo": 308,"age": 17,"marks": [97,
98, 95, 98],"gender": "Female","bloodgroup": "B+"},
```

{"_id": 6,"name": "Oats","class": "IX","rollNo": 75,"age": 18,"marks": [99, 98, 98, 95, 96],"gender": "Male","bloodgroup": "A+"}
])

```
db.marks.insertMany([
{"_id": 1,"name": "Jonny","class": "X","rollNo": 401,"age": 18,"marks": [55, 60, 70, 45, 95, 68],
"extraMarks": {"practical": [21, 18, 25, 30],"attendance": [5, 9]},"gender": "Male","bloodgroup": "A+" },
{"_id": 2,"name": "Carry","class": "IX","rollNo": 35,"age": 17,"marks": [85, 40, 90, 75, 85, 77],"gender": "Male","bloodgroup": "B+" },
{"_id": 3,"name": "Jin","class": "IX","rollNo": 49,"age": 17,"marks": [85, 70, 80, 95, 94, 81],"gender": "Female","bloodgroup": "O+" },
{"_id": 4,"name": "Thomas","class": "X","rollNo": 61,"age": 18,"marks": [91, 65, 71, 63, 98, 76],
"extraMarks": {"practical": [26, 28, 25, 29],"attendance": [8, 8]},"gender": "Male","bloodgroup": "A+"},
{"_id": 5,"name": "Mia","class": "IX","rollNo": 308,"age": 17,"marks": [97, 98, 95, 98],"gender": "Female","bloodgroup": "B+"},
{"_id": 6,"name": "Oats","class": "IX","rollNo": 75,"age": 18,"marks": [99, 98, 98, 95, 96],"gender": "Male","bloodgroup": "A+"}
])
```

db.marks.aggregate([{$match:{class:"IX"}},{$project:{_id:0,name:1,class:1,rollNo:1,
marks:1,gender:1,markssize:{$size:"$marks"}}}])

```
< {
    name: 'Carry',
    class: 'IX',
    rollNo: 35,
    marks: [
      85,
      40,
      90,
      75,
      85,
      77
    ],
    gender: 'Male',
    markssize: 6
  }
```

$rename
Collection shapes1

```
> db.shapes1.find()
< {
    _id: 1,
    area: 16,
    name: 'rectangle'
  }
  {
    _id: 2,
    area: 10,
    name: 'square'
  }
```

db.shapes1.updateMany({},{$rename: {"name": "shape"}})

```
> db.shapes1.updateMany({},{$rename: {"name": "shape"}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 7,
    modifiedCount: 7,
    upsertedCount: 0
  }
```

Update Many

```
  _id: ObjectId('64de0b5ce572de65422ce33a')
  std_name: "Ankita"
  gender: "Female"
  class: "VII"
  fees: 3500
  exam_fees: 500
  age: 13
  total_marks: 400
  result: "Pass"
  grade: 8.12
```

db.student.updateMany({"gender":"Female"},{$rename: {"grade": "cgpa"}})

```
> db.student.updateMany({"gender":"Female"},{$rename: {"grade": "cgpa"}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 2,
    modifiedCount: 2,
    upsertedCount: 0
  }
```

```
> db.student.find({"gender":"Female"})
< {
    _id: ObjectId("64de0b5ce572de65422ce339"),
    std_name: 'Aarti',
    gender: 'Female',
    class: 'IX',
    fees: 4000,
    exam_fees: 500,
    age: 15,
    total_marks: 401,
    result: 'Pass',
    cgpa: Decimal128("7.1")
  }
  {
    _id: ObjectId("64de0b5ce572de65422ce33a"),
    std_name: 'Ankita',
    gender: 'Female',
    class: 'VII',
    fees: 3500,
    exam_fees: 500,
    age: 13,
    total_marks: 400,
    result: 'Pass',
    cgpa: Decimal128("8.12")
  }
```

```
{
 {
  "_id" : 1,
  "name" : "Steve",
  "surname" : "Smith",
  "department" : "B-tech",
  "fees" : 80000
 }
 {
  "_id" : 2,
  "name" : "Sandy",
```

"surname" : "Beach",
"department" : "BCA",
"fees" : 55000
}
{
"_id" : 3,
"name" : "John",
"surname" : "Cena",
"department" : "MCA",
"fees" : 85000
}
{
"_id" : 4,
"name" : "Wick",
"surname" : "John",
"department" : "B.com",
"fees" : 60000
}
{
"_id" : 5,
"name" : "David",
"surname" : "Silva",
"department" : "null",
"fees" : 80000
}
}

MongoDB $size Operator

{
"_id" : 1,
"name" : "Jonny",
"class" : "X",
"rollNo" : 401,
"age" : 18,
"marks" : [55, 60, 70, 45, 95, 68],
"extraMarks" : {
"practical" : [21, 18, 25, 30],

"attendance" : [5, 9]
}
"gender" : "Male",
"bloodgroup" : "A+"
}
{
"_id" : 2,
"name" : "Carry",
"class" : "IX",
"rollNo" : 35,
"age" : 17,
"marks" : [85, 40, 90, 75, 85, 77],
"gender" : "Male",
"bloodgroup" : "B+"
}
{
"_id" : 3,
"name" : "Jin",
"class" : "IX",
"rollNo" : 49,
"age" : 17,
"marks" : [85, 70, 80, 95, 94, 81],
"gender" : "Female",
"bloodgroup" : "O+"
}
{
"_id" : 4,

```
  "name" : "Thomas",
  "class" : "X",
  "rollNo" : 61,
  "age" : 18,
  "marks" : [ 91, 65, 71, 63, 98, 76 ],
  "extraMarks" : {
              "practical" : [ 26, 28, 25, 29 ],
              "attendance" : [ 8, 8 ]
            }
  "gender" : "Male",
  "bloodgroup" : "A+"
}
{
  "_id" : 5,
  "name" : "Mia",
  "class" : "IX",
  "rollNo" : 308,
  "age" : 17,
  "marks" : [ 97, 98, 95, 98 ],
  "gender" : "Female",
  "bloodgroup" : "B+"
}
{
  "_id" : 6,
  "name" : "Oats",
  "class" : "IX",
  "rollNo" : 75,
```

"age" : 18,
"marks" : [99, 98, 98, 95, 96],
"gender" : "Male",
"bloodgroup" : "A+"
}

*****End*****