Experiment 4

Association Rule Mining on Employee Dataset

Aim: To create an employee.arff dataset and demonstrate Association rule process on it using apriori algorithm

Tasks:

1. Create employee.arff dataset and load it into Weka.
2. Apply Apriori algorithm with default parameters.
3. Change the parameters and observe the results

Task 1: Create employee.arff dataset and load it.

Create employee.arff with following categorical attributes and load it in to Weka.

Attribute	States
Designation	i. Manager ii. Developer iii. Tester
Beneficiary	i. Yes ii. No
GPF	i. Yes ii. No
Salary	i. Low ii. Medium iii. High
CreditRating	i. Poor ii. Fair iii. Excellent
BankLoan	i. Yes ii. No

Task 2: Apply Apriori algorithm with default parameters

Association Rule Mining is a process that finds features which occur together or features that are correlated. Popular applications are Market Basket Analysis and Cross Marketing.

Association rules are mined out after frequent itemsets in a big dataset which can be found using algorithms such as Apriori and FP Growth.

Frequent Itemset mining mines data using support and confidence measures.

$$support\ (A \Rightarrow B) = \frac{number\ of\ instances\ containing\ both\ A\ and\ B}{total\ number\ of\ instances}$$

$$confidence\ (A \Rightarrow B) = p(B/A) = \frac{number\ of\ instances\ containing\ both\ A\ and\ B}{number\ of\ instances\ containing\ A}$$

Apriori Rule Learner in Weka implements Apriori algorithm. It iteratively reduces the minimum support from its upperBound until (i) it finds the required number of rules or the minimum support reaches lowerBound.

Default values of the some important parameters:

lowerBoundMinSupport = 0.1 (10%)

upperBoundMinSupport = 1.0 (100%)

metricType = Confidence

minMetric = 0.9 (90%)

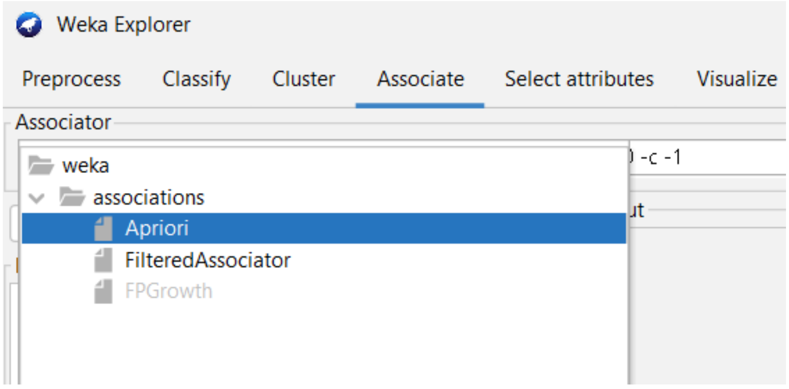
numRules = 10

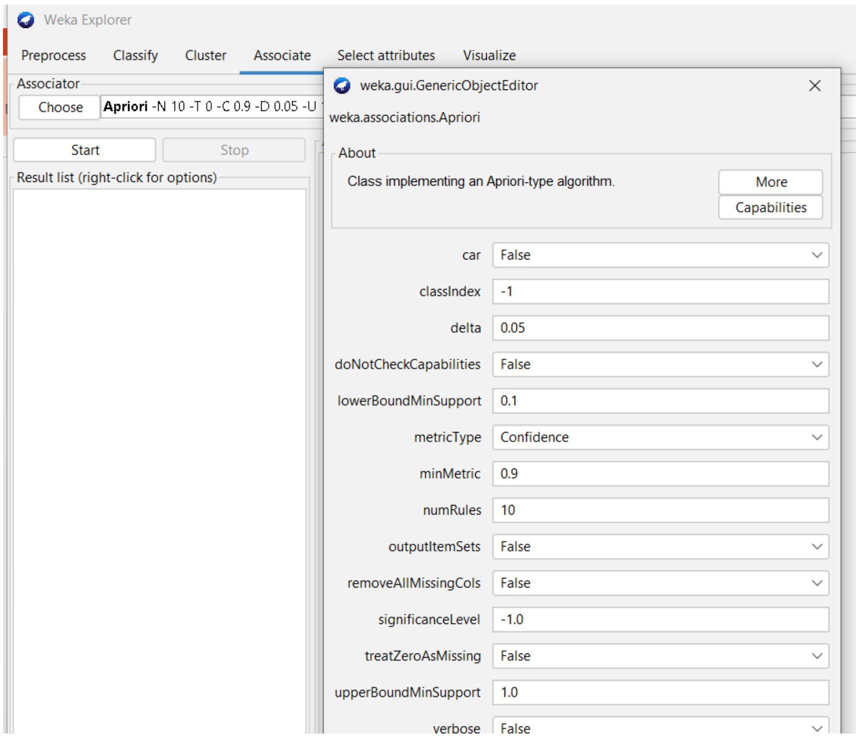
Steps:

I. Select Associate → Choose → associations → Apriori.

II. Observe the default parameter values.

III. Click on Start.





Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U

weka.associations.Apriori

Class implementing an Apriori-type algorithm.

car	False
classIndex	-1
delta	0.05
doNotCheckCapabilities	False
lowerBoundMinSupport	0.1
metricType	Confidence
minMetric	0.9
numRules	10
outputItemSets	False
removeAllMissingCols	False
significanceLevel	-1.0
treatZeroAsMissing	False
upperBoundMinSupport	1.0
verbose	False

Observations:

Default Parameters	Observations
lowerBoundMinSupport = upperBoundMinSupport = metricType = minMetric = numRules =	Minimum support = Minimum Metric <Confidence> = Number of cycles performed = Best rules found with confidence: 1. 2. 3. 4. 5. 6. 7. 8. 9. 10 .

Task 3: Apply Apriori algorithm with required parameters

Observations: Change the default parameter values and perform the experiment

Parameters	**Observations**
lowerBoundMinSupport = upperBoundMinSupport = metricType = minMetric = numRules =	Minimum support = Minimum Metric <Confidence> = Number of cycles performed = Best rules found with confidence:

Conclusion:

Experiment 5

Classification Using J48

Aim: To demonstrate Classification process on iris.arff dataset using j48 algorithm with percentage split.

Tasks:

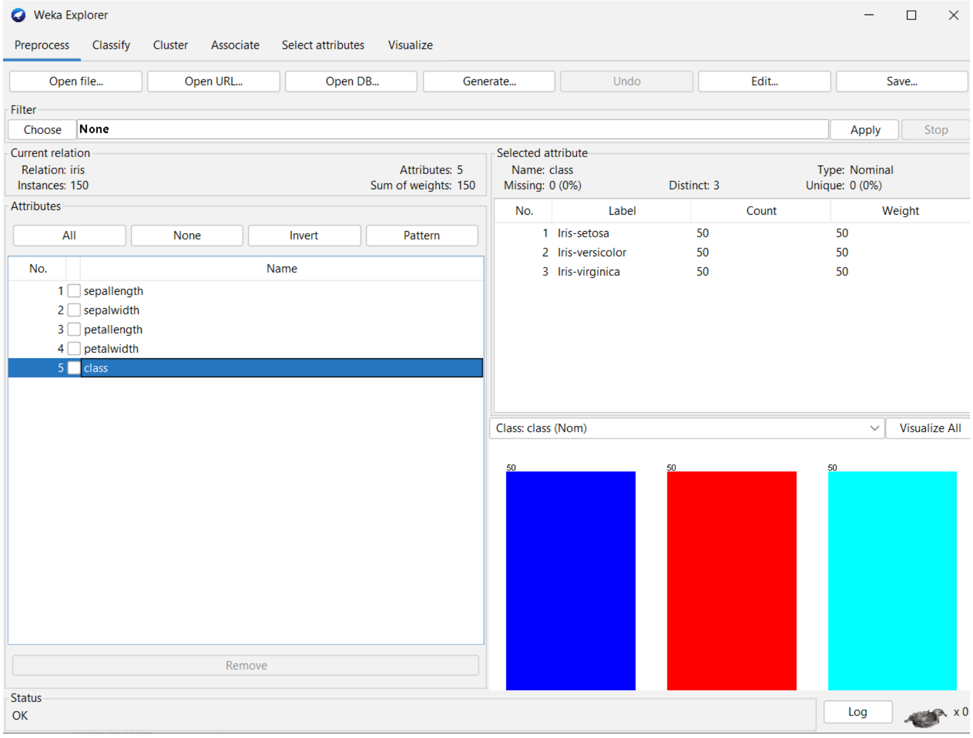
1. Load iris.arff dataset and explore it.
2. Build a classification model using J48 algorithm with percentage split.
3. Make predictions on new data.

Task 1: Load iris.arff dataset and explore it.

Load iris.arff from the Weka's data folder.

Observations:

Attribute	Type	Range / States
Number of Instances:		



Filter: None

Current relation
Relation: iris
Instances: 150
Attributes: 5
Sum of weights: 150

Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

Selected attribute
Name: class
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	Iris-setosa	50	50
2	Iris-versicolor	50	50
3	Iris-virginica	50	50

Class: class (Nom)

Status
OK

Task 2: Build a classification model using J48 algorithm with percentage split.

Classification is a process of determining the class (state) of the given instance. Examples:

Determining Play or Not play based on weather conditions.

Determining the digit (0 – 9) given the image pixel data.

Determining the Spam or Not-spam based on mail text.

J48 is Weka’s Java implementation of the C4.5 algorithm. It can generate pruned or unpruned tress with both nominal and numerical attributes for classification.

Percentage Split:

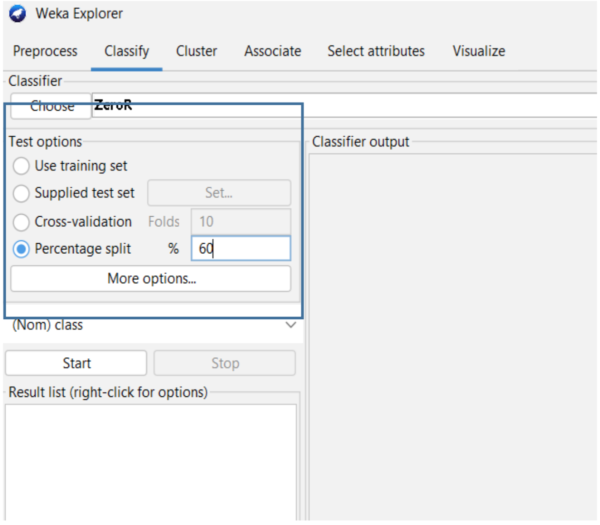
Splits the data into training and test subsets.

Training set is used to build the model.

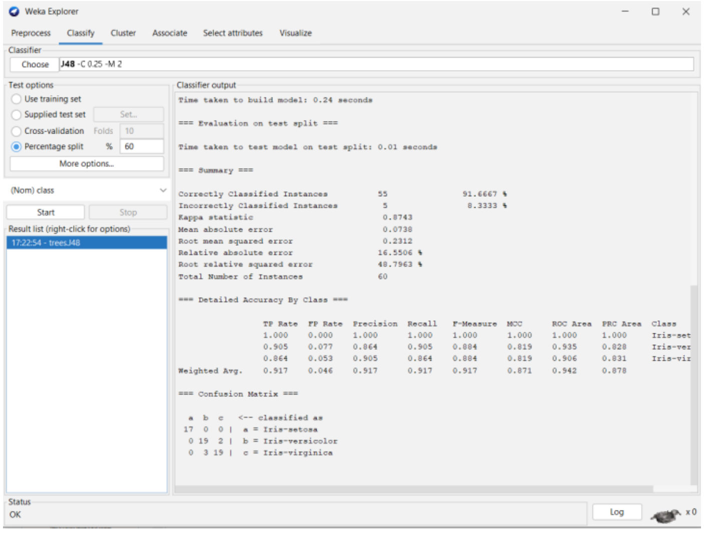
Test set is used to evaluate the model performance.

Steps to build the model:

1. Click on Classify and select Percentage split with required training percentage under Test options group.
2. Select Choose → classifiers → trees → J48.
3. Clock on Start.
4. Right click on the model and click on Visualize tree



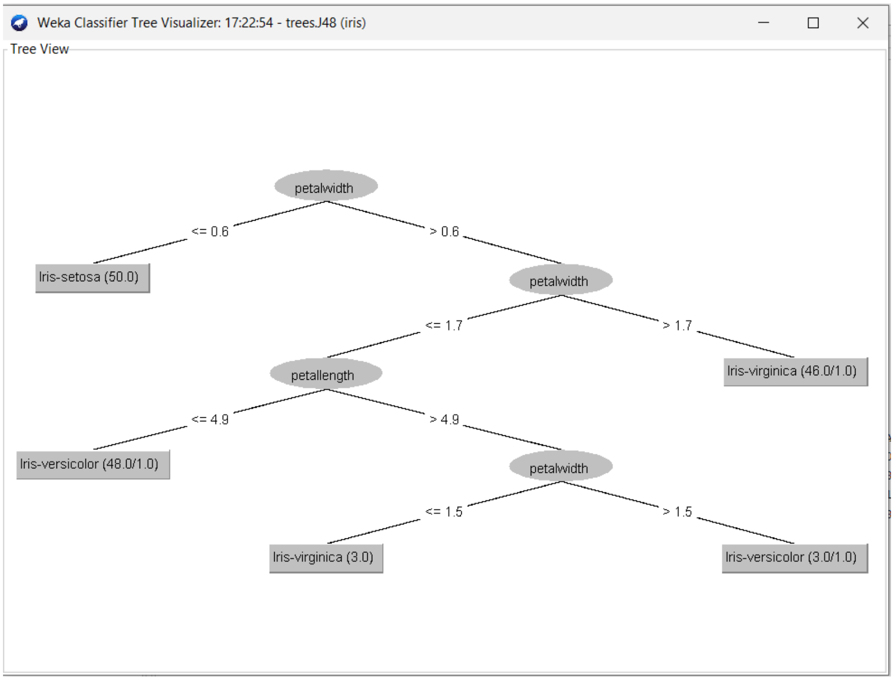
○ Use training set
○ Supplied test set
○ Cross-validation Folds 10
● Percentage split % 60



Classifier: J48 -C 0.25 -M 2

● Percentage split % 60

```
Time taken to build model: 0.24 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.01 seconds

=== Summary ===

Correctly Classified Instances          55               91.6667 %
Incorrectly Classified Instances         5                8.3333 %
Kappa statistic                          0.8743
Mean absolute error                      0.0738
Root mean squared error                  0.2312
Relative absolute error                 16.5506 %
Root relative squared error             48.7963 %
Total Number of Instances               60

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-set
                 0.905    0.077    0.864      0.905    0.884      0.819    0.935     0.828     Iris-ver
                 0.864    0.053    0.905      0.864    0.884      0.819    0.906     0.831     Iris-vir
Weighted Avg.    0.917    0.046    0.917      0.917    0.917      0.871    0.942     0.878

=== Confusion Matrix ===

  a  b  c   <-- classified as
 17  0  0 |  a = Iris-setosa
  0 19  2 |  b = Iris-versicolor
  0  3 19 |  c = Iris-virginica
```


Weka Classifier Tree Visualizer: 17:22:54 - trees.J48 (iris)
Tree View
petalwidth
<= 0.6
> 0.6
Iris-setosa (50.0)
petalwidth
<= 1.7
> 1.7
petallength
Iris-virginica (46.0/1.0)
<= 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)

Observations:

- Total number of instances:
- Correctly classified instances:
- Incorrectly classified instances:
- Accuracy:
- Calculation of Accuracy from Confusion Matrix:

Task 3: Make predictions on new data

Steps:

i. Create an ARFF file with unlabeled (use ? in the place of class label) instances.

ii. On the “Classify” tab, select the “Supplied test set” option in the “Test options” pane.

iii. Click the “Set” button, click the “Open file” button on the options window and select the new dataset.

iv. Click the “More options…” button and for the “Output predictions” option click the “Choose” button and select “PlainText”.

v. Right click on the model in the “Results list” pane and Select “Re-evaluate model on current test set”.

Observations:

Instance No.	Predicted class

Conclusion:

Experiment 6

Classification Using J48

Aim: To demonstrate Classification process on StudentResult.arff dataset using j48 algorithm with cross-validation

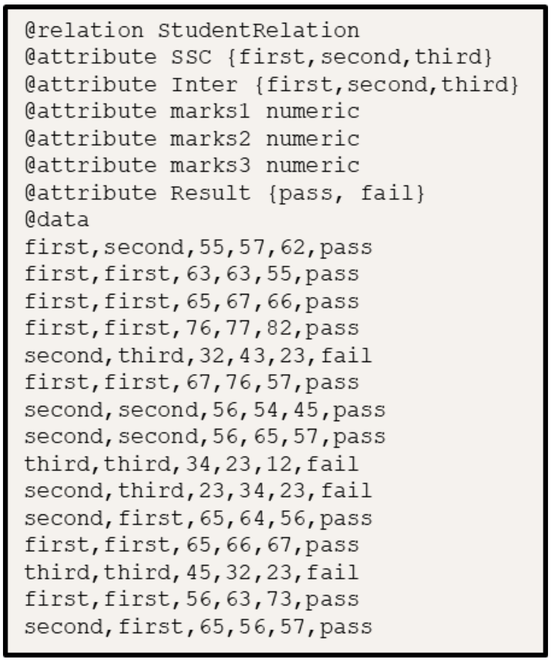
Tasks:

1. Create StudentResult.arff dataset and load it into Weka.
2. Build a classification model using J48 algorithm with k-fold cross validation.
3. Make predictions on new data.

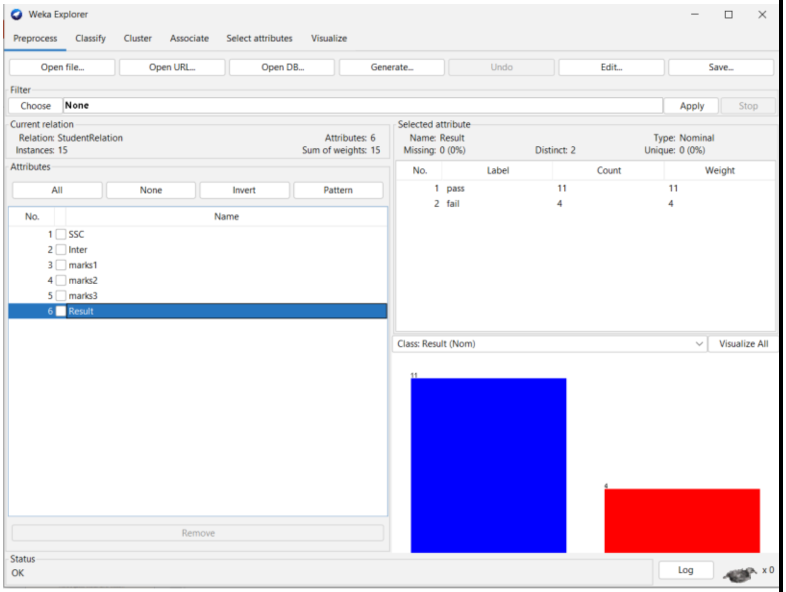
Task 1: Create StudentResult.arff dataset and load it into Weka.

Create StudentResults.arff with following attributes and load it into Weka

Attribute	Type
SSC	Nominal with states First, Second, Third.
Inter	Nominal with states First, Second, Third.
Marks1	Numeric
Marks2	Numeric
Marks3	Numeric
Result	Nominal with states Pass & Fail.



```
@relation StudentRelation
@attribute SSC {first,second,third}
@attribute Inter {first,second,third}
@attribute marks1 numeric
@attribute marks2 numeric
@attribute marks3 numeric
@attribute Result {pass, fail}
@data
first,second,55,57,62,pass
first,first,63,63,55,pass
first,first,65,67,66,pass
first,first,76,77,82,pass
second,third,32,43,23,fail
first,first,67,76,57,pass
second,second,56,54,45,pass
second,second,56,65,57,pass
third,third,34,23,12,fail
second,third,23,34,23,fail
second,first,65,64,56,pass
first,first,65,66,67,pass
third,third,45,32,23,fail
first,first,56,63,73,pass
second,first,65,56,57,pass
```



Filter: None

Current relation
Relation: StudentRelation
Instances: 15
Attributes: 6
Sum of weights: 15

Selected attribute
Name: Result
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	pass	11	11
2	fail	4	4

Attributes

No.		Name
1	□	SSC
2	□	Inter
3	□	marks1
4	□	marks2
5	□	marks3
6	□	Result

Class: Result (Nom)

Status
OK

Task 2: Build a classification model using J48 algorithm with k-fold cross validation

Classification is a process of determining the class (state) of the given instance. Examples:

Determining Play or Not play based on weather conditions.

Determining the digit (0 – 9) given the image pixel data.

Determining the Spam or Not-spam based on mail text.

J48 is Weka's Java implementation of the C4.5 algorithm. It can generate pruned or unpruned tress with both nominal and numerical attributes for classification.

K-fold Cross Validation is a resampling procedure used to evaluate data mining models on a limited data set. It's process is

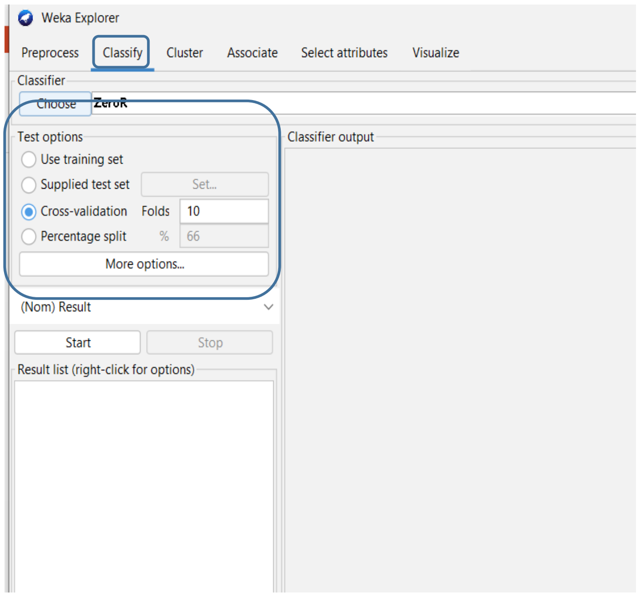
I. Split the input dataset into K groups

II. For i from 1 to k

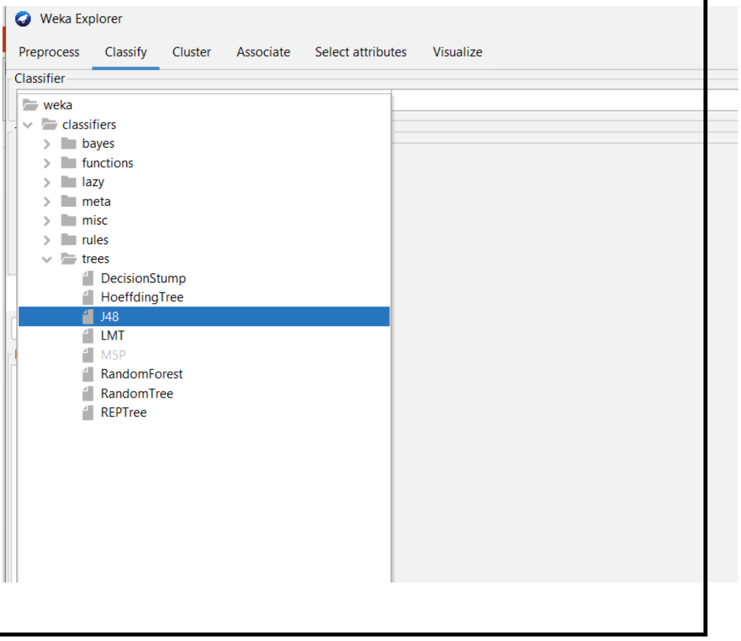
- Take ith group as test dataset.
- Use remaining K-1 groups as training dataset.
- Fit the model using training set and evaluate its performance on test set.

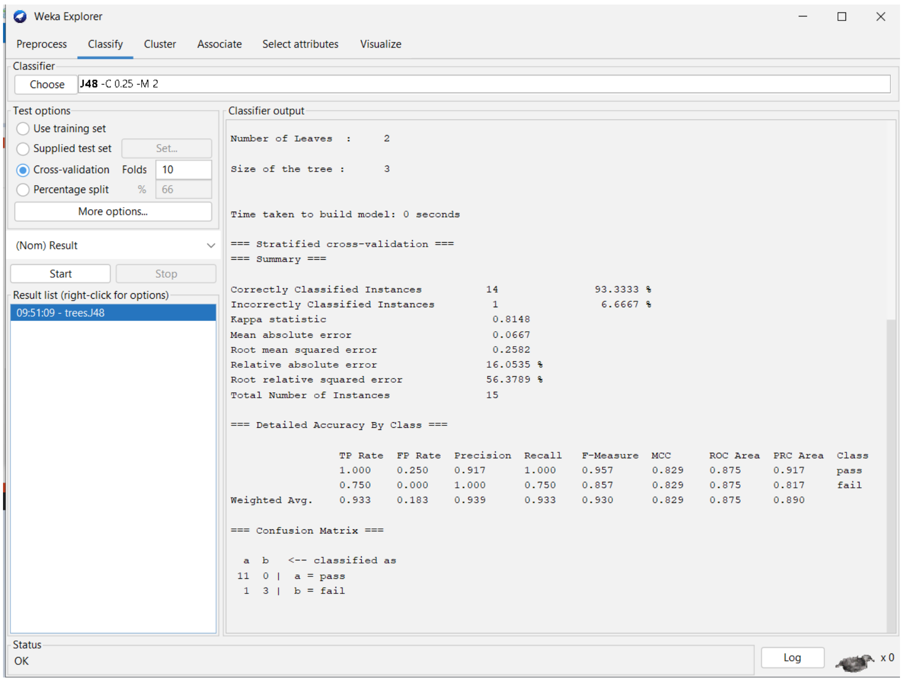
Steps to build the model:

1. Click on Classify and select Cross-validation with default 10 folds under Test options group.
2. Select Choose → classifiers → trees → J48.
3. Clock on Start.
4. Right click on the model and click on Visualize tree



Cross-validation Folds 10





Classifier

J48 -C 0.25 -M 2

Test options

○ Use training set
○ Supplied test set Set...
● Cross-validation Folds 10
○ Percentage split % 66

(Nom) Result

Result list (right-click for options)

09:51:09 - trees.J48

Classifier output

```
Number of Leaves  :     2

Size of the tree :      3


Time taken to build model: 0 seconds

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

Correctly Classified Instances          14               93.3333 %
Incorrectly Classified Instances         1                6.6667 %
Kappa statistic                          0.8148
Mean absolute error                      0.0667
Root mean squared error                  0.2582
Relative absolute error                 16.0535 %
Root relative squared error             56.3789 %
Total Number of Instances               15

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.250    0.917      1.000    0.957      0.829    0.875     0.917     pass
                 0.750    0.000    1.000      0.750    0.857      0.829    0.875     0.817     fail
Weighted Avg.    0.933    0.183    0.939      0.933    0.930      0.829    0.875     0.890

=== Confusion Matrix ===

  a  b   <-- classified as
 11  0 |  a = pass
  1  3 |  b = fail
```

Status

OK

Tree generated by J48 algorithm for the given dataset is:

Observations:

- Total number of instances:
- Correctly classified instances:
- Incorrectly classified instances:
- Accuracy:
- Calculation of Accuracy from Confusion Matrix:

Task 3: Make predictions on new data

Steps:

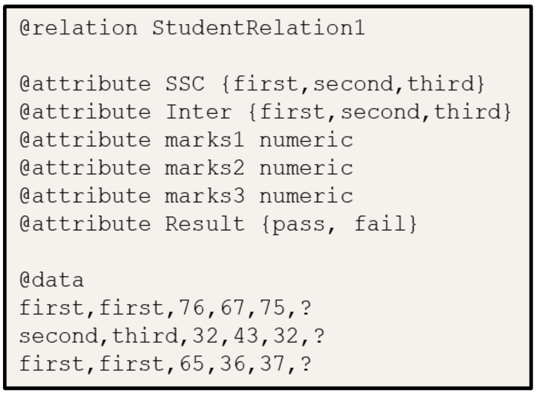
i. Create an ARFF file with unlabeled (use ? in the place of class label) instances.

ii. On the “Classify” tab, select the “Supplied test set” option in the “Test options” pane.

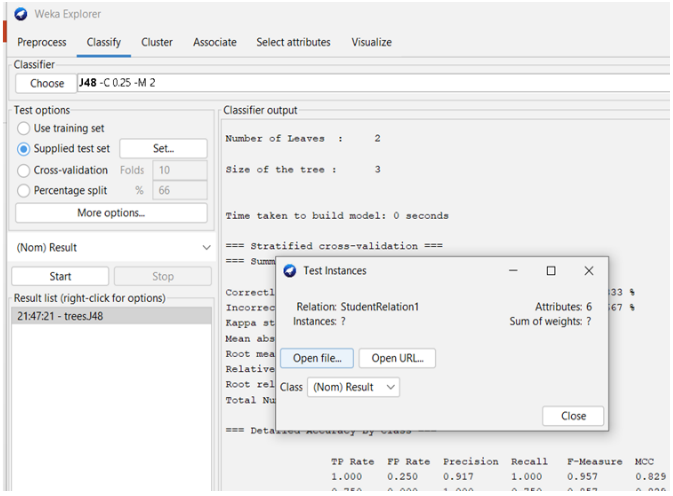
iii. Click the “Set” button, click the “Open file” button on the options window and select the new dataset.

iv. Click the “More options…” button and for the “Output predictions” option click the “Choose” button and select “PlainText”.

v. Right click on the model in the “Results list” pane and Select “Re-evaluate model on current test set”.



```
@relation StudentRelation1

@attribute SSC {first,second,third}
@attribute Inter {first,second,third}
@attribute marks1 numeric
@attribute marks2 numeric
@attribute marks3 numeric
@attribute Result {pass, fail}

@data
first,first,76,67,75,?
second,third,32,43,32,?
first,first,65,36,37,?
```



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...

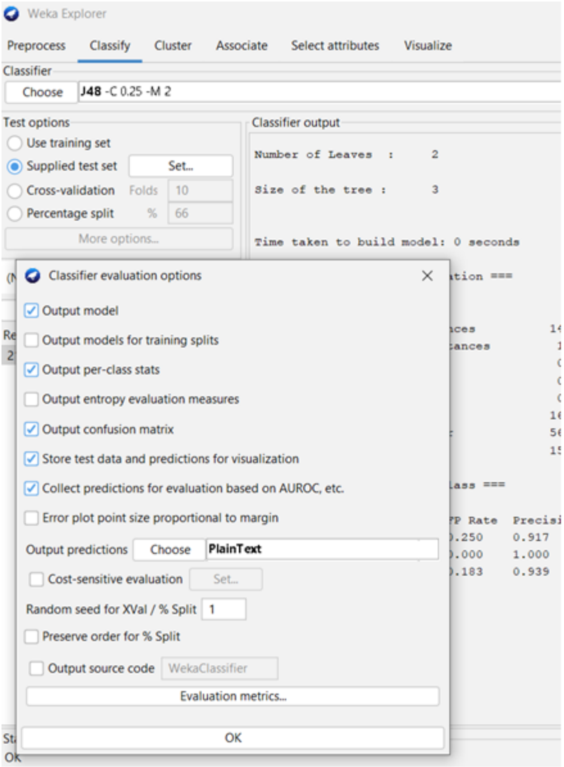
(Nom) Result

Result list (right-click for options): 21:47:21 - trees.J48

Relation: StudentRelation1 Attributes: 6
Instances: ? Sum of weights: ?
Class (Nom) Result

Classifier output

```
Number of Leaves  :     2

Size of the tree :      3


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summ

Correctl                                            33 %
Incorrec                                            67 %
Kappa st
Mean abs
Root mea
Relative
Root rel
Total Nu

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC
                 1.000    0.250    0.917      1.000    0.957      0.829
```



Classifier: Choose J48 -C 0.25 -M 2

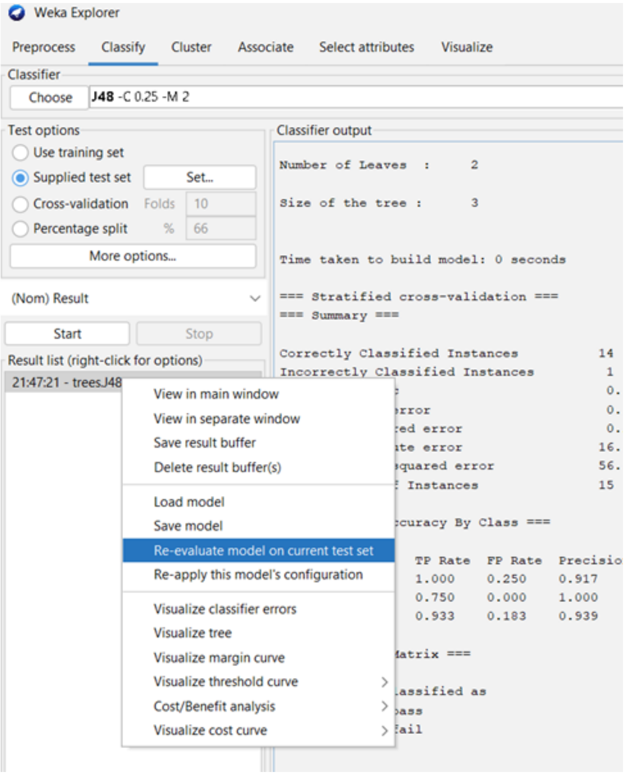
Test options: ○ Use training set ● Supplied test set (Set...) ○ Cross-validation Folds 10 ○ Percentage split % 66

Classifier evaluation options

☑ Output model
☐ Output models for training splits
☑ Output per-class stats
☐ Output entropy evaluation measures
☑ Output confusion matrix
☑ Store test data and predictions for visualization
☑ Collect predictions for evaluation based on AUROC, etc.
☐ Error plot point size proportional to margin
Output predictions Choose PlainText
☐ Cost-sensitive evaluation Set...
Random seed for XVal / % Split 1
☐ Preserve order for % Split
☐ Output source code WekaClassifier

Classifier output

```
Number of Leaves  :     2

Size of the tree :      3


Time taken to build model: 0 seconds
```



Classifier: Choose J48 -C 0.25 -M 2

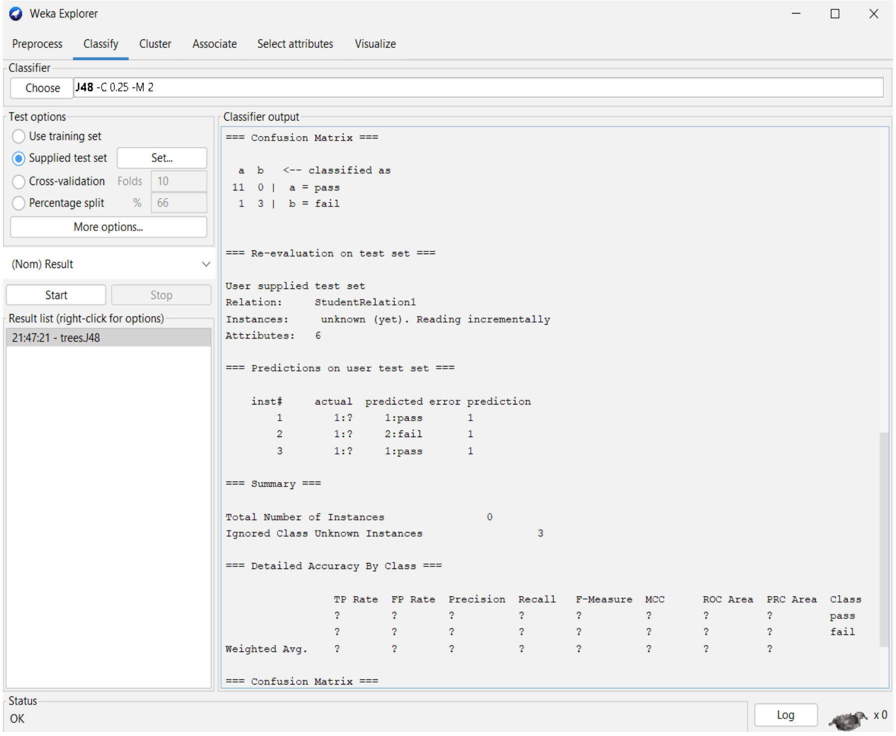
Test options: ○ Use training set ● Supplied test set (Set...) ○ Cross-validation Folds 10 ○ Percentage split % 66

(Nom) Result

Result list (right-click for options): 21:47:21 - trees.J48

Classifier output

```
Number of Leaves  :     2

Size of the tree :      3


Time taken to build model: 0 seconds

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

Correctly Classified Instances          14
Incorrectly Classified Instances         1
                                         0.
rror                                     0.
ed error                                 0.
te error                                16.
quared error                            56.
 Instances                              15

curacy By Class ===

   TP Rate  FP Rate  Precisio
   1.000    0.250    0.917
   0.750    0.000    1.000
   0.933    0.183    0.939

Matrix ===

lassified as
pass
fail
```



Classifier: J48 -C 0.25 -M 2

Folds 10

% 66

21:47:21 - trees.J48

```
=== Confusion Matrix ===


  a  b   <-- classified as
 11  0 |  a = pass
  1  3 |  b = fail



=== Re-evaluation on test set ===

User supplied test set
Relation:     StudentRelation1
Instances:     unknown (yet). Reading incrementally
Attributes:   6

=== Predictions on user test set ===

    inst#     actual  predicted error prediction
        1        1:?     1:pass       1
        2        1:?     2:fail       1
        3        1:?     1:pass       1

=== Summary ===

Total Number of Instances                0
Ignored Class Unknown Instances                  3

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 ?        ?        ?          ?        ?          ?        ?         ?         pass
                 ?        ?        ?          ?        ?          ?        ?         ?         fail
Weighted Avg.    ?        ?        ?          ?        ?          ?        ?         ?

=== Confusion Matrix ===
```

Status: OK

Observations:

Instance No.	Predicted class

Conclusion:

Experiment 7

Classification Using ID3

Aim: To demonstrate Classification process on contact-lenses.arff dataset using ID3 algorithm with cross-validation.

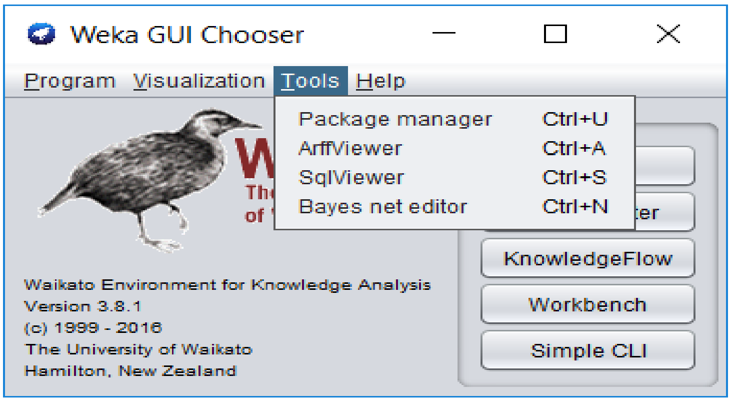
Tasks:

1. Install required package for ID3
2. Load contact-lenses.arff dataset
3. Build a classification model using ID3 algorithm with k-fold cross-validation.
4. Make predictions on new data.

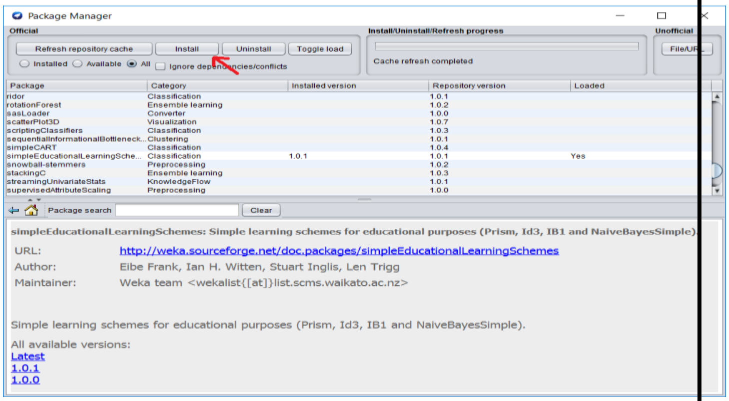
Task 1: Install required package for ID3

Select GUI Chooser → Tools → Package Manager

Search, select & install *simpleEducationalLearningSchemes* package to get ID3 classifier under trees group.



Waikato Environment for Knowledge Analysis
Version 3.8.1
(c) 1999 - 2016
The University of Waikato
Hamilton, New Zealand



Cache refresh completed

Package	Category	Installed version	Repository version	Loaded
ridor	Classification		1.0.1	
rotationForest	Ensemble learning		1.0.2	
sasLoader	Converter		1.0.0	
scatterPlot3D	Visualization		1.0.7	
scriptingClassifiers	Classification		1.0.3	
sequentialInformationalBottleneck...	Clustering		1.0.1	
simpleCART	Classification		1.0.4	
simpleEducationalLearningSche...	Classification	1.0.1	1.0.1	Yes
snowball-stemmers	Preprocessing		1.0.2	
stackingC	Ensemble learning		1.0.3	
streamingUnivariateStats	KnowledgeFlow		1.0.1	
supervisedAttributeScaling	Preprocessing		1.0.0	

simpleEducationalLearningSchemes: Simple learning schemes for educational purposes (Prism, Id3, IB1 and NaiveBayesSimple).

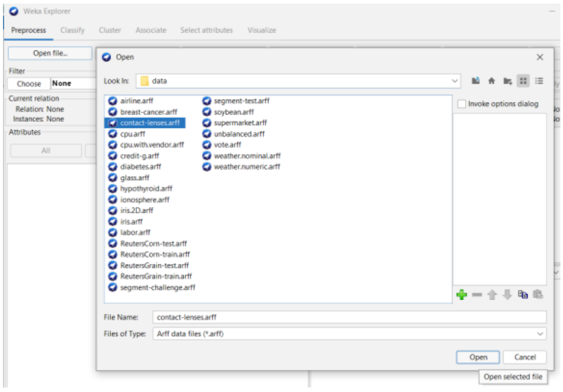
URL: http://weka.sourceforge.net/doc.packages/simpleEducationalLearningSchemes
Author: Eibe Frank, Ian H. Witten, Stuart Inglis, Len Trigg
Maintainer: Weka team <wekalist{[at]}list.scms.waikato.ac.nz>

Simple learning schemes for educational purposes (Prism, Id3, IB1 and NaiveBayesSimple).

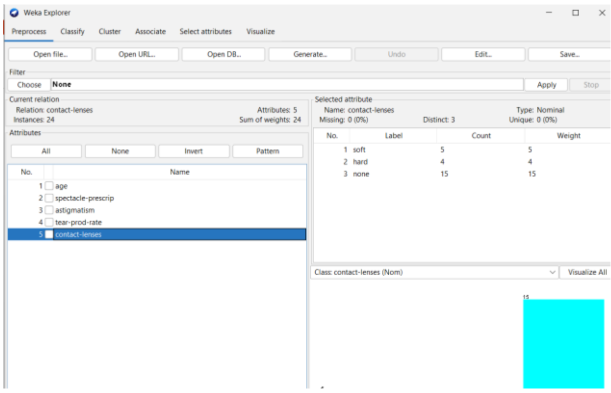
All available versions:
Latest
1.0.1
1.0.0

Task 2: Load contact-lenses.arff dataset

Load contact-lenses.arff from the Weka's data folder.



File Name: contact-lenses.arff
Files of Type: Arff data files (*.arff)



Relation: contact-lenses
Instances: 24
Attributes: 5
Sum of weights: 24

No.	Name
1	age
2	spectacle-prescrip
3	astigmatism
4	tear-prod-rate
5	contact-lenses

Name: contact-lenses
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	soft	5	5
2	hard	4	4
3	none	15	15

Class: contact-lenses (Nom)

Task 3: Build a classification model using ID3 algorithm with k-fold cross validation.

Classification is a process of determining the class (state) of the given instance.

ID3 stands for Iterative Dichotomiser 3 and is named such because the algorithm iteratively (repeatedly) dichotomizes(divides) features into two or more groups at each step. ID3 uses a top-down greedy approach to build a decision tree. ID3 is only used for classification problems with nominal features only.

K-fold Cross Validation is a resampling procedure used to evaluate data mining models on a limited data set. It's process is

I. Split the input dataset into K groups

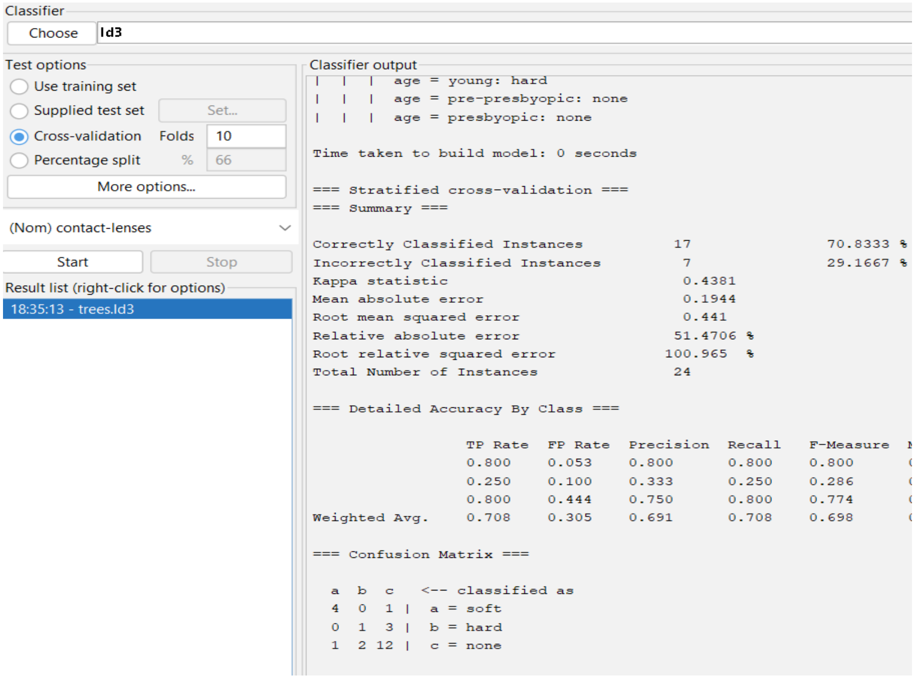
II. For i from 1 to k

- Take ith group as test dataset.
- Use remaining K-1 groups as training dataset.
- Fit the model using training set and evaluate its performance on test set.

Steps to build the model:

1. Click on Classify and select Cross-validation with some number folds under Test options group.
2. Select Choose → classifiers → trees → ID3.
3. Clock on Start.

Note that we can't visualize ID3 tree.



Classifier: Id3

Test options: ○ Use training set ○ Supplied test set ● Cross-validation Folds 10 ○ Percentage split % 66

(Nom) contact-lenses

18:35:13 - trees.Id3

```
|   |   |   age = young: hard
|   |   |   age = pre-presbyopic: none
|   |   |   age = presbyopic: none

Time taken to build model: 0 seconds

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

Correctly Classified Instances          17               70.8333 %
Incorrectly Classified Instances         7               29.1667 %
Kappa statistic                          0.4381
Mean absolute error                      0.1944
Root mean squared error                  0.441
Relative absolute error                 51.4706 %
Root relative squared error            100.965  %
Total Number of Instances               24

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure
                 0.800    0.053    0.800      0.800    0.800
                 0.250    0.100    0.333      0.250    0.286
                 0.800    0.444    0.750      0.800    0.774
Weighted Avg.    0.708    0.305    0.691      0.708    0.698

=== Confusion Matrix ===

  a  b  c   <-- classified as
  4  0  1 |  a = soft
  0  1  3 |  b = hard
  1  2 12 |  c = none
```

Observations:

- Total number of instances:
- Correctly classified instances:
- Incorrectly classified instances:
- Accuracy:
- Calculation of Accuracy from Confusion Matrix:

Task 4: Make predictions on new data

Steps:

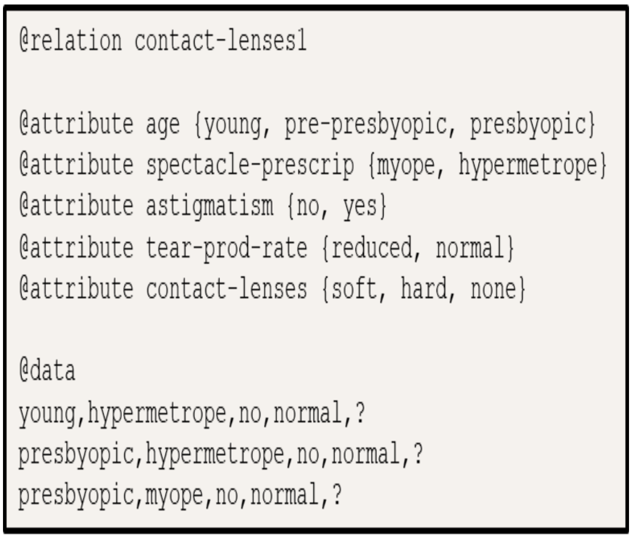
i. Create an ARFF file with unlabeled (use ? in the place of class label) instances.

ii. On the “Classify” tab, select the “Supplied test set” option in the “Test options” pane.

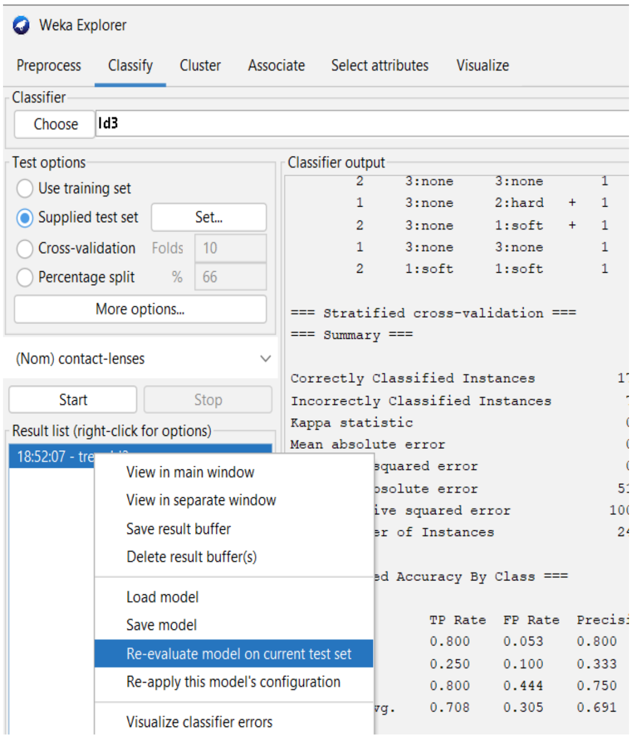
iii. Click the “Set” button, click the “Open file” button on the options window and select the new dataset.

iv. Click the “More options…” button and for the “Output predictions” option click the “Choose” button and select “PlainText”.

v. Right click on the model in the “Results list” pane and Select “Re-evaluate model on current test set”.



```
@relation contact-lenses1

@attribute age {young, pre-presbyopic, presbyopic}
@attribute spectacle-prescrip {myope, hypermetrope}
@attribute astigmatism {no, yes}
@attribute tear-prod-rate {reduced, normal}
@attribute contact-lenses {soft, hard, none}

@data
young,hypermetrope,no,normal,?
presbyopic,hypermetrope,no,normal,?
presbyopic,myope,no,normal,?
```



Classifier: Id3

Folds: 10

Percentage split %: 66

(Nom) contact-lenses

```
          2      3:none      3:none         1
          1      3:none      2:hard    +    1
          2      3:none      1:soft    +    1
          1      3:none      3:none         1
          2      1:soft      1:soft         1

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

Correctly Classified Instances              17
Incorrectly Classified Instances
Kappa statistic
Mean absolute error
          squared error
          osolute error                     51
          ive squared error                100
          er of Instances                   24

          ed Accuracy By Class ===

                 TP Rate   FP Rate   Precis
                 0.800     0.053     0.800
                 0.250     0.100     0.333
                 0.800     0.444     0.750
          vg.    0.708     0.305     0.691
```

Observations:

Instance No.	Predicted class

Conclusion: