EX.NO:29
Date:

EVALUATING ACCURACY OF THE CLASSIFIERS

AIM:

To create evaluating accuracy of the classifiers using weka tool.

DESCRIPTION:

Consider the german credit dataset which can be downloaded from the UCI repository.

PROCEDURE:

1.Download WEKA And Install
2.Start WEKA
3.Open The Data/iris.arff Dataset
4.Select And Run An Algorithm
5.Review The Results

ANALYSIS :

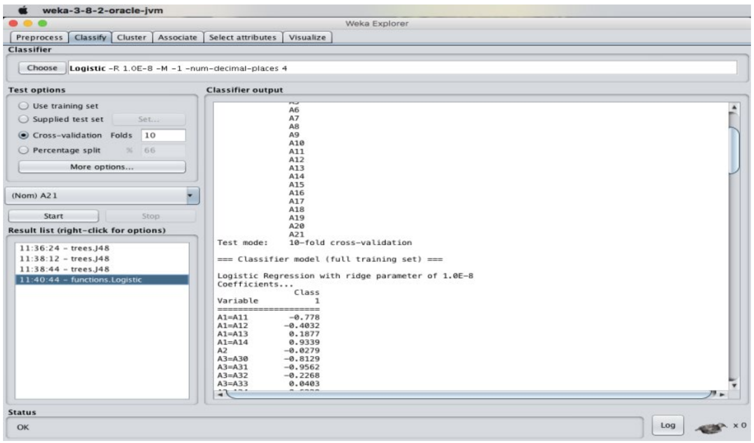
A) Logistic Regression :

Logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical).

Steps :

- Load the dataset into the weka tool and preprocess it.
- Apply the classification the logistic regression technique and execute for the result.

Output :



Classifier: Logistic -R 1.0E-8 -M -1 -num-decimal-places 4

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

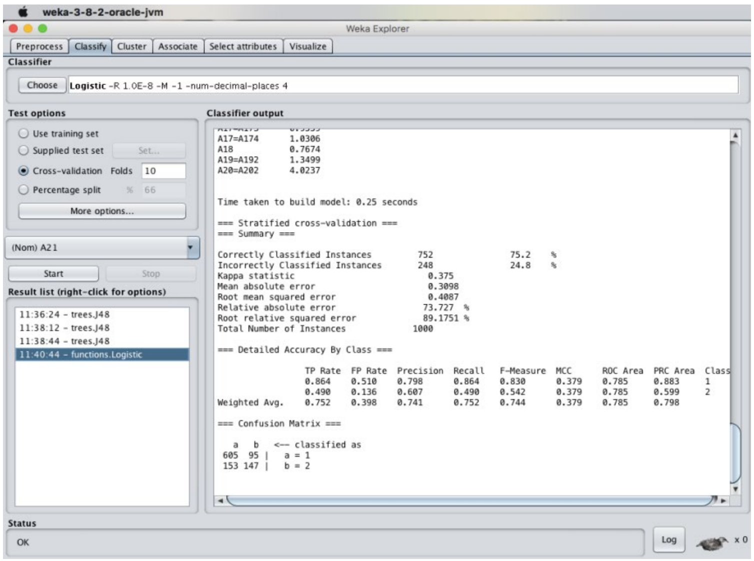
(Nom) A21

Result list (right-click for options)
- 11:36:24 - trees.J48
- 11:38:12 - trees.J48
- 11:38:44 - trees.J48
- 11:40:44 - functions.Logistic

Classifier output

```
              A6
              A7
              A8
              A9
              A10
              A11
              A12
              A13
              A14
              A15
              A16
              A17
              A18
              A19
              A20
              A21
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Logistic Regression with ridge parameter of 1.0E-8
Coefficients...
                  Class
Variable              1
=====================
A1=A11           -0.778
A1=A12          -0.4032
A1=A13           0.1877
A1=A14           0.9339
A2              -0.0279
A3=A30          -0.8129
A3=A31          -0.9562
A3=A32          -0.2268
A3=A33           0.0403
```

Status
OK



Classifier: Logistic -R 1.0E-8 -M -1 -num-decimal-places 4

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

(Nom) A21

Result list (right-click for options)
- 11:36:24 - trees.J48
- 11:38:12 - trees.J48
- 11:38:44 - trees.J48
- 11:40:44 - functions.Logistic

Classifier output

```
A17=A174      1.0306
A18           0.7674
A19=A192      1.3499
A20=A202      4.0237


Time taken to build model: 0.25 seconds

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

Correctly Classified Instances         752               75.2    %
Incorrectly Classified Instances       248               24.8    %
Kappa statistic                          0.375
Mean absolute error                      0.3098
Root mean squared error                  0.4087
Relative absolute error                 73.727  %
Root relative squared error             89.1751 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.864    0.510    0.798      0.864    0.830      0.379    0.785     0.883     1
                 0.490    0.136    0.607      0.490    0.542      0.379    0.785     0.599     2
Weighted Avg.    0.752    0.398    0.741      0.752    0.744      0.379    0.785     0.798

=== Confusion Matrix ===

   a   b   <-- classified as
 605  95 |   a = 1
 153 147 |   b = 2
```

Status
OK

B) Naïve Bayes Algorithm :

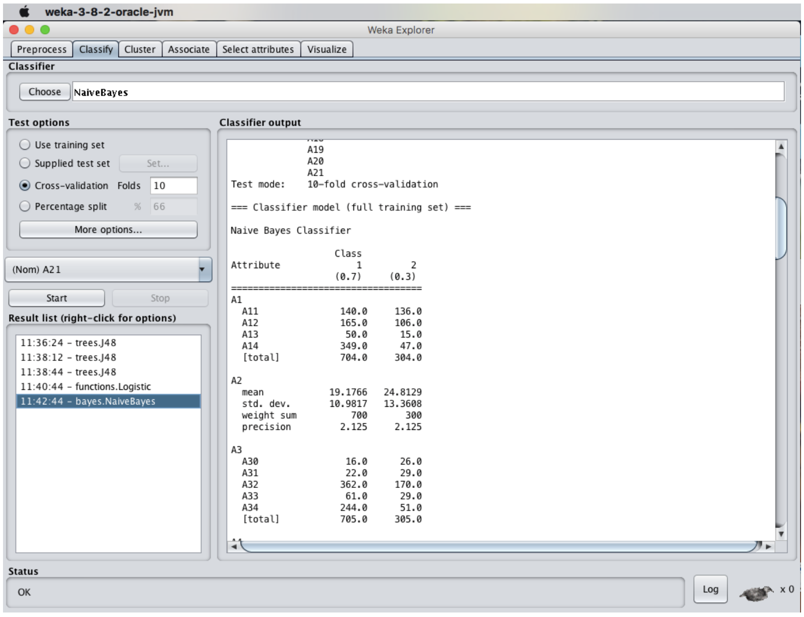
The Naive Bayesian classifier is based on Bayes' theorem with the independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the

Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods.

Steps :

- Load the dataset into the weka tool and preprocess it.
- Apply the classification the Naïve bayes technique and execute for the result.

Output :



Classifier: Choose NaiveBayes

Test options

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

(Nom) A21

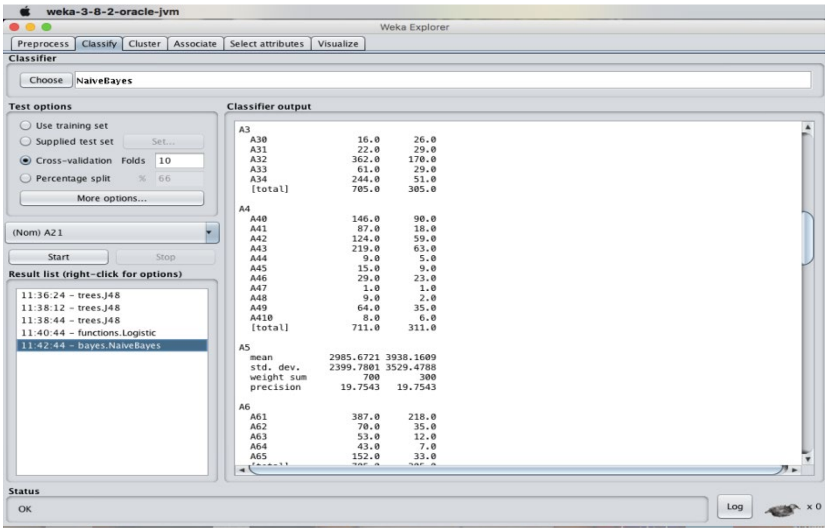
Result list (right-click for options)

```
11:36:24 - trees.J48
11:38:12 - trees.J48
11:38:44 - trees.J48
11:40:44 - functions.Logistic
11:42:44 - bayes.NaiveBayes
```

Classifier output

```
              A19
              A20
              A21
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Naive Bayes Classifier

                      Class
Attribute                 1         2
                      (0.7)     (0.3)
===================================
A1
  A11                 140.0     136.0
  A12                 165.0     106.0
  A13                  50.0      15.0
  A14                 349.0      47.0
  [total]             704.0     304.0

A2
  mean              19.1766   24.8129
  std. dev.         10.9817   13.3608
  weight sum            700       300
  precision           2.125     2.125

A3
  A30                  16.0      26.0
  A31                  22.0      29.0
  A32                 362.0     170.0
  A33                  61.0      29.0
  A34                 244.0      51.0
  [total]             705.0     305.0
```

Status: OK



Classifier: Choose NaiveBayes

Test options

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

(Nom) A21

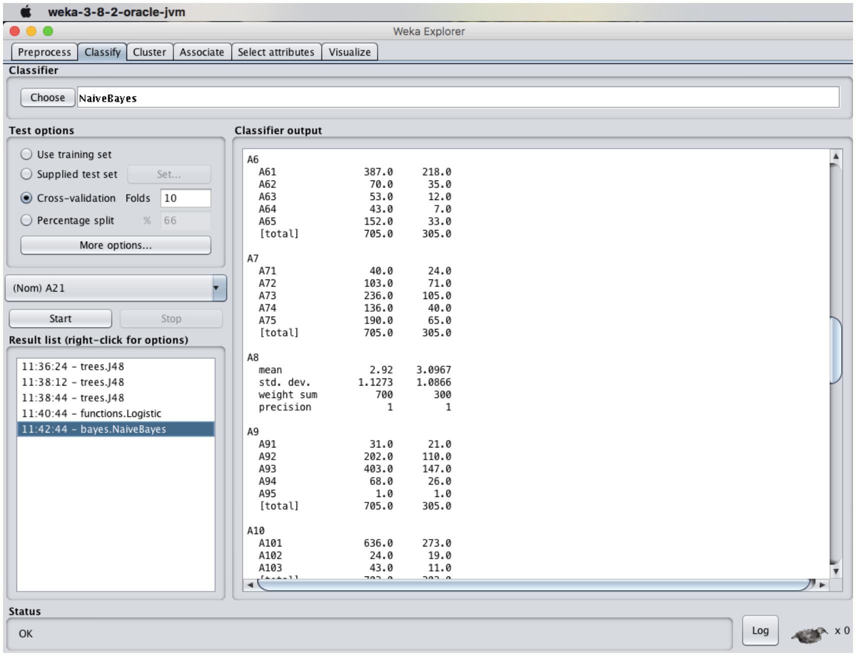
Result list (right-click for options)

```
11:36:24 - trees.J48
11:38:12 - trees.J48
11:38:44 - trees.J48
11:40:44 - functions.Logistic
11:42:44 - bayes.NaiveBayes
```

Classifier output

```
A3
  A30                  16.0      26.0
  A31                  22.0      29.0
  A32                 362.0     170.0
  A33                  61.0      29.0
  A34                 244.0      51.0
  [total]             705.0     305.0

A4
  A40                 146.0      90.0
  A41                  87.0      18.0
  A42                 124.0      59.0
  A43                 219.0      63.0
  A44                   9.0       5.0
  A45                  15.0       9.0
  A46                  29.0      23.0
  A47                   1.0       1.0
  A48                   9.0       2.0
  A49                  64.0      35.0
  A410                  8.0       6.0
  [total]             711.0     311.0

A5
  mean            2985.6721 3938.1609
  std. dev.       2399.7801 3529.4788
  weight sum            700       300
  precision         19.7543   19.7543

A6
  A61                 387.0     218.0
  A62                  70.0      35.0
  A63                  53.0      12.0
  A64                  43.0       7.0
  A65                 152.0      33.0
```

Status: OK



Classifier: NaiveBayes

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

(Nom) A21

Result list (right-click for options)

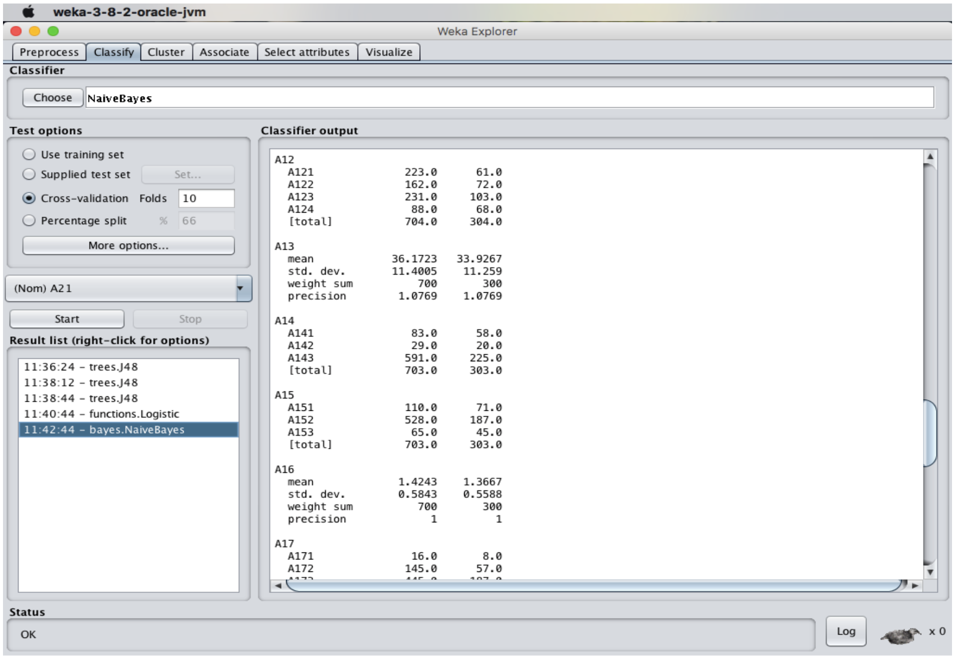
```
11:36:24 - trees.J48
11:38:12 - trees.J48
11:38:44 - trees.J48
11:40:44 - functions.Logistic
11:42:44 - bayes.NaiveBayes
```

Classifier output

```
A6
  A61             387.0     218.0
  A62              70.0      35.0
  A63              53.0      12.0
  A64              43.0       7.0
  A65             152.0      33.0
  [total]         705.0     305.0

A7
  A71              40.0      24.0
  A72             103.0      71.0
  A73             236.0     105.0
  A74             136.0      40.0
  A75             190.0      65.0
  [total]         705.0     305.0

A8
  mean             2.92    3.0967
  std. dev.      1.1273    1.0866
  weight sum        700       300
  precision           1         1

A9
  A91              31.0      21.0
  A92             202.0     110.0
  A93             403.0     147.0
  A94              68.0      26.0
  A95               1.0       1.0
  [total]         705.0     305.0

A10
  A101            636.0     273.0
  A102             24.0      19.0
  A103             43.0      11.0
```

Status

OK



Classifier: NaiveBayes

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

(Nom) A21

Result list (right-click for options)

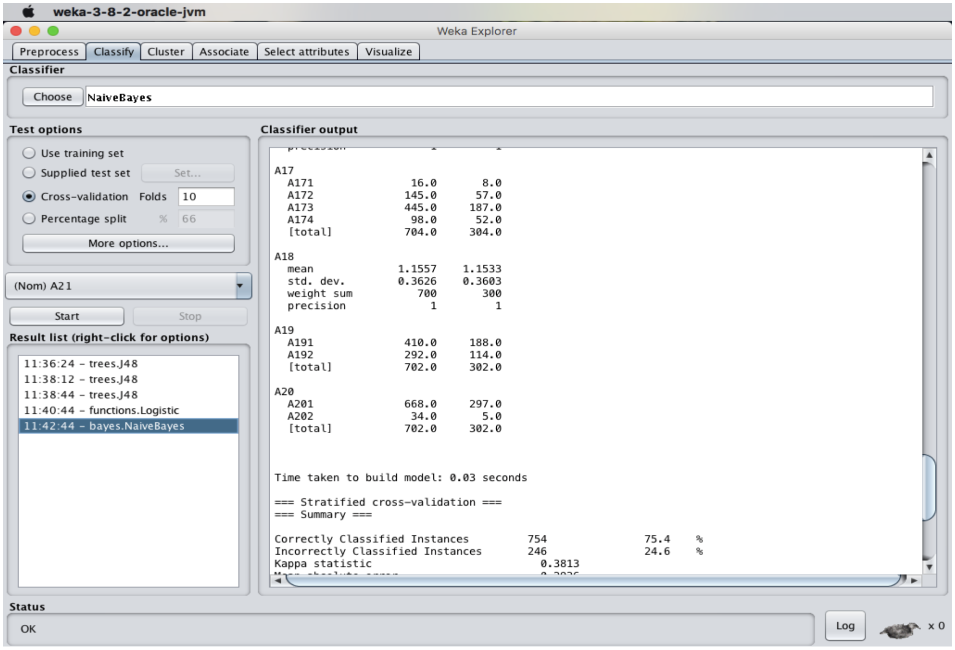
```
11:36:24 - trees.J48
11:38:12 - trees.J48
11:38:44 - trees.J48
11:40:44 - functions.Logistic
11:42:44 - bayes.NaiveBayes
```

Classifier output

```
A12
  A121            223.0      61.0
  A122            162.0      72.0
  A123            231.0     103.0
  A124             88.0      68.0
  [total]         704.0     304.0

A13
  mean          36.1723   33.9267
  std. dev.     11.4005    11.259
  weight sum        700       300
  precision      1.0769    1.0769

A14
  A141             83.0      58.0
  A142             29.0      20.0
  A143            591.0     225.0
  [total]         703.0     303.0

A15
  A151            110.0      71.0
  A152            528.0     187.0
  A153             65.0      45.0
  [total]         703.0     303.0

A16
  mean           1.4243    1.3667
  std. dev.      0.5843    0.5588
  weight sum        700       300
  precision           1         1

A17
  A171             16.0       8.0
  A172            145.0      57.0
```

Status

OK



Classifier: NaiveBayes

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

(Nom) A21

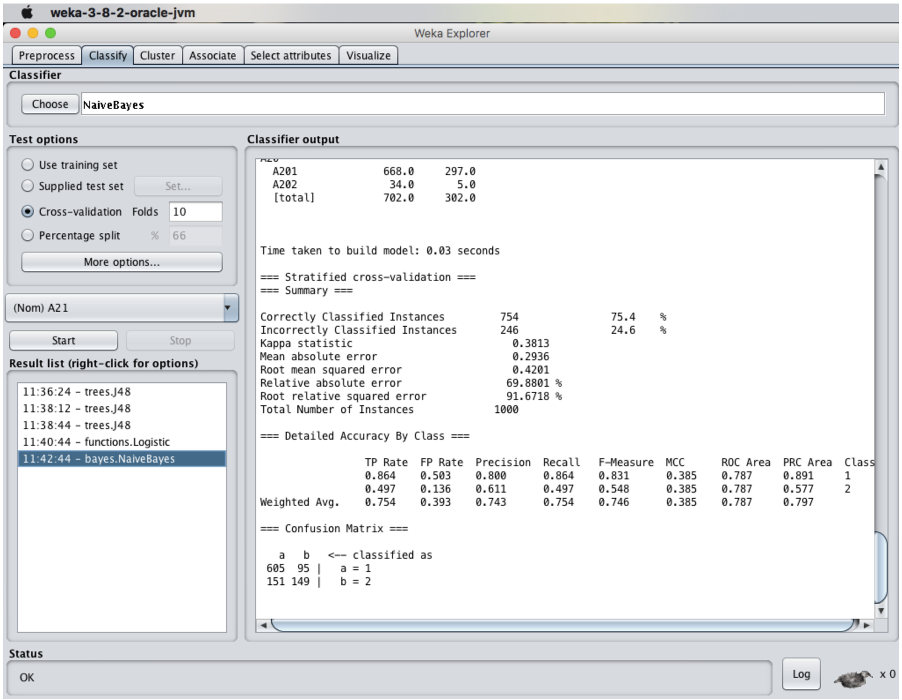
Result list (right-click for options)

```
11:36:24 - trees.J48
11:38:12 - trees.J48
11:38:44 - trees.J48
11:40:44 - functions.Logistic
11:42:44 - bayes.NaiveBayes
```

Classifier output

```
A17
  A171                16.0       8.0
  A172               145.0      57.0
  A173               445.0     187.0
  A174                98.0      52.0
  [total]            704.0     304.0

A18
  mean              1.1557    1.1533
  std. dev.         0.3626    0.3603
  weight sum           700       300
  precision              1         1

A19
  A191               410.0     188.0
  A192               292.0     114.0
  [total]            702.0     302.0

A20
  A201               668.0     297.0
  A202                34.0       5.0
  [total]            702.0     302.0



Time taken to build model: 0.03 seconds

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

Correctly Classified Instances         754               75.4    %
Incorrectly Classified Instances       246               24.6    %
Kappa statistic                          0.3813
```

Status: OK



Classifier: NaiveBayes

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

(Nom) A21

Result list (right-click for options)

```
11:36:24 - trees.J48
11:38:12 - trees.J48
11:38:44 - trees.J48
11:40:44 - functions.Logistic
11:42:44 - bayes.NaiveBayes
```

Classifier output

```
  A201               668.0     297.0
  A202                34.0       5.0
  [total]            702.0     302.0



Time taken to build model: 0.03 seconds

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

Correctly Classified Instances         754               75.4    %
Incorrectly Classified Instances       246               24.6    %
Kappa statistic                          0.3813
Mean absolute error                      0.2936
Root mean squared error                  0.4201
Relative absolute error                 69.8801 %
Root relative squared error             91.6718 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.864    0.503    0.800      0.864    0.831      0.385    0.787     0.891     1
                 0.497    0.136    0.611      0.497    0.548      0.385    0.787     0.577     2
Weighted Avg.    0.754    0.393    0.743      0.754    0.746      0.385    0.787     0.797

=== Confusion Matrix ===

   a   b   <-- classified as
 605  95 |   a = 1
 151 149 |   b = 2
```

Status: OK

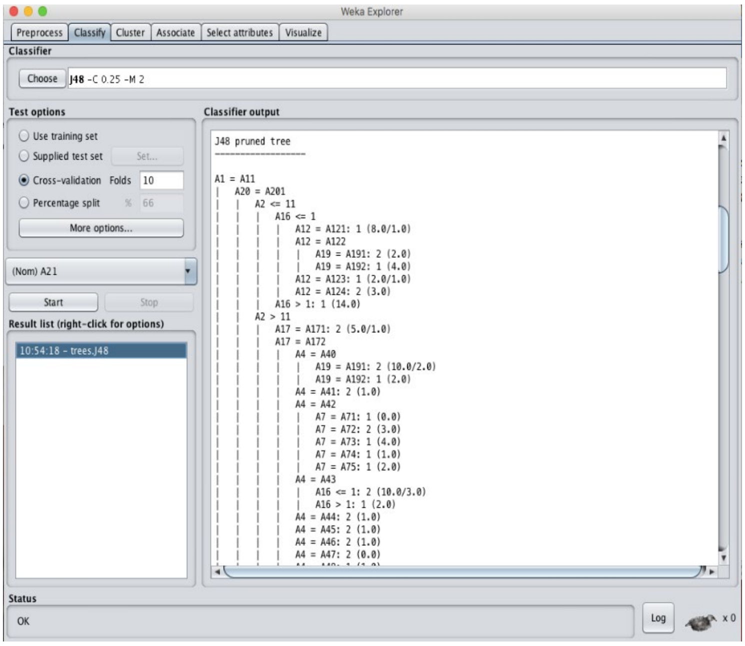
C) J48 Algorithm :

Classification is the process of building a model of classes from a set of records that contain class labels. Decision Tree Algorithm is to find out the way the attributes-vector behaves for a number of instances. Also on the bases of the training instances the classes for the newly generated instances are being found. This algorithm generates the rules for the prediction of the target variable. With the help of tree classification algorithm the critical distribution of the data is easily understandable.

Steps :

- Load the dataset into the weka tool and preprocess it.
- Apply the classification the J48 technique and execute for the result.

Output :



J48 -C 0.25 -M 2

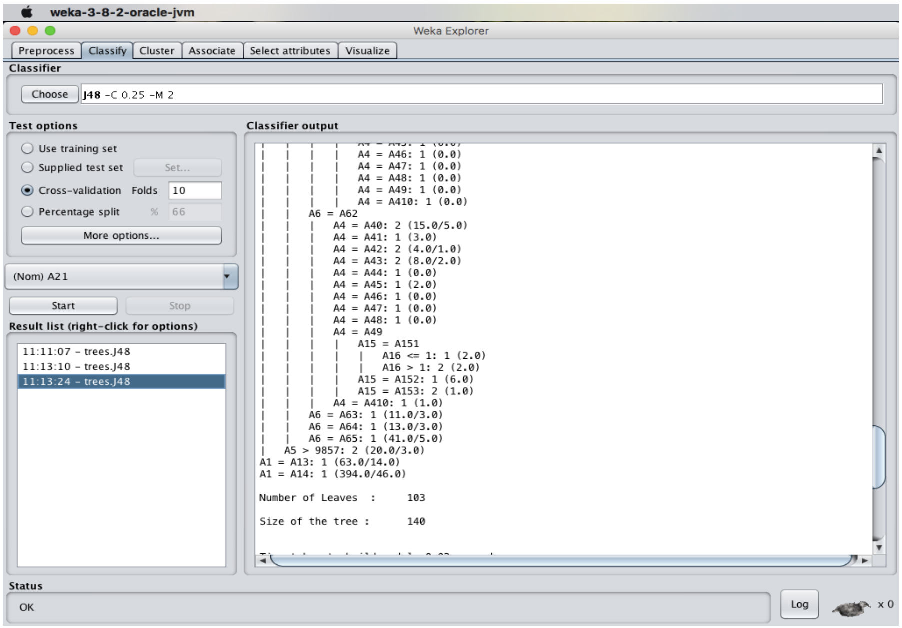
○ Use training set
○ Supplied test set
● Cross-validation Folds 10
○ Percentage split % 66

(Nom) A21

10:54:18 - trees.J48

```
J48 pruned tree
------------------

A1 = A11
|   A20 = A201
|   |   A2 <= 11
|   |   |   A16 <= 1
|   |   |   |   A12 = A121: 1 (8.0/1.0)
|   |   |   |   A12 = A122
|   |   |   |   |   A19 = A191: 2 (2.0)
|   |   |   |   |   A19 = A192: 1 (4.0)
|   |   |   |   A12 = A123: 1 (2.0/1.0)
|   |   |   |   A12 = A124: 2 (3.0)
|   |   |   A16 > 1: 1 (14.0)
|   |   A2 > 11
|   |   |   A17 = A171: 2 (5.0/1.0)
|   |   |   A17 = A172
|   |   |   |   A4 = A40
|   |   |   |   |   A19 = A191: 2 (10.0/2.0)
|   |   |   |   |   A19 = A192: 1 (2.0)
|   |   |   |   A4 = A41: 2 (1.0)
|   |   |   |   A4 = A42
|   |   |   |   |   A7 = A71: 1 (0.0)
|   |   |   |   |   A7 = A72: 2 (3.0)
|   |   |   |   |   A7 = A73: 1 (4.0)
|   |   |   |   |   A7 = A74: 1 (1.0)
|   |   |   |   |   A7 = A75: 1 (2.0)
|   |   |   |   A4 = A43
|   |   |   |   |   A16 <= 1: 2 (10.0/3.0)
|   |   |   |   |   A16 > 1: 1 (2.0)
|   |   |   |   A4 = A44: 2 (1.0)
|   |   |   |   A4 = A45: 2 (1.0)
|   |   |   |   A4 = A46: 2 (1.0)
|   |   |   |   A4 = A47: 2 (0.0)
```

Status: OK



J48 -C 0.25 -M 2

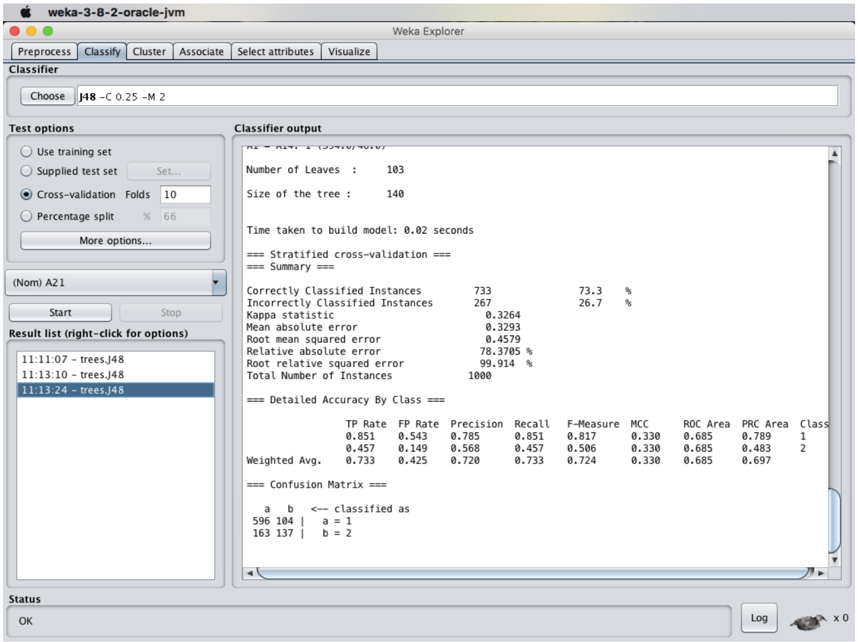
○ Use training set
○ Supplied test set
● Cross-validation Folds 10
○ Percentage split % 66

(Nom) A21

11:11:07 - trees.J48
11:13:10 - trees.J48
11:13:24 - trees.J48

```
|   |   |   |   |   A4 = A46: 1 (0.0)
|   |   |   |   |   A4 = A47: 1 (0.0)
|   |   |   |   |   A4 = A48: 1 (0.0)
|   |   |   |   |   A4 = A49: 1 (0.0)
|   |   |   |   |   A4 = A410: 1 (0.0)
|   |   |   A6 = A62
|   |   |   |   A4 = A40: 2 (15.0/5.0)
|   |   |   |   A4 = A41: 1 (3.0)
|   |   |   |   A4 = A42: 2 (4.0/1.0)
|   |   |   |   A4 = A43: 2 (8.0/2.0)
|   |   |   |   A4 = A44: 1 (0.0)
|   |   |   |   A4 = A45: 1 (2.0)
|   |   |   |   A4 = A46: 1 (0.0)
|   |   |   |   A4 = A47: 1 (0.0)
|   |   |   |   A4 = A48: 1 (0.0)
|   |   |   |   A4 = A49
|   |   |   |   |   A15 = A151
|   |   |   |   |   |   A16 <= 1: 1 (2.0)
|   |   |   |   |   |   A16 > 1: 2 (2.0)
|   |   |   |   |   A15 = A152: 1 (6.0)
|   |   |   |   |   A15 = A153: 2 (1.0)
|   |   |   |   A4 = A410: 1 (1.0)
|   |   |   A6 = A63: 1 (11.0/3.0)
|   |   |   A6 = A64: 1 (13.0/3.0)
|   |   |   A6 = A65: 1 (41.0/5.0)
|   A5 > 9857: 2 (20.0/3.0)
A1 = A13: 1 (63.0/14.0)
A1 = A14: 1 (394.0/46.0)

Number of Leaves  :     103

Size of the tree :      140
```

Status: OK



J48 -C 0.25 -M 2

● Cross-validation Folds 10

% 66

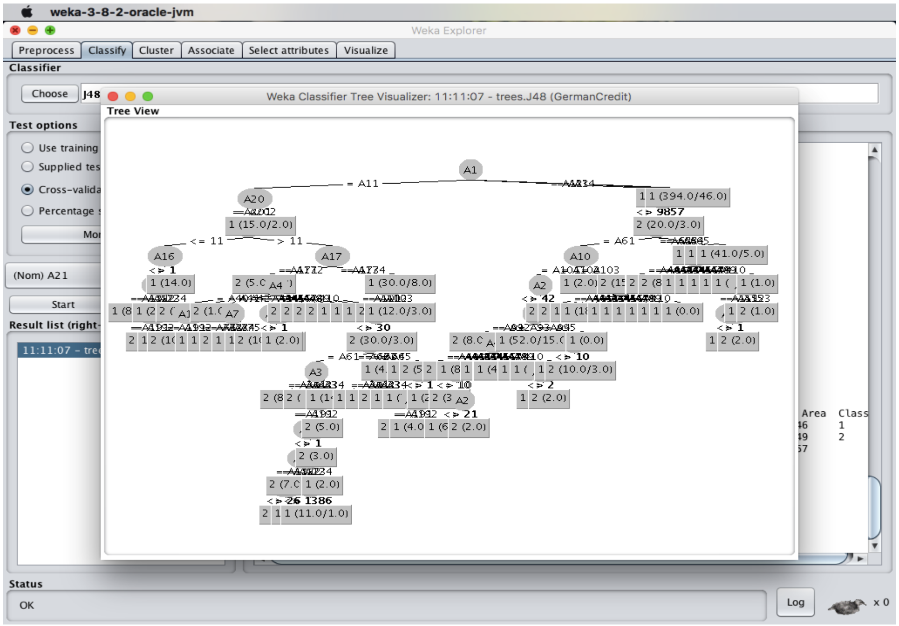
(Nom) A21

Result list (right-click for options)

11:11:07 - trees.J48
11:13:10 - trees.J48
11:13:24 - trees.J48

```
Number of Leaves  :     103

Size of the tree :      140


Time taken to build model: 0.02 seconds

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

Correctly Classified Instances         733               73.3    %
Incorrectly Classified Instances       267               26.7    %
Kappa statistic                          0.3264
Mean absolute error                      0.3293
Root mean squared error                  0.4579
Relative absolute error                 78.3705 %
Root relative squared error             99.914  %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.851    0.543    0.785      0.851    0.817      0.330    0.685     0.789     1
                 0.457    0.149    0.568      0.457    0.506      0.330    0.685     0.483     2
Weighted Avg.    0.733    0.425    0.720      0.733    0.724      0.330    0.685     0.697

=== Confusion Matrix ===

   a   b   <-- classified as
 596 104 |   a = 1
 163 137 |   b = 2
```

Status: OK



D) K-Nearest Neighbor :

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.

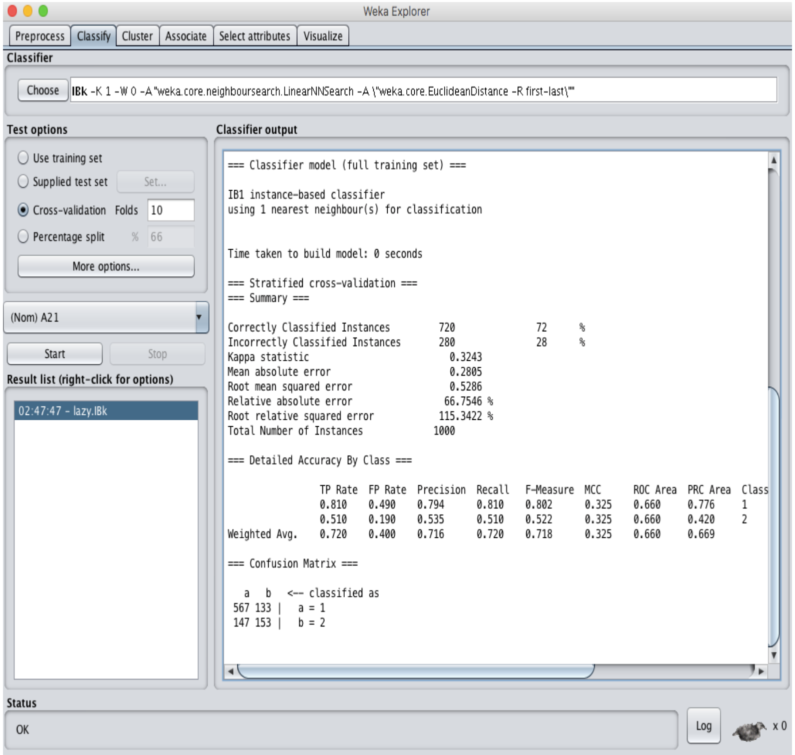
It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data).

We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

Steps :

- Load the dataset into the weka tool and preprocess it.
- Apply the classification the K- Nearest Neighbor technique and execute for the result.

Output :



IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

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

(Nom) A21

02:47:47 - lazy.IBk

OK

Classifier output

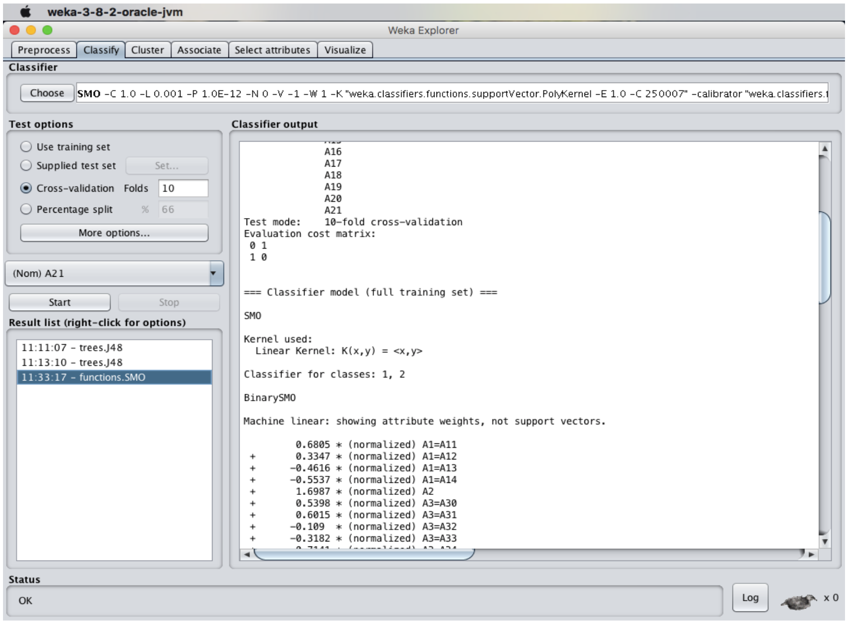
```
=== Classifier model (full training set) ===

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

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

Correctly Classified Instances         720               72      %
Incorrectly Classified Instances       280               28      %
Kappa statistic                          0.3243
Mean absolute error                      0.2805
Root mean squared error                  0.5286
Relative absolute error                 66.7546 %
Root relative squared error            115.3422 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.810    0.490    0.794      0.810    0.802      0.325    0.660     0.776     1
                 0.510    0.190    0.535      0.510    0.522      0.325    0.660     0.420     2
Weighted Avg.    0.720    0.400    0.716      0.720    0.718      0.325    0.660     0.669

=== Confusion Matrix ===

   a   b   <-- classified as
 567 133 |   a = 1
 147 153 |   b = 2
```

E) SMO Algorithm :

The iterative algorithm Sequential Minimal Optimization (SMO) is used for solving quadratic programming (QP) problems. One example where QP problems are relevant is during the training process of support vector machines (SVM). The SMO algorithm is used to solve in this example a constraint optimization problem. John Platt proposed this algorithm in 1998 and it was successfully used since then. We describe here the basics of the algorithm in the light of big data.

Steps :

- Load the dataset into the weka tool and preprocess it.
- Apply the classification the Sequential Minimal Optimization (SMO)technique and execute for the result.



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.f

Test options

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

(Nom) A21

Result list (right-click for options)

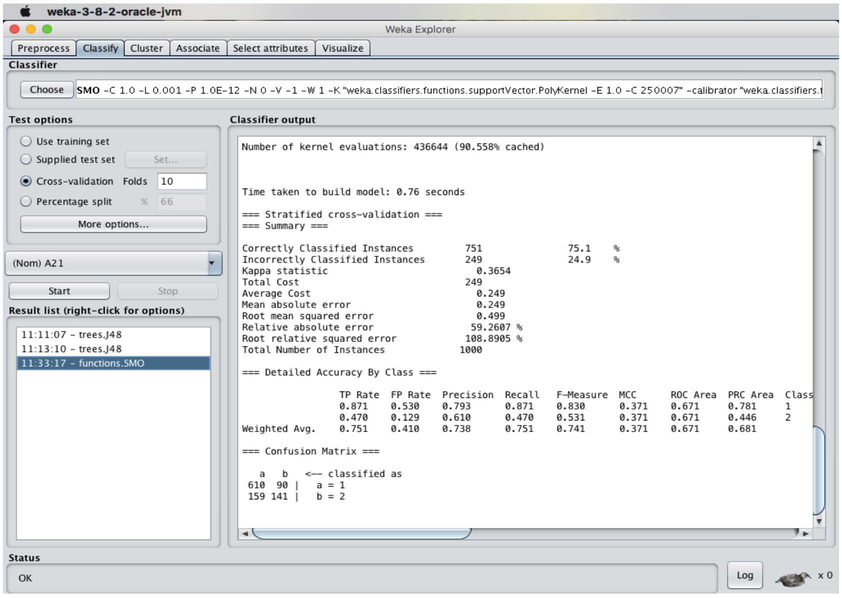
11:11:07 - trees.J48
11:13:10 - trees.J48
11:33:17 - functions.SMO

Classifier output

```
              A16
              A17
              A18
              A19
              A20
              A21
Test mode:    10-fold cross-validation
Evaluation cost matrix:
 0 1
 1 0


=== Classifier model (full training set) ===

SMO

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

Classifier for classes: 1, 2

BinarySMO

Machine linear: showing attribute weights, not support vectors.

         0.6805 * (normalized) A1=A11
 +       0.3347 * (normalized) A1=A12
 +      -0.4616 * (normalized) A1=A13
 +      -0.5537 * (normalized) A1=A14
 +       1.6987 * (normalized) A2
 +       0.5398 * (normalized) A3=A30
 +       0.6015 * (normalized) A3=A31
 +      -0.109  * (normalized) A3=A32
 +      -0.3182 * (normalized) A3=A33
```

Status

OK



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.f

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

(Nom) A21

11:11:07 - trees.J48
11:13:10 - trees.J48
11:33:17 - functions.SMO

```
Number of kernel evaluations: 436644 (90.558% cached)



Time taken to build model: 0.76 seconds

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

Correctly Classified Instances         751               75.1    %
Incorrectly Classified Instances       249               24.9    %
Kappa statistic                          0.3654
Total Cost                             249
Average Cost                             0.249
Mean absolute error                      0.249
Root mean squared error                  0.499
Relative absolute error                 59.2607 %
Root relative squared error            108.8905 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.871    0.530    0.793      0.871    0.830      0.371    0.671     0.781     1
                 0.470    0.129    0.610      0.470    0.531      0.371    0.671     0.446     2
Weighted Avg.    0.751    0.410    0.738      0.751    0.741      0.371    0.671     0.681

=== Confusion Matrix ===

   a   b   <-- classified as
 610  90 |   a = 1
 159 141 |   b = 2
```

Status
OK

RESULT :

Thus, the comparison of the confusion matrix for all the methods and techniques. Out of the comparing matrix with all the techniques there is a change in instances. Naïve bayes has more number of correct instances than other but when compared to time K-nearest neighbor is best. The above graphs will show the variations of values in the parameters.