**Experiment – 1**

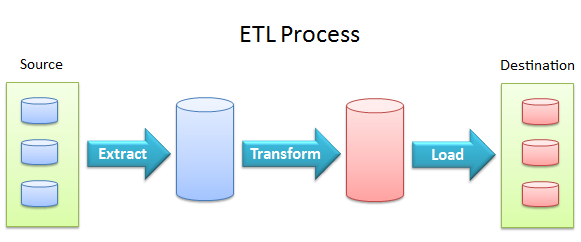
**Aim: Study of ETL process and its tools.**

**ETL process:**

ETL stands for Extract, Transform and Load. In computing, ETL refers to a process in database usage and especially in data warehousing that performs:

* Data extraction – extracts data from homogeneous or heterogeneous data sources
* Data transformation – transforms the data for storing it in the proper format or structure for the purposes of querying and analysis
* Data loading – loads it into the final target (database, more specifically, operational data store, data mart, or data warehouse)

Since the data extraction takes time, it is common to execute the three phases in parallel. While the data is being extracted, another transformation process executes. It processes the already received data and prepares it for loading. As soon as there is some data ready to be loaded into the target, the data loading kicks off without waiting for the completion of the previous phases.



**Process:**

**Extract:**

The Extract step covers the data extraction from the source system and makes it accessible for further processing. The main objective of the extract step is to retrieve all the required data from the source system with as little resources as possible. The extract step should be designed in a way that it does not negatively affect the source system in terms or performance, response time or any kind of locking.

There are several ways to perform the extract:

* Update notification - if the source system is able to provide a notification that a record has been changed and describe the change, this is the easiest way to get the data.
* Incremental extract - some systems may not be able to provide notification that an update has occurred, but they are able to identify which records have been modified and provide an extract of such records. During further ETL steps, the system needs to identify changes and propagate it down. Note, that by using daily extract, we may not be able to handle deleted records properly.
* Full extract - some systems are not able to identify which data has been changed at all, so a full extract is the only way one can get the data out of the system. The full extract requires keeping a copy of the last extract in the same format in order to be able to identify changes. Full extract handles deletions as well.

When using Incremental or Full extracts, the extract frequency is extremely important. Particularly for full extracts; the data volumes can be in tens of gigabytes.

**Clean:**

The cleaning step is one of the most important as it ensures the quality of the data in the data warehouse. Cleaning should perform basic data unification rules, such as:

* Making identifiers unique (sex categories Male/Female/Unknown, M/F/null, Man/Woman/Not Available are translated to standard Male/Female/Unknown)
* Convert null values into standardized Not Available/Not Provided value
* Convert phone numbers, ZIP codes to a standardized form
* Validate address fields, convert them into proper naming, e.g. Street/St/St./Str./Str
* Validate address fields against each other (State/Country, City/State, City/ZIP code, City/Street).

**Transform:**

The transform step applies a set of rules to transform the data from the source to the target. This includes converting any measured data to the same dimension (i.e. conformed dimension) using the same units so that they can later be joined. The transformation step also requires joining data from several sources, generating aggregates, generating surrogate keys, sorting, deriving new calculated values, and applying advanced validation rules.

**Load:**

During the load step, it is necessary to ensure that the load is performed correctly and with as little resources as possible. The target of the Load process is often a database. In order to make the load process efficient, it is helpful to disable any constraints and indexes before the load and enable them back only after the load completes. The referential integrity needs to be maintained by ETL tool to ensure consistency.

**ETL Tools:**

The times of increasing data-dependence forced a lot of companies to invest in complicated data warehousing systems. Their differentiation and incompatibility led to an uncontrolled growth of costs and time needed to coordinate all the processes. The ETL (Extract, transform, load) tools were created to simplify the data management with simultaneous reduction of absorbed effort.

Depending on the needs of customers there are several types of tools.

One of them perform and supervise only selected stages of the ETL process like data migration tools, data transformation tools. Another are complete (ETL Tools) and have many functions that are intended for processing large amounts of data or more complicated ETL projects. Some of them like server engine tools execute many ETL steps at the same time from more than one developer , while other like client engine tools are simpler and execute ETL routines on the same machine as they are developed.

There are two more types. First called code base tools is a family of programing tools which allow you to work with many operating systems and programing languages. The second one called GUI base tools remove the coding layer and allow you to work without any knowledge (in theory) about coding languages.

**How do the ETL tools work?**

The first task is data extraction from internal or external sources. After sending queries to the source system data may go indirectly to the database. However usually there is a need to monitor or gather more information and then go to Staging Area . Some tools extract only new or changed information automatically so we dont have to update it by our own.

The second task is transformation which is a broad category:

* transforming data into a structure which is required to continue the operation (extracted data has usually a structure typical to the source)
* sorting data
* connecting or separating
* cleansing
* checking quality

The third task is loading into a data warehouse.

As you can see the ETL Tools have many other capabilities (next to the main three: extraction, transformation and loading) like for instance sorting , filtering , data profiling , quality control, cleansing , monitoring , synchronization and consolidation.

**ETL TOOLS PROVIDERS:**

Here is a list of the most popular commercial and freeware (open-source) ETL Tools.

Commercial ETL Tools:

IBM Infosphere DataStage, Informatica PowerCenter, Oracle Warehouse Builder (OWB), Oracle Data Integrator (ODI), SAS ETL Studio, Business Objects Data Integrator(BODI), Microsoft SQL Server Integration Services(SSIS), Ab Initio

Freeware, open source ETL tools:

Pentaho Data Integration (Kettle), Talend Integrator Suite, CloverETL, Jasper ETL

**Experiment – 2**

**Aim: Program of Data warehouse cleansing to input names from users (inconsistent) and format them.**

**Data Cleansing:**

Data cleansing, data cleaning, or data scrubbing is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data. Data cleansing may be performed interactively with data wrangling tools, or as batch processing through scripting.

After cleansing, a data set should be consistent with other similar data sets in the system. The inconsistencies detected or removed may have been originally caused by user entry errors, by corruption in transmission or storage, or by different data dictionary definitions of similar entities in different stores.

**Program:**

#include <bits/stdc++.h>

using namespace std;

int main(int argc, char const \*argv[])

{

vector<string> buff;

string data;

// open a file in read mode.

ifstream infile;

infile.open("dmbi2.txt");

while ( getline (infile,data) ) {

transform(data.begin(), data.end(), data.begin(), ::tolower);

if(data.length()>0) data[0]=toupper(data[0]);

buff.push\_back(data);

}

infile.close();

// open a file in write mode.

ofstream outfile;

outfile.open("ankit.txt");

cout << "Correcting the data in file........." << endl;

for(int i=0;i<buff.size();i++)

outfile<<buff[i]<<endl;

// close the opened file.

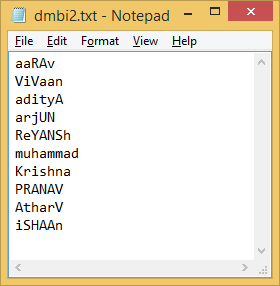
outfile.close();

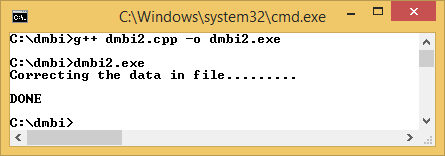
cout<<"\n\nDONE";

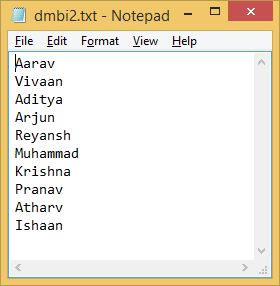
return 0;

}

**Output:**







**Experiment – 3**

**Aim: Program of Data warehouse cleansing to remove redundancy in data.**

**Program:**

#include <bits/stdc++.h>

using namespace std;

bool isPresent(vector<string> buff,string d)

{

for(int i=0;i<buff.size();i++)

if(d.compare(buff[i])==0)

return true;

return false;

}

int main () {

vector<string> buff;

string data;

// open a file in read mode.

ifstream infile;

infile.open("dmbi3.txt");

while ( getline(infile,data) ) {

if(!isPresent(buff,data))

buff.push\_back(data);

}

infile.close();

// open a file in write mode.

ofstream outfile;

outfile.open("dmbi3.txt");

cout << "Removing redundant data in file........." << endl;

for(int i=0;i<buff.size();i++)

outfile<<buff[i]<<endl;

// close the opened file.

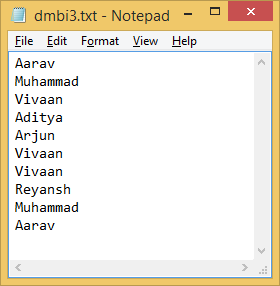
outfile.close();

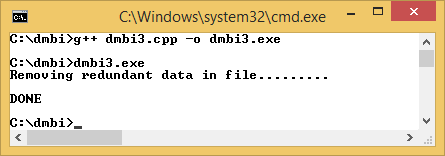
cout<<"\nDONE\n";

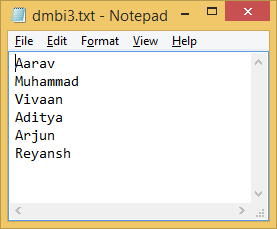
return 0;

}

**Output:**







**Experiment – 4**

**Aim: Introduction to WEKA tool.**

**WEKA:**

Weka (Waikato Environment for Knowledge Analysis) is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

It is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. It is free software licensed under the GNU General Public License.



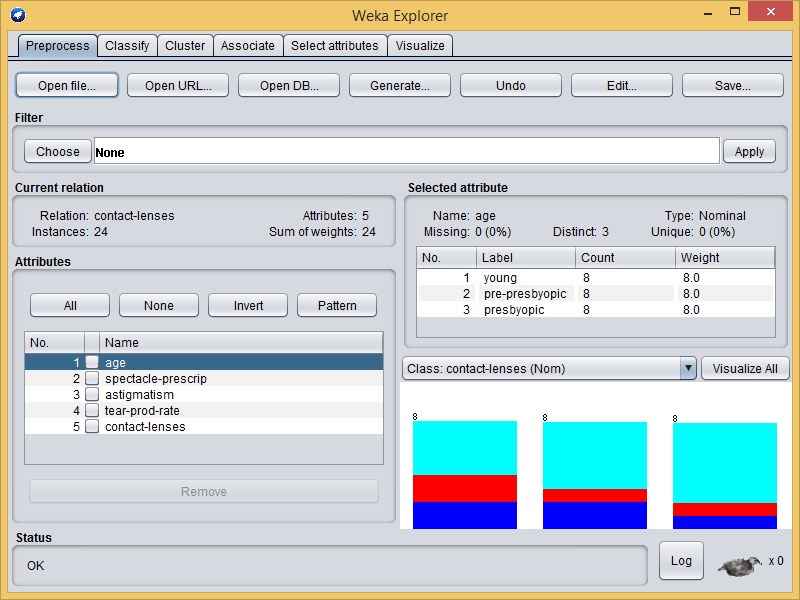
Main features of WEKA:

* 49 data pre-processing tools
* 76 classification/regression algorithms
* 8 clustering algorithms
* 15 attribute/sunset evaluators + 10 search algorithms for feature selection
* 3 algorithms for finding association rules
* 3 graphical user interface
  + “The Explorer” (exploratory data analysis)
  + “The Experimenter” (experimental environment)
  + “The KnowledgeFlow” (new process model inspired interface)
* Custom extensions and plugins can be developed

More algorithms/features being added….

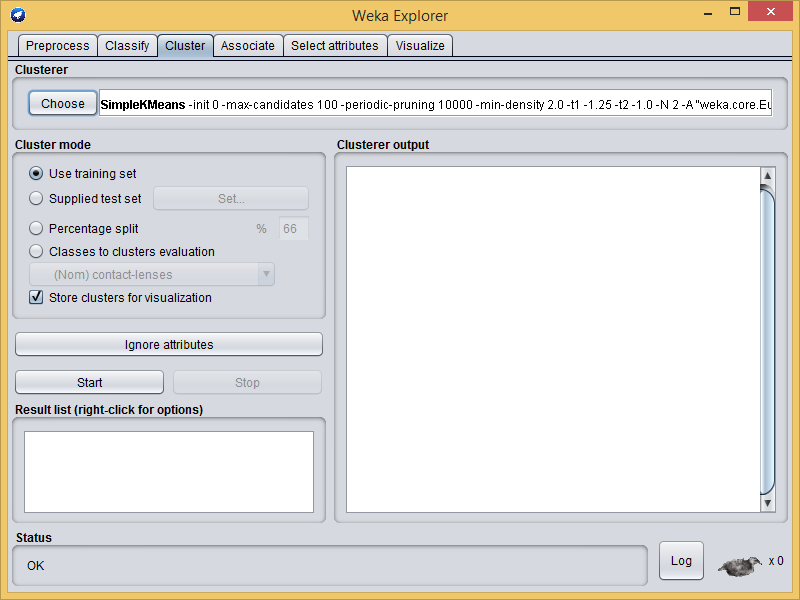
Weka's main user interface is the Explorer, featuring several panels which provide access to the main components of the workbench: the Pre-process Panel, the Classify Panel, the Associate Panel, the Cluster Panel, the Select Attributes Panel, and the Visualize Panel.

**Preprocess Panel:** The Preprocess Panel has facilities for importing data from a database, a CSV file, or other data file types, and for preprocessing this data using a so-called filtering algorithm. These filters can be used to transform the data (e.g., turning numeric attributes into discrete ones) and make it possible to delete instances and attributes according to specific criteria.



**Classify Panel:** The Classify Panel enables the user to apply classification and regression algorithms (indiscriminately called classifiers in Weka) allowing you to the resulting data set, to estimate the accuracy of the resulting predictive model, and to visualize erroneous predictions, ROC curves, etc., or the model itself (if the model is amenable to visualization like, e.g., a decision tree).

**Cluster Panel:** The Cluster Panel gives access to the clustering techniques in Weka, e.g., the simple k-means algorithm. There is also an implementation of the expectation maximization algorithm for learning a mixture of normal distributions.

****

**Associate Panel:** The Associate Panel provides access to association rule learners that attempt to identify all important interrelationships between attributes in the data.

**Select Attributes:** The Select Attributes Panel provides algorithms for identifying the most predictive attributes in a data set.

**Visualize Panel:** The Visualize Panel shows a scatter plot matrix, where individual scatter plots can be selected, enlarged and analysed using various selection operators.

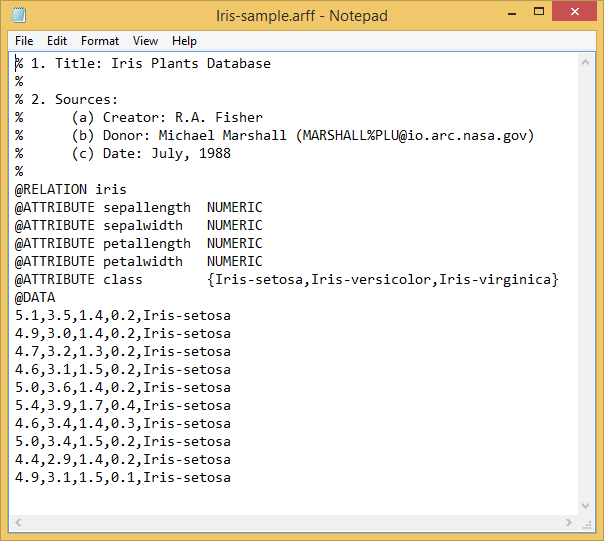
WEKA supports two type of files for input data:

1. **ARFF (Attribute-Relation File Format):**

It is a file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files have two distinct sections. The first section is the Header information, which is followed the Data information.

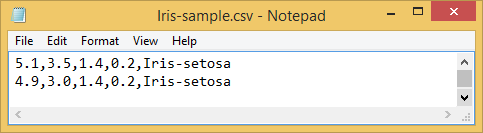
Header Part: The Header of the ARFF file contains the name of the relation, a list of the attributes (the columns in the data), and their types.

DATA Part: It contains the actual data.



1. **CSV (Comma-separated values):**

A CSV is a comma separated values file, which allows data to be saved in a table structured format. CSVs look like a garden-variety spreadsheet but with a .csv extension (Traditionally they take the form of a text file containing information separated by commas, hence the name).



**Advantages of WEKA:**

* Free availability
  + Under the GNU General Public License
* Portability
  + Fully implemented in the Java programming language and thus runs on almost any modern computing platforms like Windows, Mac OS X and Linux
* Comprehensive collection of data pre-processing and modelling techniques
  + Supports standard data mining tasks: data pre-processing, clustering, classification, regression, visualization, and feature selection.
* Easy to use GUI
* Provides access to SQL databases
  + Using Java Database Connectivity and can process the result returned by a database query.

**Disadvantages of WEKA:**

* Sequence modelling is not covered by the algorithms included in the Weka distribution
* Not capable of multi-relational data mining
* Memory bound

**Experiment - 5**

**Aim: Implementation of Classification technique on ARFF files using WEKA.**

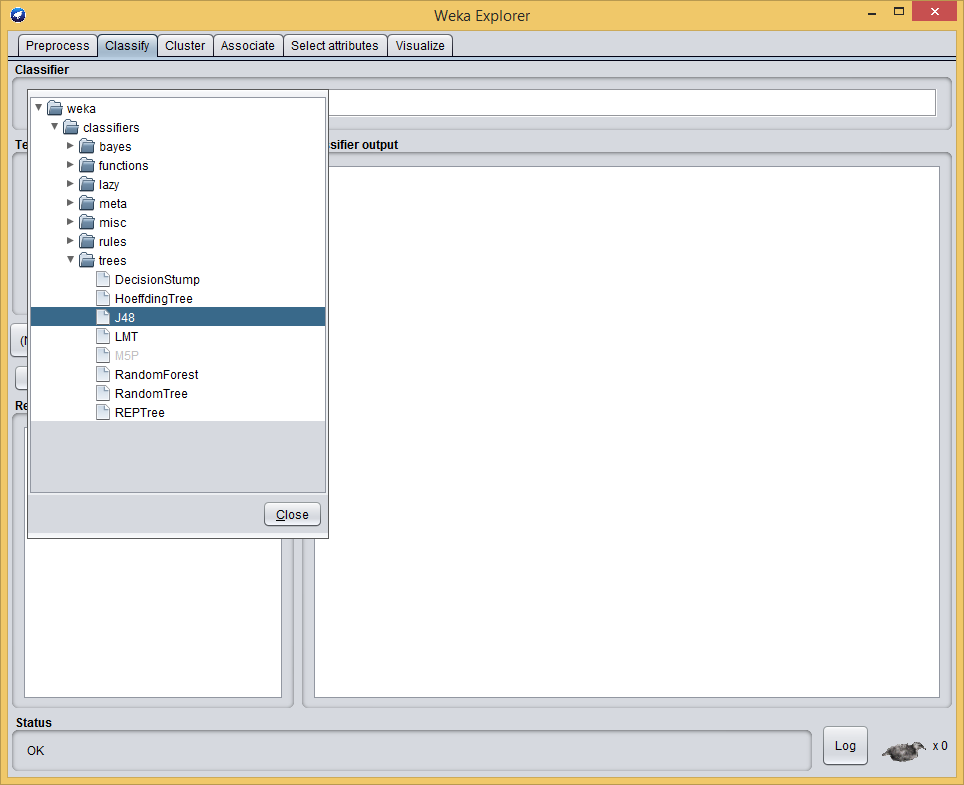
**Classification in WEKA:**

Classifiers in WEKA are the models for predicting nominal or numeric quantities. The learning schemes available in WEKA include decision trees and lists, instance-based classifiers, support vector machines, multi-layer perceptron, logistic regression, and bayes’ nets. “Meta”- classifiers include bagging, boosting, stacking, error-correcting output codes, and locally weighted learning.

In this experiment we are working on Iris Dataset. Once you have your data set loaded, all the tabs are available to you. Click on the ‘Classify’ tab. ‘Classify’ window comes up on the screen. Now you can start analysing the data using the provided algorithms. In this exercise we will analyse the data with C4.5 algorithm using J48, WEKA’s implementation of decision tree learner.

1. **Choosing a Classifier:**

Click on ‘Choose’ button in the ‘Classifier’ box just below the tabs and select C4.5 classifier **WEKA -> Classifiers -> Trees -> J48.**



1. **Setting Test Options:**

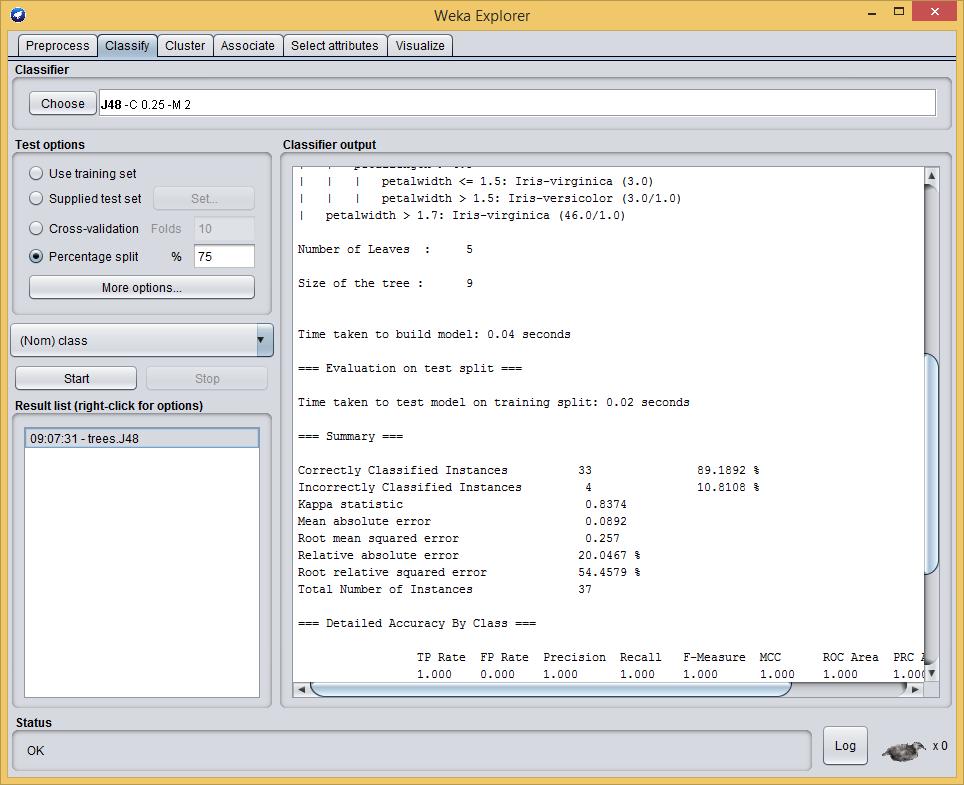
Before you run the classification algorithm, you need to set test options. Set test options in the ‘Test options’ box. The test options that available to you are:

* 1. Use training set. Evaluates the classifier on how well it predicts the class of the instances it was trained on.
  2. Supplied test set. Evaluates the classifier on how well it predicts the class of a set of instances loaded from a file. Clicking on the ‘Set…’ button brings up a dialog allowing you to choose the file to test on.
  3. Cross-validation. Evaluates the classifier by cross-validation, using the number of folds that are entered in the ‘Folds’ text field.
  4. Percentage split. Evaluates the classifier on how well it predicts a certain percentage of the data, which is held out for testing. The amount of data held out depends on the value entered in the ‘%’ field.

In this experiment we will evaluate classifier based on how well it predicts 75% of the tested data. Check ‘Percentage split’ radio-button and change value to 75%.

Now, you can run the classification algorithm. Click on ‘Start’ button to start the learning process. You can stop learning process at any time by clicking on ‘Stop’ button.

When training set is complete, the ‘Classifier’ output area on the right panel of ‘Classify’ window is filled with text describing the results of training and testing. A new entry appears in the ‘Result list’ box on the left panel of ‘Classify’ window.



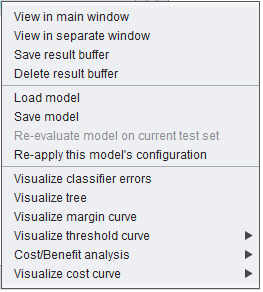
1. **Analysing the results:**

Result is:

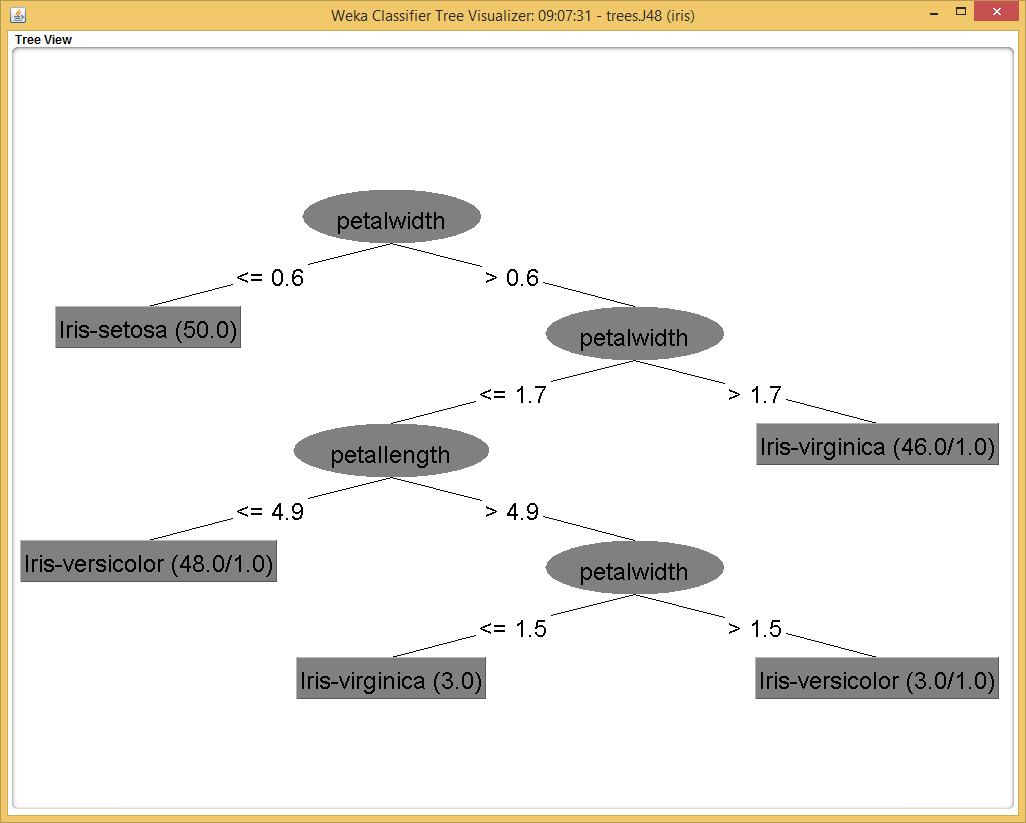
|  |
| --- |
| === Run information ===  Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2  Relation: iris  Instances: 150  Attributes: 5  sepallength  sepalwidth  petallength  petalwidth  class  Test mode: split 75.0% train, remainder test  === Classifier model (full training set) ===  J48 pruned tree  ------------------  petalwidth <= 0.6: Iris-setosa (50.0)  petalwidth > 0.6  | petalwidth <= 1.7  | | petallength <= 4.9: Iris-versicolor (48.0/1.0)  | | petallength > 4.9  | | | petalwidth <= 1.5: Iris-virginica (3.0)  | | | petalwidth > 1.5: Iris-versicolor (3.0/1.0)  | petalwidth > 1.7: Iris-virginica (46.0/1.0)  Number of Leaves : 5  Size of the tree : 9  Time taken to build model: 0.04 seconds  === Evaluation on test split ===  Time taken to test model on training split: 0.02 seconds  === Summary ===  Correctly Classified Instances 33 89.1892 %  Incorrectly Classified Instances 4 10.8108 %  Kappa statistic 0.8374  Mean absolute error 0.0892  Root mean squared error 0.257  Relative absolute error 20.0467 %  Root relative squared error 54.4579 %  Total Number of Instances 37  === 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-setosa  0.846 0.083 0.846 0.846 0.846 0.763 0.917 0.793 Iris-versicolor  0.846 0.083 0.846 0.846 0.846 0.763 0.917 0.793 Iris-virginica  Weighted Avg. 0.892 0.059 0.892 0.892 0.892 0.833 0.941 0.854  === Confusion Matrix ===  a b c <-- classified as  11 0 0 | a = Iris-setosa  0 11 2 | b = Iris-versicolor  0 2 11 | c = Iris-virginica |

1. **Visualization of Results:**

WEKA lets you to see a graphical representation of the classification tree. Right-click on the entry in ‘Result list’ for which you would like to visualize a tree. It invokes a menu containing the following items:



Select the item ‘Visualize tree’, a new window comes up to the screen displaying the tree.



**Experiment – 6**

**Aim: Implementation of Clustering technique on ARFF files using WEKA.**

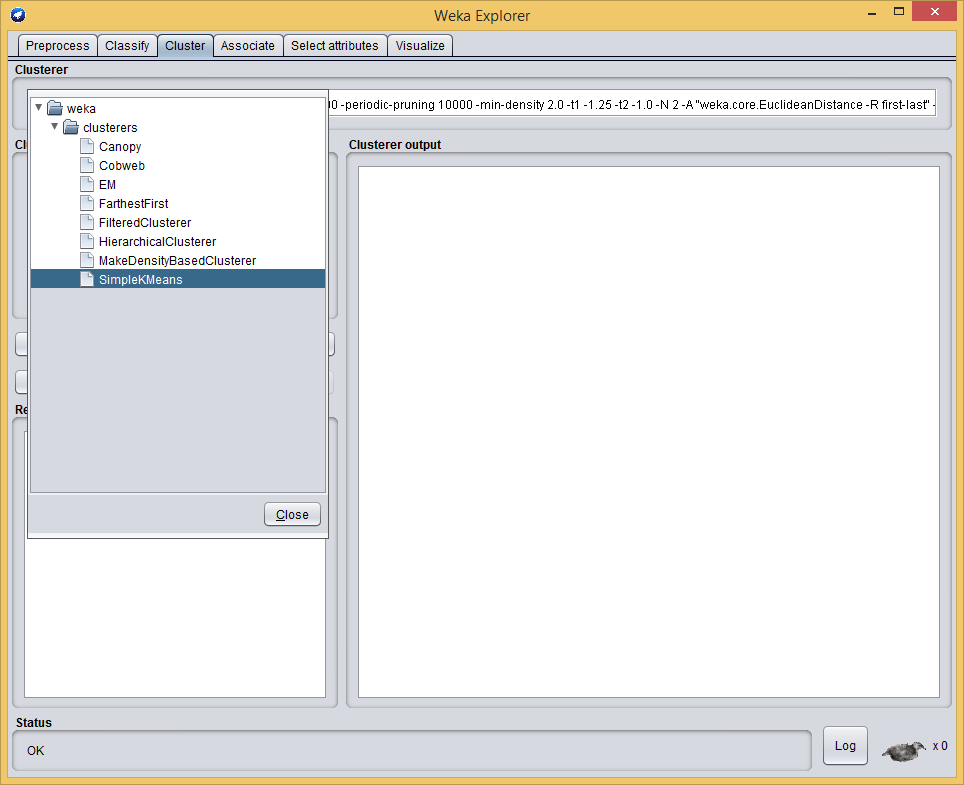
**Clustering in WEKA:**

WEKA contains “clusters” for finding groups of similar instances in a dataset. The clustering schemes available in WEKA are k-Means, EM, Cobweb, X-means, FarthestFirst. Clusters can be visualized and compared to “true” clusters (if given). Evaluation is based on log likelihood if clustering scheme produces a probability distribution.

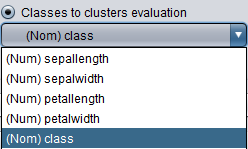
In this experiment we are working on Iris Dataset. Once you have your data set loaded, all the tabs are available to you. Click ‘Cluster’ tab at the top of WEKA Explorer window. In this exercise we will analyse the data with SimpleKMeans.

1. **Choosing Clustering Scheme:**

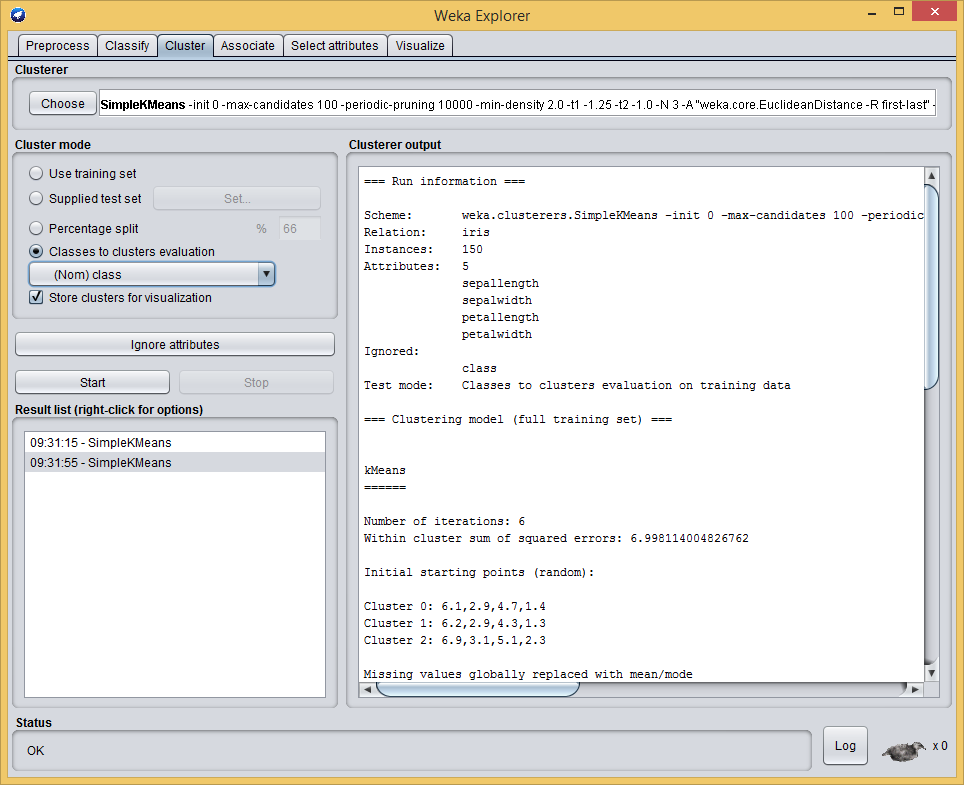
In the ‘Cluster’ box click on ‘Choose’ button. In pull-down menu select WEKA -> Clusters, and select the cluster scheme ‘SimpleKMeans’.



1. **Setting Test Options:**

Before you run the clustering algorithm, you need to choose ‘Cluster mode’. Click on ‘Classes to cluster evaluation ’radio-button in ‘Cluster mode’ box and select classes in the pull-down box below.

Once the options have been specified, you can run the clustering algorithm. Click on the ‘Start’ button to execute the algorithm.



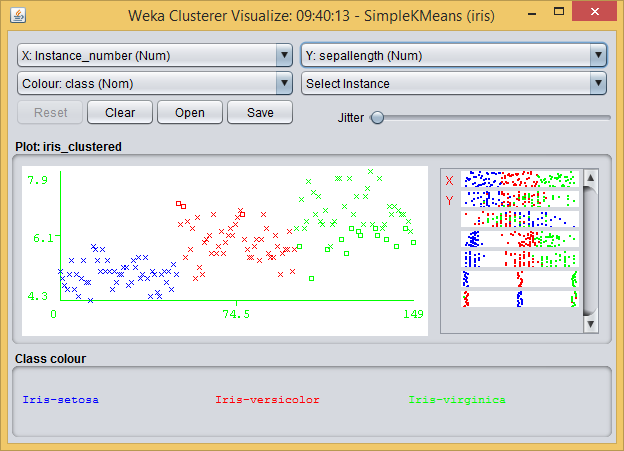
1. **Analysing the results:**

Result is:

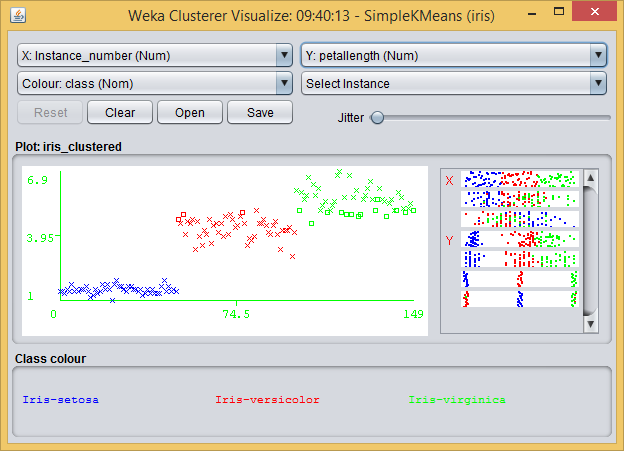
|  |
| --- |
| === Run information ===  Scheme: weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10  Relation: iris  Instances: 150  Attributes: 5  sepallength  sepalwidth  petallength  petalwidth  Ignored:  class  Test mode: Classes to clusters evaluation on training data  === Clustering model (full training set) ===  kMeans  ======  Number of iterations: 6  Within cluster sum of squared errors: 6.998114004826762  Initial starting points (random):  Cluster 0: 6.1,2.9,4.7,1.4  Cluster 1: 6.2,2.9,4.3,1.3  Cluster 2: 6.9,3.1,5.1,2.3  Missing values globally replaced with mean/mode  Final cluster centroids:  Cluster#  Attribute Full Data 0 1 2  (150.0) (61.0) (50.0) (39.0)  =========================================================  sepallength 5.8433 5.8885 5.006 6.8462  sepalwidth 3.054 2.7377 3.418 3.0821  petallength 3.7587 4.3967 1.464 5.7026  petalwidth 1.1987 1.418 0.244 2.0795  Time taken to build model (full training data) : 0.01 seconds  === Model and evaluation on training set ===  Clustered Instances  0 61 ( 41%)  1 50 ( 33%)  2 39 ( 26%)  Class attribute: class  Classes to Clusters:  0 1 2 <-- assigned to cluster  0 50 0 | Iris-setosa  47 0 3 | Iris-versicolor  14 0 36 | Iris-virginica  Cluster 0 <-- Iris-versicolor  Cluster 1 <-- Iris-setosa  Cluster 2 <-- Iris-virginica  Incorrectly clustered instances : 17.0 11.3333 % |

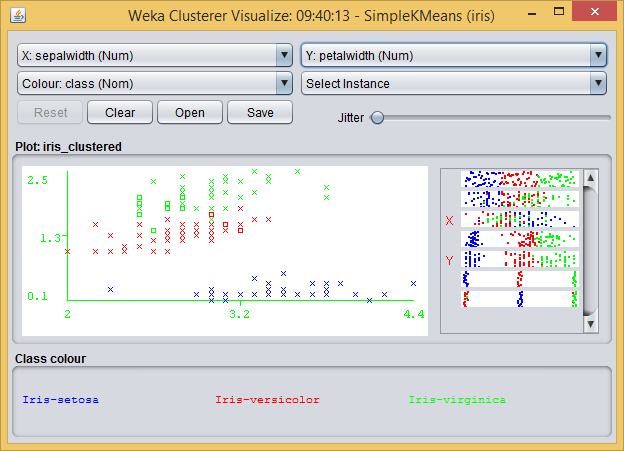
1. **Visualization of Results:**

Right-click on the entry in ‘Result list’ for which you would like to visualize. It invokes a menu containing many items. Click on ‘Visualize Cluster Assignments’.



We can apply different options to get different cluster visualizations.





**Experiment – 7**

**Aim: Implementation of Association Rule technique on ARFF files using WEKA.**

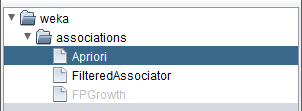
**Finding Associations in WEKA:**

WEKA contains an implementation of the Apriori algorithm for learning association rules. It works only with discrete data and will identify statistical dependencies between groups of attributes, milk, peanut butter and bread, jelly, beer and diapers, with confidence 40% and support 30%. Apriori can compute all rules that have a given minimum support and exceed a given confidence.

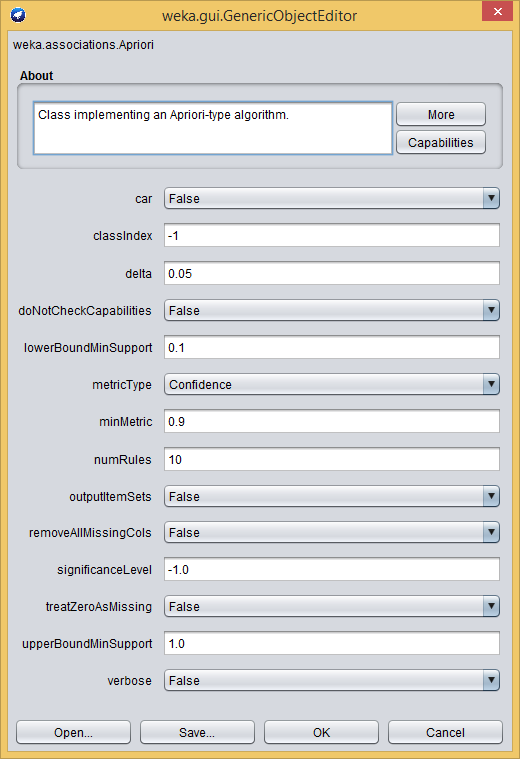
In this experiment we are working on Weather (nominal) Dataset. Once you have your data set loaded, all the tabs are available to you. Click ‘Associate’ tab at the top of WEKA Explorer window. In this exercise we will analyse the data with Apriori algorithm.

1. **Choosing Association algorithm:**

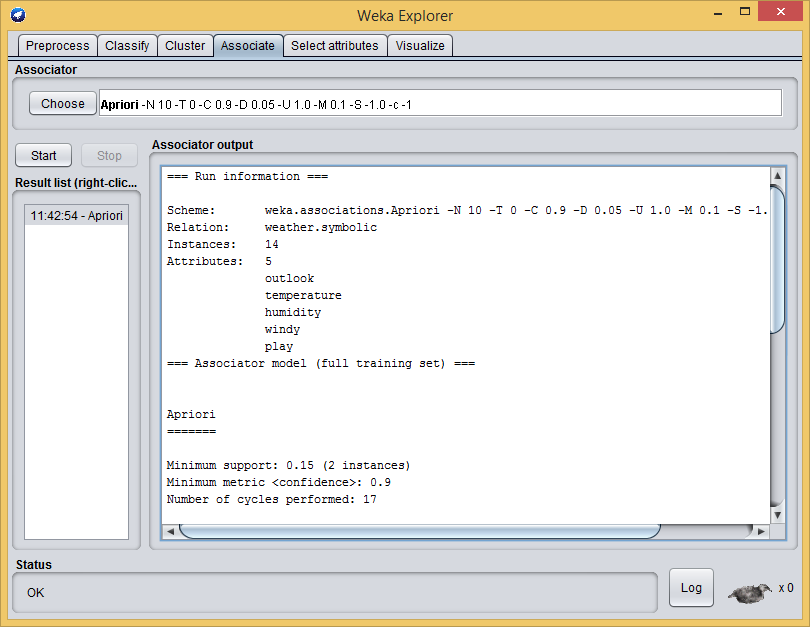
In the ‘Associator’ box click on ‘Choose’ button. In pull-down menu select WEKA -> Associations -> Apriori.

****

1. **Setting Test Options:**

As you can see, there are no other associators to choose and no extra options for testing the learning scheme.Right-click on the ‘Associator’ box, ‘GenericObjectEditor’ appears on your screen. In the dialog box, change the value in ‘minMetric’ to 0.4 for confidence = 40%. Make sure that the default value of rules is set to 100. You can also set other options.

Once the options have been specified, you can run Apriori algorithm. Click on the ‘Start’ button to execute the algorithm.



1. **Analysing the result:**

Result is:

|  |
| --- |
| === 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: weather.symbolic  Instances: 14  Attributes: 5  outlook  temperature  humidity  windy  play  === Associator model (full training set) ===  Apriori  =======  Minimum support: 0.15 (2 instances)  Minimum metric <confidence>: 0.9  Number of cycles performed: 17  Generated sets of large itemsets:  Size of set of large itemsets L(1): 12  Size of set of large itemsets L(2): 47  Size of set of large itemsets L(3): 39  Size of set of large itemsets L(4): 6  Best rules found:  1. outlook=overcast 4 ==> play=yes 4 <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)  2. temperature=cool 4 ==> humidity=normal 4 <conf:(1)> lift:(2) lev:(0.14) [2] conv:(2)  3. humidity=normal windy=FALSE 4 ==> play=yes 4 <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)  4. outlook=sunny play=no 3 ==> humidity=high 3 <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)  5. outlook=sunny humidity=high 3 ==> play=no 3 <conf:(1)> lift:(2.8) lev:(0.14) [1] conv:(1.93)  6. outlook=rainy play=yes 3 ==> windy=FALSE 3 <conf:(1)> lift:(1.75) lev:(0.09) [1] conv:(1.29)  7. outlook=rainy windy=FALSE 3 ==> play=yes 3 <conf:(1)> lift:(1.56) lev:(0.08) [1] conv:(1.07)  8. temperature=cool play=yes 3 ==> humidity=normal 3 <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)  9. outlook=sunny temperature=hot 2 ==> humidity=high 2 <conf:(1)> lift:(2) lev:(0.07) [1] conv:(1)  10. temperature=hot play=no 2 ==> outlook=sunny 2 <conf:(1)> lift:(2.8) lev:(0.09) [1] conv:(1.29) |

**Experiment – 8**

**Aim: Implementation of Visualization technique on ARFF files using WEKA.**

**Visualization in WEKA:**

WEKA’s visualization allows you to visualize a 2-D plot of the current working relation. Visualization is very useful in practice, it helps to determine difficulty of the learning problem. WEKA can visualize single attributes (1-d) and pairs of attributes (2-d), rotate 3-d visualizations (Xgobi-style). WEKA has “Jitter” option to deal with nominal attributes and to detect “hidden” data points.

In this experiment we are working on Iris Dataset. Once you have your data set loaded, all the tabs are available to you.

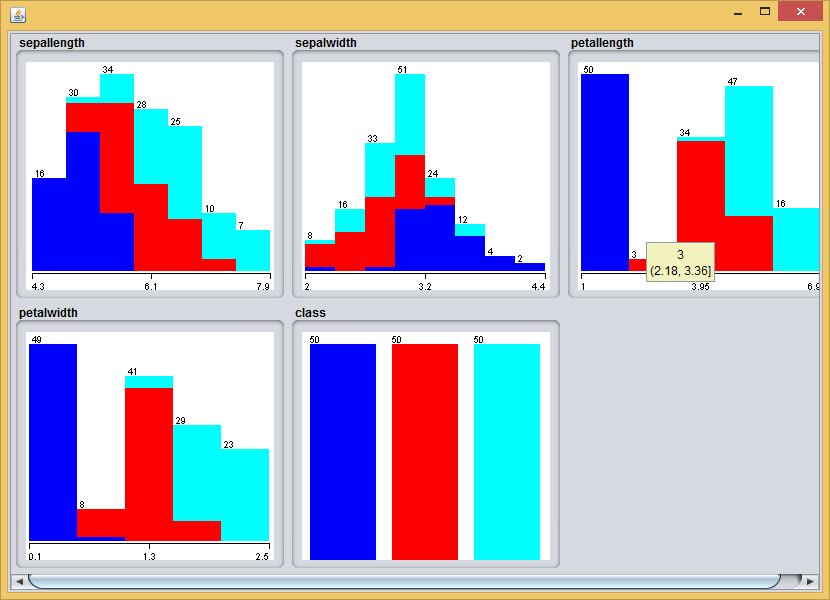
In WEKA, we can visualize data in two forms:

1. For every attribute, there is different Bar graph indicating the number of instances in a particular class.
2. Visualization with each other.

**Visualization:**

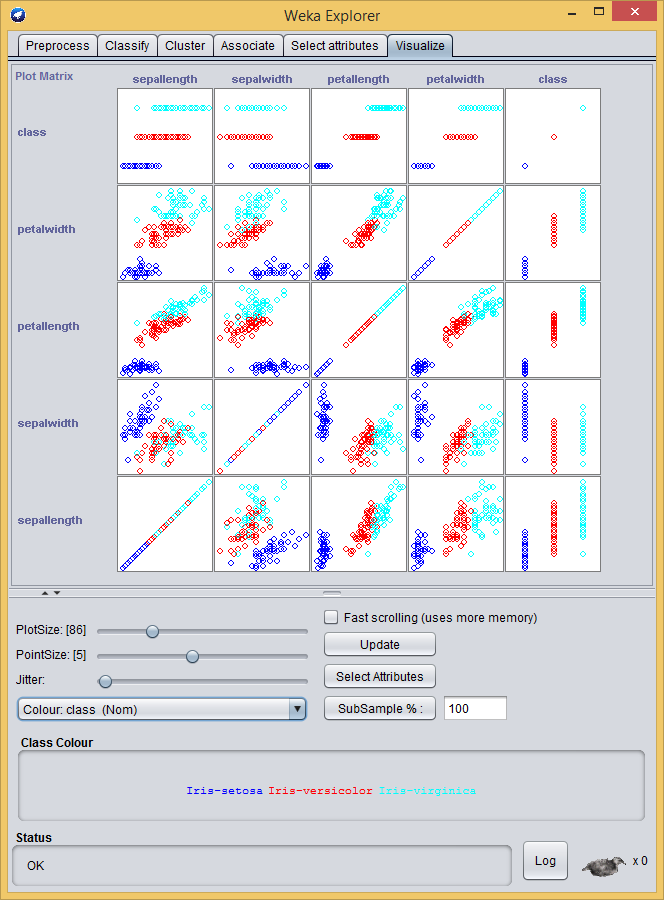
1. **By using Preprocess tab:**

Click on visualize all in preprocess tab.



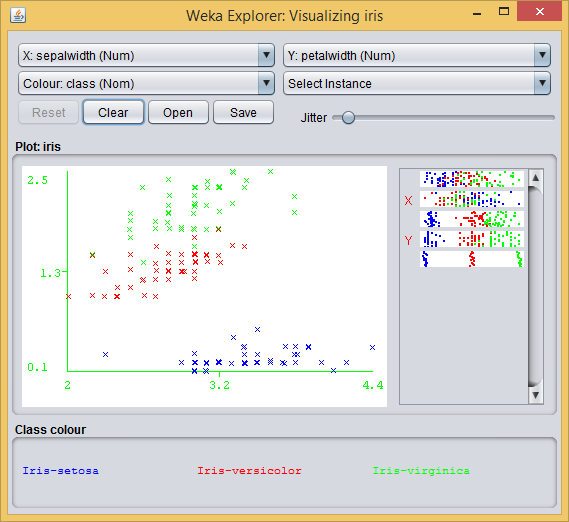
1. **By using Visualize tab:**

To open Visualization screen, click ‘Visualize’ tab.

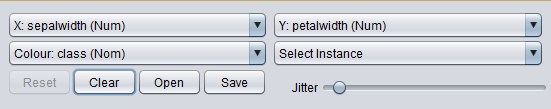


Select a square that corresponds to the attributes you would like to visualize. For example, let’s choose ‘petalwidth’ for X – axis and ‘sepalwidth’ for Y – axis. Click anywhere inside the square that corresponds to ‘petalwidth’ on the left and ‘sepalwidth’ at the top.

A ‘Visualizing Iris’ window appears on the screen.



You can also change the axis, colour of points by using following menu:



In this menu Select Instance is used for selecting a particular area of the graph.