BayelviraApp Documentation.

Table of Contents

1 Introduction.

Bayelvira cytoscape plugin automatically builds Bayesian networks using a file of patterns as input data. To build the network we provide a set of algorithms:

- Naive Bayes.
- Semi Naive Bayes.
- Selective Naive Bayes.
- TAN.
- KDB.

The strength of the inferred relations is highlighted using the edge thickness and depending on the selected algorithm some attributes could be removed or grouped. Measures of the network performance as classifier are also provided.
This plugin is an extension of Elvira system http://leo.ugr.es/elvira/ and aims to ease the process of building Bayesian networks for non-expert users in machine learning.

2 Development.

The plugin has been developed by Víctor Potenciano as member of Project MINER, funded by Campus CEI BioTIC Granada http://biotic.ugr.es/.
Elvira system is a tool to construct model based decision support systems developed by DECSAI (Department of Computer Science and Artificial Intelligence) at University of Granada.
Elvira system website can be found at http://leo.ugr.es/elvira/.

3 Installation.

Installing the plugin from web:
In Cytoscape, go to Apps -> “App Manager”, choose bayelviraApp under “Network generation” category and click on the install button. In case of successful installation, the plugin menu “bayelvira” should appear under “Apps” menu.

4 License.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License version 3. The license can be found at http://www.gnu.org/licenses/gpl.html .

5 Citation.

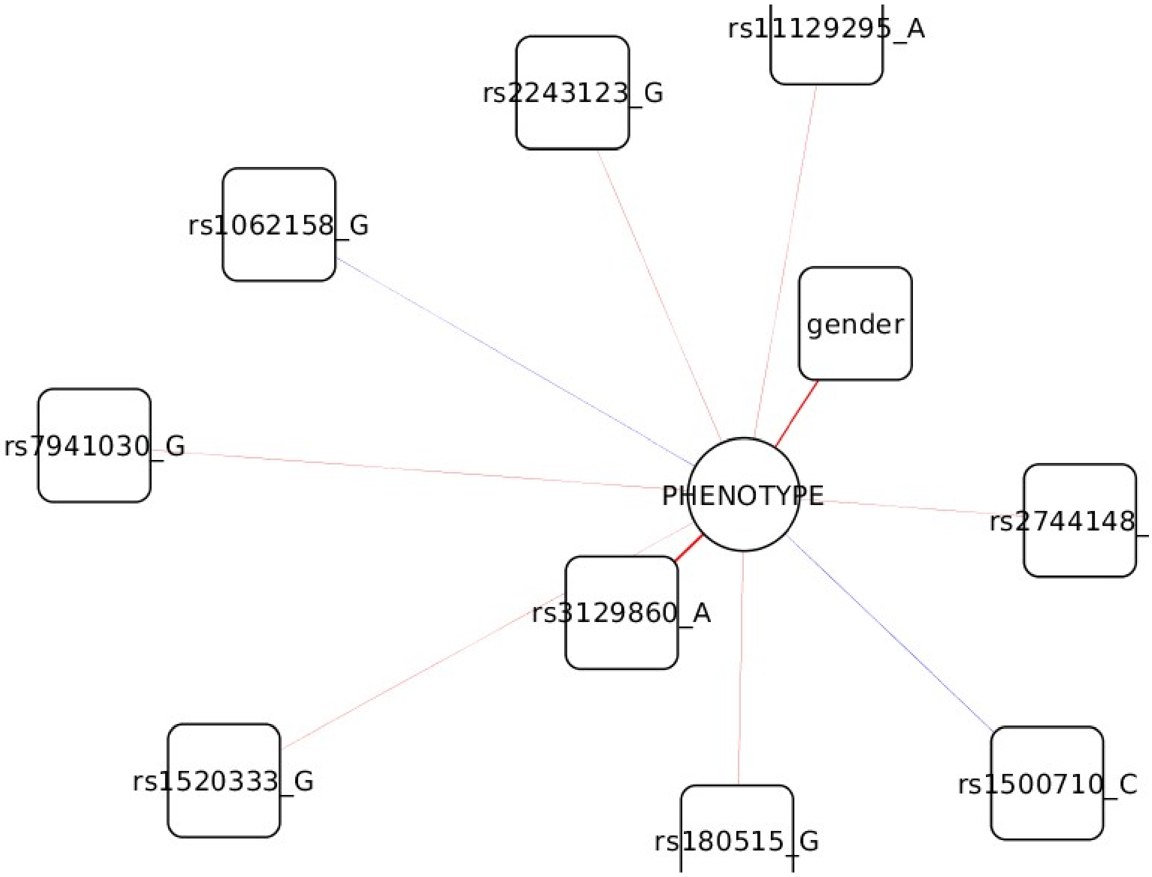
When using the app in your research, please refer to the app webpage.

6 Bayesian networks.

A **Bayesian network**[1] is a probabilistic graphical model (a type of statistical model) that represents

[1] http://en.wikipedia.org/wiki/Bayesian_networks

a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between phenotypes and SNPs. Given genotypes, the network can be used to compute the probabilities of the presence of various phenotypes.



Example of Bayesian network generated by bayelvira plugin.

Formally, Bayesian networks are directed acyclic graphs whose nodes represent random variables in the Bayesian sense: they may be observable quantities, latent variables, unknown parameters or hypotheses. Edges represent conditional dependencies; nodes that are not connected represent variables that are conditionally independent of each other. Each node is associated with a probability function that takes as input a particular set of values for the node's parent variables and gives the probability of the variable represented by the node.

In the simplest case, a Bayesian network is specified by an expert and is then used to perform inference. In other applications the task of defining the network is too complex for humans. In this case the network structure and the parameters of the local distributions must be learned from data. The latter is the purpose of bayelvira plugin, learning the network structure and probability distributions of each node. The learning process of the best network (DAG) given a set of data is an unfeasible task, so we need to make good approximations with faster algorithms that explore a subset of the whole DAG space induced by the input dataset.

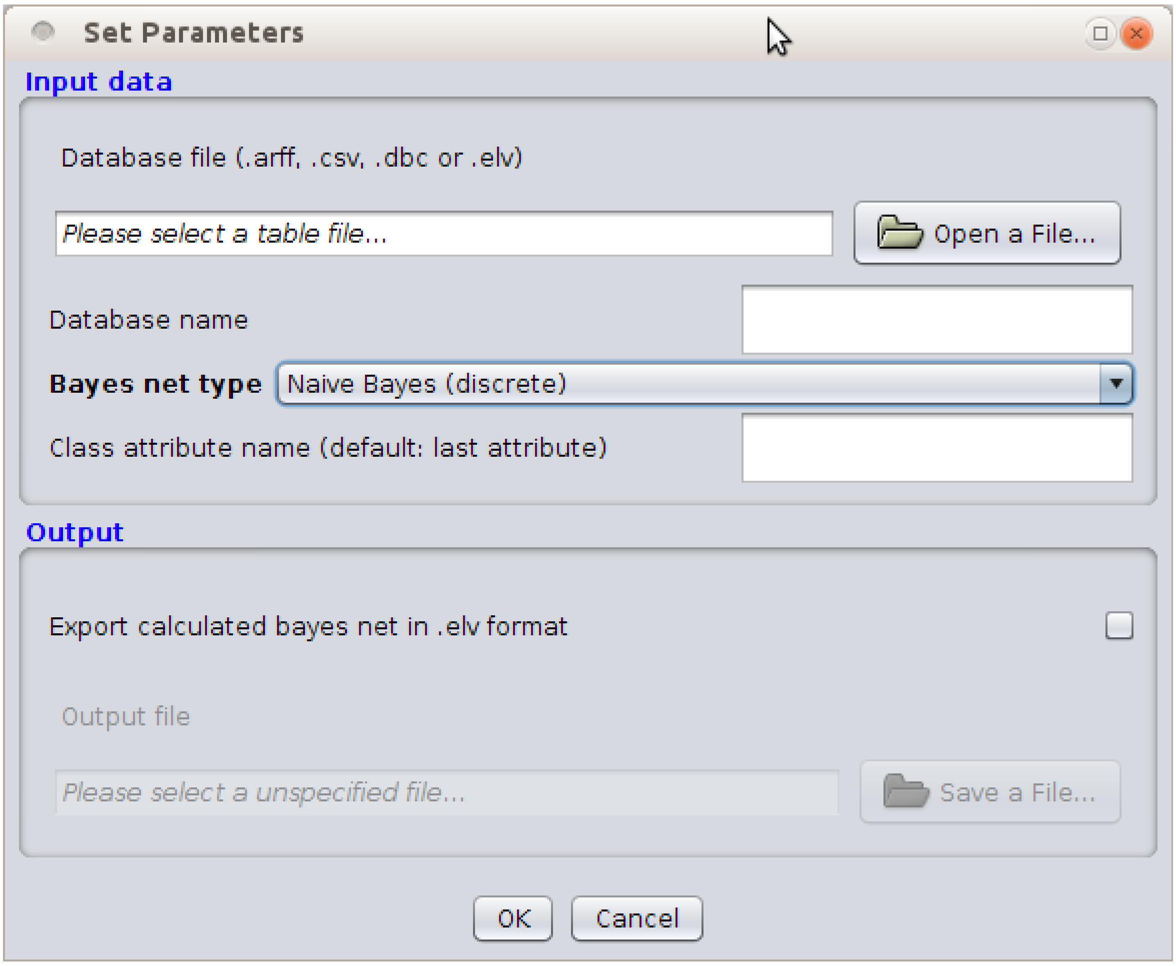
7 Tutorial.

7.1 Getting started.

Create a new empty network before use of bayelvira: File->New->Network->”Empty network” and leaving the default options click on OK button.

7.2 Run bayelvira.

Bayelvira dialog must appear selecting bayelvira menu option under Apps:



Set Parameters

Input data

Database file (.arff, .csv, .dbc or .elv)

Please select a table file...

Database name

Bayes net type Naive Bayes (discrete)

Class attribute name (default: last attribute)

Output

Export calculated bayes net in .elv format □

Output file

Please select a unspecified file...

Bayelvira plugin dialog.

7.3 Input data.

This app makes use of patterns to build a bayesian network. Each pattern also known as an instance, record or example, is characterized by a tuple (x, y), where x is the attribute set and y is a special attribute, designated as the class label (also known as category or target attribute). In the following table we can see an example of a dataset that could be hypothetically extracted from a GWAS study. The data could be used to predict the phenotype of an individual (affected or unaffected) given his genotype.

SNP1	SNP2	SNP3	SNP4	Phenotype
0	0	0	2	affected
1	0	1	1	unaffected
2	2	1	0	unaffected
2	2	1	0	affected
2	2	2	2	affected
1	1	2	1	affected
0	0	2	1	unaffected

In this example each attribute takes discrete values, the genotypes (SNPs) could be 0, 1 or 2 and the phenotype could be *affected* or *unaffected.* To use bayelvira all attribute values must be coded as integers, so *phenotype* coding values must be changed, for example, affected as 1 and unaffected as 0.

Supported file formats.

CSV: Comma separated values with header line. Each attribute is double quoted. An example of CSV file for the previous table example could be:

```
“SNP1”,”SNP2”,”SNP3”,”SNP4”,”Phenotype”
“0”,”0”,”0”,”2”,”1”
“1”,”0”,”1”,”1”,”0”
“2”,”2”,”1”,”0”,”0”
“2”,”2”,”1”,”0”,”1”
“2”,”2”,”2”,”2”,”1”
“1”,”1”,”2”,”1”,”1”
```

CSV files are usually exported from spreadsheets.

Weka ARFF files: Weka[2] is a collection of machine learning algorithms for data mining tasks. The standard for Weka dataset files is ARFF (Attribute-Relation File Format) and the specification for this format can be found at http://weka.wikispaces.com/ARFF. ARFF files with nominal (discrete) attributes can be loaded by bayelvira. An example of ARFF file for the previous data could be:

```
@relation gwas

@ATTRIBUTE SNP1 {0,1,2}
@ATTRIBUTE SNP2 {0,1,2}
@ATTRIBUTE SNP3 {0,1,2}
@ATTRIBUTE SNP4 {0,1,2}
@ATTRIBUTE phenotype {0,1}

@data
0,0,0,2,1
1,0,1,1,0
```

[2] http://www.cs.waikato.ac.nz/ml/weka/

```
2,2,1,0,0
2,2,1,0,1
2,2,2,2,1
1,1,2,1,1
```

Legacy Elvira files: For compatibility with Elvira system **DBC** (database cases) and **ELV** (Elvira network) files can also be loaded.

7.4 Bayesian network learning algorithms.

Naive Bayes.

This is the simplest algorithm. It assumes total independence between attributes and the generated network has as many nodes as attributes and links from each attribute to class node.

Semi Naive Bayes.

This algorithm extends the NB classifier in order to detect the dependencies between attributes. Some attributes could be removed and others could be grouped under dependence assumption between them.

Selective Naive Bayes.

The Selective Bayesian classifier is a variant of the naive method that uses only a subset of the given attributes in making predictions. It tries to achieve improved accuracy in domains with redundant attributes.

TAN.

Tree augmented Naive Bayes. Learns a maximum weighted spanning tree based on the conditional mutual information between two attributes given the class label. Then, the arcs in the tree are oriented by choosing a root and the model is completed by adding a link from the class to each attribute.

KDB.

K-dependence bayesian classifier. Introduced the notion of k-dependence estimators, from which the probability of each attribute value is conditioned by the class and, at most, k other attributes. Throughout the KDB algorithm it is possible to construct classifiers across the whole spectrum, from the NB structure to the full BN structure, by varying the value of k, i.e. the maximum number of parents that every attribute can have. In our implementation we have limited **k** up to 5.

Discrete and mixed classifiers.

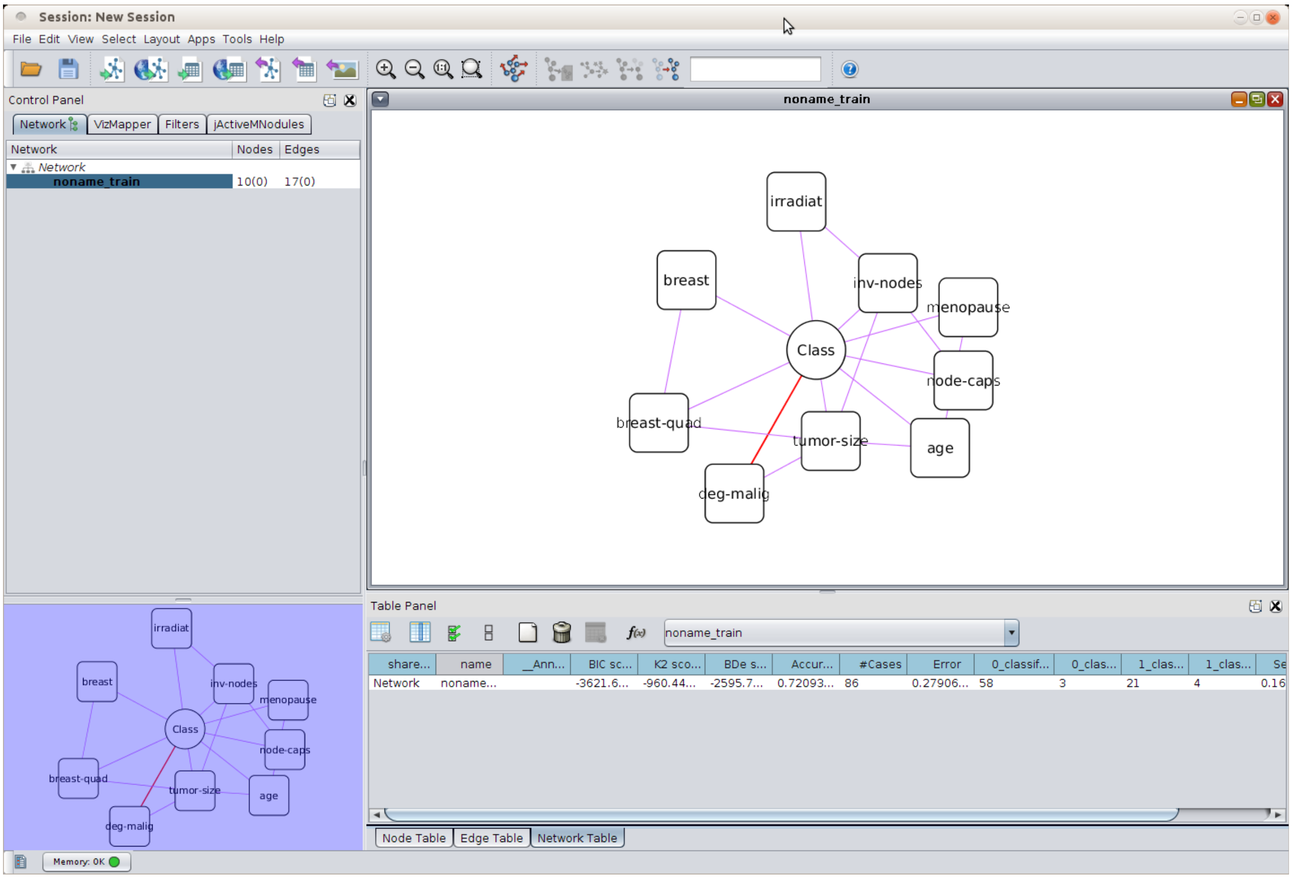
Use discrete classifiers for your datasets, in CSV or ARFF file formats. There are mixed classifiers suitable only for DBC datasets. In mixed datasets real attributes are permitted, but the class attribute must be discrete.

7.5 Output.

If desired, the resulting learned network could be saved and loaded later since some algorithms are more time consuming for large datasets. To save the resulting network check “Export calculated bayes net in .elv format” option and set the file name where to save it.

Interpreting the resulting network.

During network learning, the dataset is divided into training and test sets. The training set keeps the 70% of patterns and is used to build the model. At the end of the learning process, we use the test set (remaining 30% of patterns) to assess the network accuracy.



Network	Nodes	Edges
Network		
noname_train	10(0)	17(0)

noname_train

share...	name	_Ann...	BIC sc...	K2 sco...	BDe s...	Accur...	#Cases	Error	0_classif...	0_clas...	1_clas...	1_clas...	Se
Network	noname...		-3621.6...	-960.44...	-2595.7...	0.72093...	86	0.27906...	58	3	21	4	0.16

Example of TAN network learned from breast-cancer dataset (included in weka).

The resulting network has the following features:

- Class node is drawn as an oval.
- Attribute nodes are drawn as rounded corner rectangles.
- Edge thickness represents the strength of the relationship.
- The color of the edge represents “positive correlation”, “negative correlation” or

"uncorrelated" as red, violet or black respectively.

Feel free to change the layout to display the network properly.

The measures of the network as classifier are given in Network Table columns:

- Accuracy: the percentage of correctly classified instances.
- #Cases: amount of cases in test set.
- Error: the percentage of incorrectly classified instances.
- Sensitivity (binary classification only): relates to the ability to identify positive results.
    - Sensitivity = #TP/(#TP+#FN)[3].
- Specificity (binary classification only): relates to the ability to identify negative results.
    - Specificity = #TN/(#TN+#FP).
- Confusion matrix entries: show the amount of misclassified instances per class. Column names for this entries follow the pattern X_classified_as_Y.

8 Troubleshooting.

- Make sure that you run bayelvira over a new empty network. If you run bayelvira over an existing network the error *'column already exists with name: 'node_type' with type: class java.lang.String'* is raised.
- There is an extra Bayesian network classifier called 'Class tree naïve', not documented yet. This classifier is unstable and may cause some problems.

[3] TP=true positive, TN=true negative, FP=false positive, FN=false negative