Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Máster Universitario en Ciencia de Datos

Master's Final Project

Extending the Bnclassify R package: Bayesian Network Classifiers with Continuous Variables

Author: SHANSHAN CHENG

Supervisor: BOJAN MIHALJEVIC, PEDRO LARRAÑAGA

Madrid, July 2020

This Master's Final Project is submitted to the ETSI Informáticos at Universidad Politécnica de Madrid in partial fulfillment of the requirements for the degree of MSc. in Data Science.

Master's Final Project
MSc. In Data Science
Title: Extending the Bnclassify R package:
Bayesian Network Classifiers with Continuous Variables

July 2020
Author: Shanshan Cheng

Supervisor: Bojan Mihaljevic and Pedro Larrañaga
ETSI Informáticos
Universidad Politécnica de Madrid

Abstract

Nowadays, most of the open-source software for Bayesian networks only handles discrete variables. This limitation makes it difficult to use Bayesian networks in real life. The original bnclassify package provides the code that deals with the *Bayes multinomial network* (BMN). This kind of Bayesian network is not capable of handling continuous variables, if a data set contains continuous variables, it will be discretized, therefore, a loss of information will be introduced. The goal of this work is to extend the package and make it able to deal with continuous variables. To avoid the loss, this work assumes all variables follow the Gaussian distribution, therefore, new functionalities are added to the package including structural learning, parameter learning, and prediction with *conditional Gaussian network* (CGN). In order to determine the correctness and the efficiency of the new implementations called predictCGNs and GaussianImplement, two experiments are designed in this work. The experimental results demonstrate the usability of the program and there is no inconsistency between the new implementations and Bayesian networks.

Index

Index of figures

Index of tables

1. Introduction

A supervised algorithm is the most used technique for data analysis and pattern recognition. Supervised learning can be divided into two tasks, regression, and classification. Regression is used when the output of the problem is a continuous variable. Classification is when the output is a discrete variable, a category of a class. There are different classifiers to solve the classification problem, among which are *Bayesian networks* (BN), also known as *Belief networks.* This network is a directed acyclic graph composed of nodes and arcs. This graph is based on the probabilistic graphical model (PGM) for representing conditional probability distributions.

The kind of Bayesian network that works with discrete variables is known as a Bayesian multinomial network (BMN). There are different BMN-based classifiers: naive Bayes, tree augmented Bayesian network, k-dependence Bayesian classifier, and semi naive Bayes. All the classifiers mentioned before handle a discrete input variable, this means if there is some continuous variable, it will be discretized, therefore, losing information [1].

To avoid losing information, another alternative is assuming that data follows multidimensional Gaussian distribution conditioned on each value of the class. This technique is known as the Conditional Gaussian Network (CGN). CGN provides a network model that learns data composed of continuous and discrete variables.

This work is elaborated on the basis of [2]. The main purpose of the project is to extend the package bnclassify allowing the Bayesian network classifier to handle continuous variables. Therefore, the bnclassify package is required to be installed. In order to achieve the goal, I suppose all the predictor variables follows the normal distribution, then I can estimate the probability distribution by using the new functions designed in this work.

The rest of this work is structured as follows: I begin by describing the state of the art of the Bayesian classifiers and the CGNs model. In the third section, I will describe the learning process to build a complete Bayesian network classifier. In the fourth section, I will introduce the changes elaborated in this work which allows the classifier to work with the continuous variables. In the fifth section, I will discuss the result obtained from the new implementation and compare the result obtained from the BMNs model and CGNs model. In the sixth section, a brief conclusion about this work and possible future works.

2. Bayesian networks

Bayesian networks have become a popular algorithm for reasoning with uncertainty and probabilistic modeling in artificial intelligence. The use of Bayesian networks has led to many successful products or applications in medical diagnosis, natural language understanding, image processing, object recognition, and prediction. Therefore, Bayesian network is involved in a number of areas.

The biggest problem with other algorithms such as neural network is that they require large amounts of data. This problem can lead to undesirable results, for example, in the driverless field, where it is difficult to have large amounts of data for every emergency or special situations. Kai Xu, the founder of Baidu IDL, says "More causal reasoning needs to be added, and this framework of causal reasoning is derived from Bayesian networks" [3]. Bayesian networks use data to make an inference-based decision. The reasoning and the computational process characteristics of Bayesian networks make it stand out from other algorithms.

In recent years, adapting Bayesian networks to big data has become a problem that researchers want to solve. Many new methods and theories are proposed during this period, for example, Bayesian learning is combined with many other pieces of knowledge such as *parallel computing.* It can be expected that Bayesian learning will definitely have better results in the future.

2.1 Bayes theorem

Before introducing the Bayesian formula, we should understand four concepts: a posteriori probability, a priori probability, likelihood estimation, and evidence. In order to facilitate the understanding of the four concepts, I give an example: the neighbor was going to a park, he had the option of walking or driving (cause) and then he took a while (outcome) to get to the park by either way [4].

- Posterior probability: know the outcome and seek the cause.

Assuming we know the neighbor took 1 hour to get to the park, we are not sure 100% whether he chose walking or driving. If the neighbor took 5 hours to get to the park, the probability of choosing walking is more than choosing driving. Therefore, we estimate the probability of the cause given the outcome.

$$p(transportation|time) = p(cause|outcome) = p(\theta|x)$$

- A priori probability: seek the cause without taking into account the outcome.

Now, we do not think about the time he spent on the road but we only consider his personal situation. Assuming the neighbor is an athlete, we can estimate the probability of walking is more than driving. Therefore, the cause is not relevant to the outcome since we estimate the cause before the outcome happened.

$$p(transportation) = p(cause) = p(\theta)$$

- Likelihood estimation: know the cause and seek the outcome.

In the case of choosing walking, the outcome can be half an hour. If he chooses driving, the probability of spending thirty minutes is less than walking. Therefore, we estimate the probability of the outcome based on the cause

$$p(time|transportation) = p(cause|outcome) = p(x|\theta)$$

- Evidence: seek the outcome without taking into account the cause.

We ignore the transportation he chooses to get to the park, we only consider the time of each arrival at the park

$$p(time) = p(outcome) = p(x)$$

The Bayesian equation is:

$$p(\theta|x) = \frac{p(x|\theta)*p(\theta)}{p(x)} \quad (1)$$

In the formula, we can intuitively see that the result is not a specific value, but a value that can be affected by other factors. For instance, the probability of choosing to walk knowing he took 20 minutes to get to the park is high if we have data that shows he chose to drive multiple times and arrived at the park in half an hour.

2.2 Basics of Bayesian networks

Bayesian networks are *Probabilistic Graphical models* (PGMs) that represent the conditional relationship between random variables through a *directed acyclic Graph* (DAG). The random variable in a DAG is also known as node. If two nodes are connected by a single arc, one of the nodes will be parent node (cause) and the other node will be child node (outcome). The probability linked to each node depends on the value given by its parent node. The arcs that connect the nodes represent the conditional independencies among the variables [1].

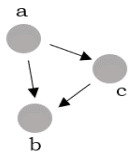

Figure 1. DAG

As we can see in Figure 1, the node c is a random variable conditioned on the probability of a. The node b is a random variable conditioned on the probability of a and c. The joint probability of this DAG is:

$$P(Sp, A, Sn) = P(a) * P(c|a) * P(b|c, a) \quad (2)$$

Therefore, we have two types of variables:

- Class variable: it is the variable that does not depend on any node of the graph. For example, node a in Figure 1. We can also understand this variable as the target variable in the classification task. A class variable consists of different categories and the classification task predicts the category given instances.

- Predictor variables: the variable that is conditioned on other variables in a DAG. For example, the node b and the node c in Figure 1 are example of predictor variables. Each predictor variable is a conditional probabilistic distribution:

$$P(\mathrm{x}|c) = P(\mathrm{x}|Pa(\mathrm{x})) \tag{3}$$

  where x is the random variable, and $Pa(x)$ is a set of variables that are parents of the variable x_i. With the previous formula we are able to classify the class of an instance:

$$P(c|x) \propto P(c,x) = p(c)p(x|c) = p(c)\prod_{i=1}^{n} P(x_i|Pa(x_i)) \tag{4}$$

In the second part of this section, I present the D-separation [5]. D-separation is the method that determine whether the two nodes are conditionally independent. Before discussing this method in more detail, we need to understand the three possible subgraphs which form a *Bayesian networks:*

- Head-to-head: c is unknown so a and b are independent.

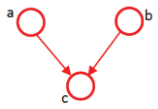

Figure 2. Head-to-head subgraph

- Tail-to-tail: a and b are independent given c

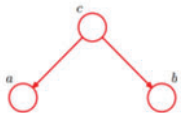

Figure 3. Tail-to-tail subgraph

- head-to-tail: a and b are independent given c

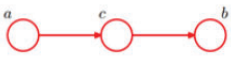

Figure 4. Head-tail subgraph

The d-separation is a criterion for deciding whether a variable A is independent of another variable B, given a variable c. All of them belong to the same *Directed Acyclic Graph*. Node A and B are independent if they satisfy one of the following conditions:

1. If A, B, and c form the "head-to-tail" or "tail-to-tail" subgraph, A and B both pass through C.

2, If A, B, and C form the "head-to-head" subgraph, A and B do not pass through C and C's descendants.

2.3 Bayesian networks classifiers

This subsection presents the state-of-the-art Bayesian classifiers: Naive Bayes (NB), Tree Augment Naive Bayes (TAN), k-dependence Bayesian classifier (k-DB), and Semi-naive Bayes (SNB). All the classifiers are supported by CGNs paradigms.

2.3.1 Naive Bayes

Naive Bayes [6] is a simple but powerful predictive algorithm. The reason why this structure is called Naive Bayes is that it assumes all the predictor variables are conditionally independent given the class variable. This is a simple assumption that cannot be satisfied in real cases, but the technique is still very effective for a complex problem. The structure of Naive Bayes [2] is:

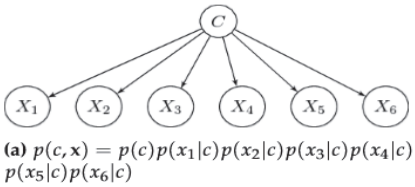

Figure 5. Naive Bayes Network structure

Where the parent variable is the class variable c and the rest of them are predictor variables.

Using the Bayesian equation (1) and assuming each variable is conditionally independent of each other [7]:

$$p(c|\mathrm{x}) = \frac{p(c) * p(\mathrm{x}|c)}{p(\mathrm{x})} = \frac{p(c)}{p(\mathrm{x})} \prod_{i=1}^{n} p(\mathrm{x}_i|c) \tag{5}$$

The prediction process in Naive Bayes is to multiply the priori probability of the class variable with the conditional probability, $p(\mathrm{x}\,|\,c)$, for each predictor variable. Since $p(\mathrm{x})$ is the same for all categories, the Naive Bayesian classification criterion turns out to be:

$$c^* = arg\ max_c\ p(c) * \prod_{i=1}^{n} p(x_i|c) \tag{6}$$

Where,

$$p(c) = \frac{|D_c|}{D} \tag{7}$$

D_c denotes the number of instances that have label c in the data set D. For example, the iris data set provided by UCI contains 50 instances of 'setosa', 50 instances of 'Versicolor', and 50 instances of 'virginica'. Then, the a priori probability is 1/3 for each category. n is the number of predictor variables.

2.3.2 Tree Augmented Naive Bayes

The TAN model [8] allows each node, except the class variable, can depend on the class and at most one another predictor variable [1]. Therefore, this type of structure breaks the assumption of the NB structure where all the variables are conditionally independent of each other given the class.

Here is an example of structure of TAN Bayes:

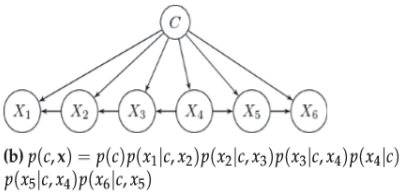

Figure 6. TAN structure

The key of TAN is that this type of model is a Tree structure [1] [9]. This Bayes network was proposed by Friedman. In order to carry out this approach, he designed the algorithm that follows the general outline of Chow-Liu Tree. The procedure is shown below [8]:

1. We need to calculate the mutual information between each pair of variables. The mutual information measures the correlation score between two random variables. After obtaining the scores table, we have enough information to construct the model.

2. Construct the maximum weight spanning tree: we add edges between the variables according to the scores table in descending order until n-1 edges have been selected. n is the number of variables in the structure. The algorithm will reject the current edge if they form a cycle with the previous edges and then, it continues with the next edge.

Figure 7 shows an example of a maximum weight spanning tree. The tree is composed of 5 variables and 4 edges.

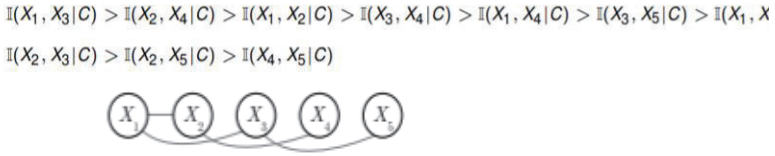

Figure 7. Maximum weight spanning tree

3. The purpose of this step is to determine the direction of the edges. Any of the variables can be selected as the root node. Here we select x_1:

Figure 8. x_1 as the root node

4. Add the class node for each predictor variable node:

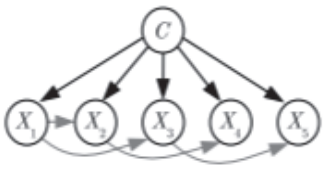

Figure 9. TAN structure

After constructing the TAN, we can estimate the posterior probability for each class label and it is calculated as a product of prior probability, the conditional probability of the root node given the class label and the conditional probability of the predictor variables given the parents node and the class label:

$$p(c|\mathrm{x}) \propto p(c) * p(\mathrm{x}_r|c) \prod_{i=1, i\neq r}^{n} p\left(\mathrm{x}_i|c, \mathrm{x}_{j,i}\right) \tag{8}$$

Where x_r denotes the root node. And $\mathrm{x}_{j,i}$= Pa(x_i) for any $i \neq r$

Moreover, TAN is then adapted by Pérez et al [1] to work with the conditional Gaussian distribution. The new version of Pérez et al. calculates the conditional mutual information between the continuous variables given the class variable. Next show the proposition 2 from the article [1] published by Pérez et al:

$$I\left(\mathrm{x}_i; \mathrm{x}_j|c\right) = -\frac{1}{2}\sum_{c=1}^{r} p(c) log\ (1 - \rho_c^2\left(\mathrm{x}_{i,} \mathrm{x}_j\right) \tag{9}$$

Where $\rho_c^2(x_{i,}x_j)$ denotes the correlation coefficient between the variable x_i *and* x_j given class variable c:

$$\rho_c^2\left(x_{i,}x_j\right) = \frac{\sigma_{ij|c}}{\sqrt{\sigma_{j|c}^2 \sigma_{i|c}^2}} \tag{10}$$

$\sigma_{ij|c}$ is the covariance between the variables x_i *and* x_j given class c and $\sigma_{j|c}^2$ is the variance of the variable x_j given the class variable c.

According to [8], the whole procedure of searching the structure has time complexity $O(n^2N)$ where n is the number of predictor variables in the structure, and N is the number of instances in the dataset.

In the second part of this section, I introduce two more methods: *Hill-Climbing Tree Augmented Naive Bayes* (TAN-HC) and *Hill-Climbing Super-Parent Tree Augmented Naive Bayes* (TAN-HCSP).

- Hill-Climbing Tree Augmented Naive Bayes

Pazzani proposed an extended version of TAN in 2002 [10]. The algorithm proposed follows the main restriction of TAN structure: each variable except class variable may have no more than one predictor parent. The differences between TAN and TAN-HC are reflected in the structure learning. TAN-HC presents the following learning steps [11]:

1. The structure initializes with naive Bayes and a full set of *orphans*. Orphans contains the nodes that exist in the current structure except the parent variable. In the example shows below, $A = \{A_1, A_2, A_3, A_4\}$ are members of Orphans:

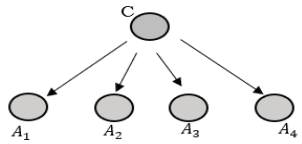

Figure 10.NB structure. Orphans = {A1, A2, A3, A4}

2. Add all the possible arcs that go from A_i to A_j, where $A_i \neq A_j$ and $A_j \in Orphans$.

Each arc added forms a new Bayes structure.

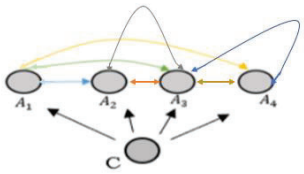

Figure 11. Possible structures

3. The arc added that allows the structure to obtain the highest accuracy will be retained and the variable pointed by this arc will be removed from the Orphans set. $A = \{A_1, A_3, A_4\}$

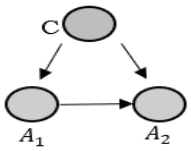

Figure 12. Selected structure

Then, go back to step 2. The process will end when the Orphans set contains only one variable and returns the current classifier.

In terms of the complexity of computing, $O(n)$ arcs may be added and $O(n^2)$ classifiers are evaluated for each arc added. Hence, the complexity of HC-TAN is $O(n^3)$

- Hill-climbing super-parent tree augmented naive Bayes

As I mentioned before, HC-TAN requires $O(n^3)$ to search the best TAN structure. In order to minimize the complexity, Pazzani introduced *SuperParent* (SP) technique [10]. This technique allows us to obtain improvement in accuracy and less computational time compared to HC-TAN. The SP technique consists of finding the best root node and then finding the best child of the root, instead of adding all possible arcs. The procedure is described below [11]:

1. The structure initializes with naive Bayes and a full set of Orphans, $A = \{A_1, A_2, A_3, A_4\}$.

2. Find the best root node by estimating the accuracy of each variable given the class. We assume A_1 is the root node that obtains the best improvement in the accuracy.

3. Find the best child variable of the root node by adding arc from the root node to each orphan and then, each arc added is evaluated by estimating the accuracy of the network

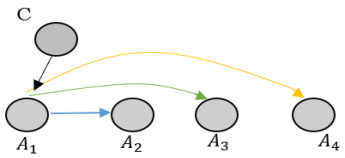

Figure 13. Possible child variables

Suppose A_2 is the best child variable, A_2 will be retained in the current structure and removed from the Orphans set. Then, repeat the step 2.

If no improvement in accuracy is presented and the Orphans set contains more than one variable, go to step 4. If |O|=1, the current classifier is returned since Orphans set only contains the root node.

4. Add all possible arcs from child variables to the variables contained in the Orphans set. Then, each arc added is evaluated by estimating the accuracy. The algorithm will finish when|O|=1.

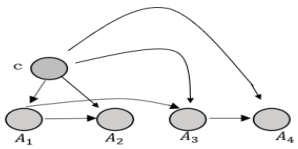

Figure 14. HC-SP-TAN structure

The complexity for TAN-HCSP is $O(n^2)$ since the arcs may be added is $O(n)$ and $O(n)$ classifiers are evaluated for each arc added. Hence, the complexity is lower than TAN-HC.

2.3.3 k-Dependence Bayesian Classifiers

k-Dependence Bayesian, k-DB, is an extended version of the TAN structure that allows each predictor variable to have a maximum of k parents variables plus the class variable.

In order to construct k-DB structure, Sahami [12] proposes an algorithm which uses the mutual information between the predictor and the class variable, $I(\mathrm{x_i}; c)$ and the class conditional mutual information between the predictor variables, $I(\mathrm{x}_i; \mathrm{x}_j|c)$ to construct k-DB structure. The procedure of the algorithm is shown below [13]:

First, the variables are ranked by comparing the scores of the mutual information between the predictor variable and class c. For example, we have 4 predict variable and we suppose the score order is $I(\mathrm{x}_1; c) > I(\mathrm{x}_2; c) > I(\mathrm{x}_3; c) > I(\mathrm{x}_4; c)$, so we have $\{\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \mathrm{x}_4\}$

Second, once we have the variables ranked, we start to work with the highest variable, x_i. Continuing with the example the highest variable is x_1. We compute the conditional mutual information among the x_1 and the rest of the predict variables given class c, $I(\mathrm{x}; \mathrm{x}_j|c)$ where $j \neq i$. Then, the k parents variables which have the highest mutual information score are selected. Moreover, we must take into account x_i will have $i - 1$ predictor parents when $i \leq k$ and, those variables where $i > k$ must have exactly k parents. In this example, if k = 2, the predictor variable x_1 will only have the class variable as a parent, and the variable x_2 will have one predictor and one class variable as parents [14]. Repeat the second steps and the loop will finish when i is equal to n-1 where n is the number of predictor variables.

Figure 15 shows the k-DB structure of this example where k = 2.

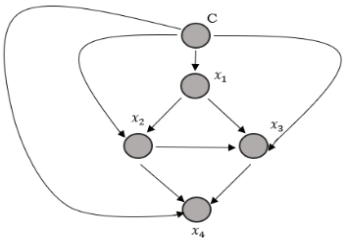

Figure 15. 2-DBstructure

Finally, the Bayes equation turns out to be:

$$p(x) \propto p(c) * \prod_{i=1}^{n} p\left(c, x_{i,1}, \ldots, x_{i,p}\right) \tag{11}$$

Where $p = min\ (i\text{-}1,\ k)$. For example, the factorization of Figure 15 is:

$$p(c|x) \propto p(c)p(x_1|c)p(x_2|c, x_1)p(x_3|c, x_1x_2)p(x_4|c, x_2, x_3) \tag{12}$$

2.3.4 Semi-Naive Bayes

When two variables are highly correlated, the independence criterion of Naive Bayes can cause them to lose this relationship and then, the accuracy of the classifier may deteriorate. Semi-Naive Bayesian classifier is a classification method that takes into account the dependency relationships among the variables. Therefore, this type of classifier is useful when it is difficult to satisfy the conditional independence assumption of NB structure.

The key of SNB is a new kind of variable called joint variable,y_k. The joint variable is formed by a set of original variables that are highly correlated with each other and each of the variables cannot appear more than once in the SNB structure

Pazzani presents two different ways to construct the SNB structure, *forward sequential selection and joining,* FSSJ*,* and *backward sequential elimination and joining,* BSEJ.

- Forward Sequential Selection and Joining

The algorithm starts with an empty structure. Two operations are considered to perform the structure searching process [15] [13]:

a. Adding each variable not used by the current structure and the variable added u
被 is treated as conditionally independent of all other iused in the structure.

b. Joining each variable not used by the current structure with each feature present in the structure.

At each step in the searching process, the result of the operation a and b is a multiple structure and all possible structures will be evaluated by estimating the accuracy. Then, the structure with high accuracy is chosen and used to perform the next step. option a and b will be repeated until no improvement in the accuracy.

- Backward sequential elimination and joining

The algorithm starts with a Naïve Bayes structure treating all predictor variables as conditionally independent given the class variable. Like the FSSJ algorithm, BSEJ algorithm has two operations [15] [13]:

a. Each pair of variables in the current structure is replaced with a new variable that joins the pair of variables.

b. Deletes each attribute in the current structure.

All possible structures formed in each step are evaluated by estimating the accuracy and the structure which has the best improvement will be retained for the next iteration. The algorithm will finish when any structure provides a better accuracy.

In terms of complexity, both techniques require a total of $O(n^3)$ structures in the worst case [15].

2.4 Conditional Gaussian Bayesian networks

The multinomial Bayesian network is a kind of Bayesian that works with discrete variables. As I mentioned before, if the random variable is continuous, all BMN-based classifiers will discrete the continuous variable, so the loss of information is introduced. Another way to handle the continuous variable is assuming that the continuous variables follow the CGNs [1] [16] [17]

$$P\left(x|Pa(\mathrm{x}_i)\right)\sim\mathcal{N}\left(m_{i|c},{\sigma^2}_{i|c}\right) \tag{13}$$

Where $m_{i|c}$ is the expectation of the distribution for x_i given by class c and its parents $Pa(x_i)$; ${\sigma^2}_{i|c}$ is the conditional variance of the distribution. $m_{i|c},{\sigma^2}_{i|c}$ can be generated as [18] [19] :

$$m_{i|c} = \mu_{i|c} + \sum_{j=1}^{n_i} \beta_{ij|c} * (\mathrm{x}_j - \mu_{j|c}) \tag{14}$$

$\mu_{i|c}$ is the regression intercept of the node i given by class c. $\beta_{ij|c}$ is the regression coefficient of the variable x_i on x_j conditioned to the class c. n_i is the number of parents of the node i. And,

$$\sigma^2_{i|c} = \frac{\sum_{i=1}^{m}(\mathrm{x}_i-\bar{x_\iota})}{m-1} \tag{15}$$

x_i is the predictor variable. $\bar{x}$ is the mean of the predictor variable x_i. m is the number of data points. Thus, the distribution of the node i is a conditional Gaussian with $m_{i|c}$ and $\sigma^2_{i|c}$.

Finally, the density probability function can be used for continuous data:

$$p(x|c) = \frac{1}{\sqrt{2\pi}\sigma_{i|c}} exp\left(\frac{-(\mathrm{x}_i-m_{i|c})^2}{2\sigma^2_{i|c}}\right) \tag{16}$$

The equation of Bayes turns out to be:

$$P(c|\mathrm{x}_i) \propto p(c)p(\mathrm{x}_i|c) = p(c)\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma_{i|c}} exp\left(\frac{-(\mathrm{x}_i-m_{i|c})^2}{2\sigma^2_{i|c}}\right) \tag{17}$$

3.Learning Bayesian networks

The process can be divided into structural learning and parameter learning.

3.1 Structural learning

The objective of the learning process is to find a Bayes structure from the data that allows us to obtain high accuracy in the classification and short computation time. There are two ways to learning the structure [1]:

a. Constructing the structure depending on the conditional independence relation between the predictor variables.

b. Searching all possible structures from data and each structure will be evaluated by estimating the accuracy. The structure with high accuracy will be chosen.

The option b is NP-hard since multiple structures can be constructed from data, and the complexity increases depending on a number of predictor variables. Therefore, heuristic search algorithms are needed to reduce computation time. One of the algorithms that are commonly used is called Hill-Climbing. Pazzani improves the Tree Augment Naive Bayes with this algorithm. You can see more details in section 2.3.2.

Penalizing the score of dense structure is another way to reduce the complexity of the classifiers [2]. Among other penalizing techniques, *Akaike information criterion* (AIC) and *Bayesian information criterion* (BIC) are used in this work

3.2 Parameter learning

The parameter of a graph, θ, may be assumed to follow a certain distribution [20] [21]. Different parameters can produce different probabilities so the accuracy of the classifier is changed depending on the value of the parameter used.

The parameters [1] needed in Bayesian Gaussian network are an array of covariance matrices, $\sum_{:} = (\sum_{.}^{1}, \ldots, \sum_{.}^{r}.)$ and mean vector$\mu = \{\mu^1 \ldots \mu^r\}$. Where r is the category of the class c. The method to compute both parameters is described in section 2.4. One of the advantages of CGNs is the fewer number of parameters needed to model the continuous random variables.

4. Development description

The purpose of this work is to extend the *bnclassify* package allowing the classifiers of this package handle continuous variables without losing the information.

In section 4.1, I will introduce the bnclassify and talk about the functionalities that this package provides. Then, I will describe the new functionalities and adaptions that I have made in this work in section 4.2

4.1 bnclassify package

The bnclassify package provides algorithms for learning Bayesian network from data. Once the network is learned, we can evaluate this model by predicting accuracy. The learning process can be divided into two parts: structure learning and parameter learning. The second one is optional.

- Structure learning

The bnclassify provides the following state-of-the-art algorithm for learning the structure:

- Naive Bayes
- Tree-Augmented Naive Bayes:
  - Chow and Liu Tree-Augmented Naive Bayes
  - Hill-Climbing Tree-Augmented Naive Bayes
  - Hill-Climbing Super-Parent Tree-Augmented Naive Bayes
- k-Dependence Bayesian classifier
- Semi-Naive Bayes
  - Forward sequential selection and joining, FSSJ
  - Backward sequential elimination and joining, BSEJ
- averaged one-dependence estimator

A table is provided in the [2] to lists the algorithms mentioned before with the corresponding search algorithm used, score, feature selection, and function.

Structure	Search algorithm	Score	Feature selection	Function
NB	-	-	-	nb
TAN/FAN	CL-ODE	log-lik, AIC, BIC	-	tan_cl
TAN	TAN-HC	accuracy	-	tan_hc
TAN	TAN-HCSP	accuracy	-	tan_hcsp
SNB	FSSJ	accuracy	forward	fssj
SNB	BSEJ	accuracy	backward	bsej
AODE	-	-	-	aode
kDB	kDB	accuracy	-	kdb

Table 1. Implemented structure learning algorithms in bnclassify package

As we can see in the table, NB is a very simple network that does not need any method and the structure is constructed given the predictor variables. The code to obtain the structure is shown below:

```
1.  n <- nb ('class', dataset = car)
```

where the class is the name of the class variable and the car is the data used in this example.

TAN is implemented with three different search algorithms: Chow and Liu Tree-Augmented Naive Bayes is produced by using *Chow-Liu's algorithm for one-dependence estimators* (CL-ODE) [8] as a search algorithm, Hill-Climbing Tree-Augmented Naive Bayes is produced by using TAN-HC and Hill-Climbing Super-Parent Tree-Augmented Naive Bayes is produced by TAN-HCSP. The first one uses two different scores to penalize the dense structure: AIC, and BIC. The code to construct the TAN structure is:

```
1. tan_cl('class', dataset = car, score = 'aic', root = 'buying')
2. tan_hc('class', dataset = car, k = 5, epsilon = 0.01, smooth = 0)
3. tan_hcsp('class', dataset = car, k = 5, epsilon = 0.01, smooth = 0)
```

where the root is optional, k is the number of folds used in the cross-validation, epsilon is the minimum absolute improvement in accuracy required to keep searching and smooth is the smoothing value (*a*) for Bayesian parameter estimation.

The next structure is Semi Naive Bayes. The author provides two methods to construct this structure: forward sequential selection and joining and backward sequential elimination and joining.

```
1. fssj('class', dataset = car, k = 5, epsilon = 0.01, smooth = 0)
2. bsej('class', dataset = car, k = 5, epsilon = 0.01, smooth = 0)
```

k-Dependence Bayesian classifier uses k-DB as search algorithm where kdbk is the maximum number of parents per variable:

```
1. kdb ('class', dataset = car, k = 5, kdbk = 2, epsilon = 0.01, smooth = 0)
```

- Parameter learning

The parameter learning algorithms provided by the bnclassify package only handles with discrete variables, so a loss of information is introduced when the data set contains continuous variables. In order to estimate the parameters, the author uses maximum likelihood or Bayesian estimation:

- WANBIA
- AWNB
- MANB

The details of the implementation will not be explained in this work, as this work is more oriented towards dealing with continuous variables. The code which allows us to obtain the parameter is:

```
1. nb <- lp (n, smooth = 1, manb_prior = 0.5)
```

where n is the network obtained in the structure learning. Moreover, a function for learning structure and parameters in a single step is provided:

```
1. nb <- bnc ('tan_cl', 'class', car, smooth=0, dag_args=list(score = 'loglik))
```

- Predicting

The class label can be predicted with function predict( ):

```
1.  predict (nb, car, prob = TRUE)
```

where nb is the model obtained after learning the structure and parameter. The code will return posterior probability when prob equal TRUE. The result is shown below

```
predict(nb,car,prob=TRUE)
          unacc          acc         good        vgood
 [1,] 0.99713851 1.543686e-04 1.058041e-03 1.649081e-03
 [2,] 0.99902297 2.324635e-05 3.111970e-06 9.506731e-04
 [3,] 0.99980259 7.031122e-05 1.250107e-04 2.085808e-06
 [4,] 0.99784883 2.385274e-04 1.681312e-03 2.313289e-04
 [5,] 0.99961087 9.206620e-05 1.106181e-04 1.864421e-04
```

Figure 16. Prediction result of a Naive Bayes structure. Data=car. Prob=TRUE

Each row presents the posterior probabilities of an instance. When prob equal FALSE, the code returns the category of an instance which has the high posteriorly probability:

```
predict(nb,car,prob=FALSE)
 [1] unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc
[13] unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc
[25] unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc
[37] unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc unacc
```

Figure 17. Prediction result of a Naive Bayes structure. Data=car. Prob=FALSE

4.2 Extended bnclassify package

BMN discretize the continuous variables in order to be able to handle them. with a subsequent loss of information. The degree of information loss depending on the discrete methods used, so we can infer the classification results will vary according to the discrete method. A very effective way to avoid the loss of information is to use CGNs. CGN is a network that handles continuous variables assuming all the variables follow a Gaussian distribution. However, we should take into account that the results will not be ideal if the real distribution of the variables is very different from the Gaussian distribution [1].

The original package Bnclassify only handles with the categorical variable. To avoid the loss of information, I have extended the package in order to handle the discrete variables assuming all the variables follow the normal distribution. To make Bayesian networks operable with continuous variables is the main purpose of this work. In order to achieve the goal, I made the following changes：

- Structure learning with continuous attributes
- Gaussian parameters learning
- Estimate the probability

4.2.1 Structure learning with continuous variables

- Naive Bayes, NB*:* Because of the simple structure (the features are independent given class), it does not require any search algorithm to find the optimal network. We can ignore the problem of different variable properties in the NB structure learning process. No changes have been made in this structure.

- Tree-Augment Naive, TAN: The algorithm learns the mutual information between the predictor variables given the class variable and then, construct a score table according to the values calculated in the previous step. The score table is used to construct the TAN structure. The change introduced in this algorithm is when it detects that the dataset contains continuous variables, the function will call the equation 9 proposed by Pérez et al [1] to obtain the score table. The rest of the process is maintained.

- Hill-Climbing TAN (TAN-HC), Hill-Climbing Super-Parent TAN (TAN-HCSP) and k-Dependencies Bayes (KDB): What these algorithms have in common is that they need to decide whether to continue to change the structure by comparing the accuracy of the current structure with the previous one or not. If the accuracy does not improve, the current structure will be returned. Otherwise, keep improving it. Therefore, the calculation of the Gaussian parameters and the prediction that handles continuous variables are needed in each iteration. The details of these two new implementations will be explained at the end of this section.
  The change introduced is when the algorithm detects the data set contains continuous variables, it will proceed to learn the Gaussian parameters of all the nodes taking into account the current structure, and then, it will predict the accuracy.

- Forward sequential selection and joining, FSSJ: This algorithm builds the structure starting with a single variable, class variable, and adds new variables or join the existing variables until there are no improvements. The change introduced is when it detects only one variable in the structure, it will call the parameter learning function that works with the categorical variables since all the class variable must be categorical in this work. Once the first accuracy is calculated, the algorithm adds or joins the variables depending on the accuracy obtained, but here the Gaussian parameter learning function and Gaussian prediction function are used to estimate the accuracy since the new variables are continuous.

- Backward sequential elimination and joining, BSEJ: the algorithm is very similar to FSSJ but in a different direction. It starts with a Naive Bayes structure and then, eliminates or joins the variables depending on the accuracy obtained in the corresponding structure. The change introduced in this algorithm is that it uses the Gaussian parameter learning function and Gaussian prediction function to estimate the accuracy in each iteration. In case it has removed all variables from the structure except the class variable, it will call the parameter learning function that works with the categorical variables.

4.2.2 Parameter learning with continuous variables

In this section, I will introduce a new function integrated into the bnclassify package that allows us to obtain the Gaussian parameters. To make it easier to understand, I'll use an example to illustrate the parameters returned after invoking the new function.

To demonstrate how the program works, suppose we have the data set called 'iris', available in RStudio. The dataset has 150 observations and has a total of 5 variables, 4 of them are numerical (Sepal.Length, Sepal.Width, Petal.Length and Petal.Width) and 1 categorical (Species). The categorical variable has 3 levels, setosa, versicolor, and virginica. After the structure learning, we have the following TAN structure.

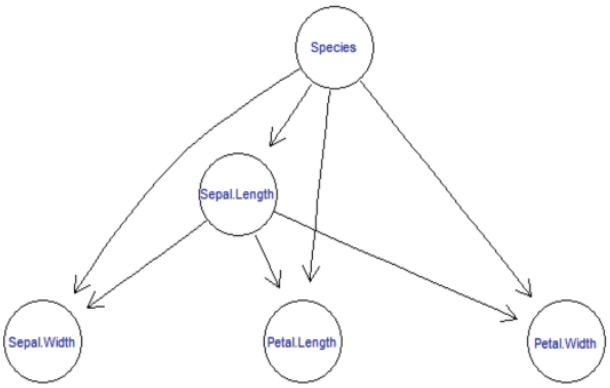

Figure 18. TAN-structure using the iris data package

As we can see in Figure 18, there are different cases, when x_i is a numerical variable, when x_i is a categorical variable, when the parent variable of the node x_i is a categorical variable, and when the parent variables are mixed of numerical and categorical variables. The program (GaussianImplement) designed in this work which allows us to obtain the parameters of a Gaussian distribution covers all the cases mentioned above.

- When x_i is categorical:

The program returns an error message informing that it is not possible to obtain parameters for a category node.

- When x_i is numerical and it has an only categorical parent:

Follow Figure 18, the node that satisfies the condition is Sepal.Length. In this case, the DAG is defined by Sepal.Length ~ Species. Two types of parameters are returned: the expectation and variance vector.

The expectation and variance vector are obtained by calculating the mean and variance value of the variable x_i. The data set used to calculate the coefficient and variance is divided according to the category variable. In this case, the dataset 'iris' will be divided into three subsets since Species has three levels. Thus, the parameters of Gaussian distribution of the node Sepal.Length are obtained in the following way:

Subset	Linear function	Expectation	Variance
Setosa	$--$	$\mu_{Sepal.Length\|c=setosa}$	$\sigma^2_{Sepal.Length\|c=setosa}$
Versicolor	$--$	$\mu_{Sepal.Length\|c=versic.}$	$\sigma^2_{Sepal.Length\|c=versic.}$
Virginica	$--$	$\mu_{Sepal.Length\|c=virginica}$	$\sigma^2_{Sepal.Length\|c=virginica}$

Table 2. Parameter obtained if the node has not numerical parents

The number of parameters obtained from the node Sepal.Length is 6

- When x_i is numerical and its parents are mixed of numerical and categorical:

In this case, the node x_i can be Sepal. Width, Petal.Length or Petal.Width. As an example, the Sepal.Width node has a DAG defined by Sepal.Width ~ Sepal.Length + Species where Sepal.Length and Species are parents of Sepal.Width.

As in the previous case, the Gaussian parameters are obtained using a different subset, but here, the expectation vector of the Sepal.Length node, in addition to the class variable, also depends on another predictor variable, so the expectation vector is obtained using the equation 14 by calculating the coefficient of relationship between the variable x_i and its numerical parents

Subset	Linear function	Expectation	Variance
Setosa	$Sepal.Width \sim \mu_{S.W\|c=setosa} + \beta_{S.W;S.L\|c=setosa} * Sepal.Length$	$\mu_{S.w\|c=setosa}$ $\beta_{S.W;S.L\|c=setosa}$	$\sigma^2_{R(S.w;S.l)\|c=setosa}$
Versicolor	$Sepal.Width \sim \mu_{S.W\|c=versicolor} + \beta_{S.W;S.L\|c=versicolor} * Sepal.Length$	$\mu_{S.w\|c=versicolor}$ $\beta_{S.W;S.L\|c=versicol}$	$\sigma^2_{R(S.w;S.l)\|c=versico}$
virginica	$Sepal.Width \sim \mu_{S.W\|c=virginica} + \beta_{S.W;S.L\|c=virginica} * Sepal.Length$	$\mu_{S.w\|c=virginica}$ $\beta_{S.W;S.L\|c=virginic}$	$\sigma^2_{R(S.w;S.l)\|c=virgini}$

Table 3. Parameter obtained if the node has categorical and numerical parents

The number of parameters obtained from the node Sepal.Width is 9.

The code that allow us to obtain the Gaussian parameters is:

```
1. nb <- tan_cl('Species',iris,score='loglik')
2. str <- lp(nb,iris)
```

Where nb is the TAN structure we obtained after structure learning process. Then, each node of the data set is evaluated by lp() function, if all variable members are categorical, it will invoke the original function that work with categorical variable. Otherwise, it will invoke the new function that calculates the gaussian parameter for each numerical variable, for example:

```
> str$params$Sepal.Width
$coef
                setosa versicolor virginica
(Intercept)  -0.5694327  0.8721460 1.4463054
Sepal.Length  0.7985283  0.3197193 0.2318905

$sd
     setosa versicolor virginica
1 0.2565263  0.2696638 0.2897845
```

Figure 19. Gaussian parameter of the node Sepal.Width

Figure 19 shows the coefficients and standard deviation vector for the node Sepal.Width. Using the equation 14, the expectation vectors is:

$$m_{Sepal.Width|setosa,S.L} = -0.5694327 + 0.7985283 * \mathrm{x}_{Sepal.Length}$$

$$m_{Sepal.Width|versicolor,S.L} = -0.5694327 + 0.7985283 * \mathrm{x}_{Sepal.Length}$$

$$m_{Sepal.Width|virginica,S.L} = -0.5694327 + 0.7985283 * \mathrm{x}_{Sepal.Length}$$

The shorthand function bnc() that learns the structure and parameter in a single step is also modified in this work to adapt the continuous variables.

To extend this example, suppose now the dataset has another categorical column called 'season' and it has four levels: spring, summer, autumn and winter. The structure graph is shown in Figure 20. The way to obtain the expectation vector is the same by using $Sepal.Width \sim \mu_{i|c} + \beta_{i,j|c} * Sepal.Length$, but now, the data set is divided into subsets according to the combinations of Species and season, for example, the subset when Species is setosa and season is spring, the subset when Species is setosa and season is summer and so on.

The number of parameters obtained is 2*12+12.

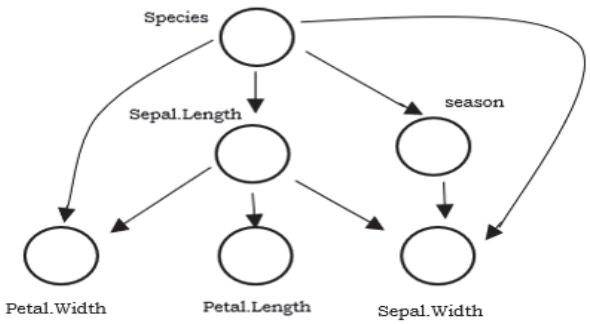

Figure 20. Bayesian network with multiple categorical parents

4.2.3 Predicting with Gaussian Network

Once we have the structure and Gaussian parameter learned, we can estimate the posterior probability, $p(c|\mathrm{x}_i)$, of each category. Where c is the class variable, and x_i are different predictor variables.

The original bnclassify package provides a function called predict() that estimates the probability mentioned above. However, this function only handles the structure and parameters constructed from a categorical data set. Therefore, a function that estimates the probability of continuous variables is designed in this work. The function is called PredictGCN and is integrated within the predict() function.. The function designed returns the posterior probabilities after estimating the likelihood and the prior probability. For example, the posterior probability of an instance to be setosa is:

$$p(c = setosa|S.Length, S.Width, P.Length, P.Width) = p(setosa) * \prod_{i=1}^{d} p\left(\mathrm{x}_i \middle| setosa, pa(\mathrm{x}_i)\right) \quad (18)$$

When $i = 1$,

$$p((S.Length|setosa, pa(\mathrm{x}_i)) = \frac{1}{\sqrt{2\pi}\sigma_{S.L|setosa,pa(x_i)}} exp\left(\frac{-\left(\mathrm{x}_{S.L} - \mu_{S.L|setosa,pa(x_i)}\right)^2}{2\sigma^2_{S.L|setosa,pa(x_i)}}\right)$$

When $i = 2$,

$$p((\mathrm{S.Width}|setosa, pa(\mathrm{x}_i)) = \frac{1}{\sqrt{2\pi}\sigma_{S.W|setosa,pa(x_i)}} exp\left(\frac{-\left(x_{S.W} - \mu_{S.W|setosa,pa(x_i)}\right)^2}{2\sigma^2_{S.W|setosa,pa(x_i)}}\right)$$

And so on.

The code that we need to use in order to obtain the prediction result is similar to the code that works with categorical variables. When the predict function detects the input parameter was calculated from a continuous data set, it will invoke the predictGCNs function. Otherwise, it will invoke the original code. An example of use is shown below:

```
1.  Prediction < -predict (iris, structure)
```

5. Implementation

In this section, I present the results of the experiments carried out in this work. The experiments work with the conditional Gaussian Network classifiers mentioned before and different datasets. The datasets used are:

Column vertebral: each row of the dataset presents information about the shape of the vertebral column of a patient. Each patient can be classified in one out of three categories: Normal, Disk Hernia, and Spondylolisthesis. The dataset contains 310 instances and 7 attributes; one of them is the categorical variable.

Mammography: This dataset is used to predict the stage of a mammographic mass lesion, benign, and malignant. The dataset has 6 attributes, five of them are continuous predictor variables that present the relative data about breast cancer screening. One attribute presents a category class variable. The dataset contains 884 instances: 422 instances are benign (no) stage and 462 instances are malignant (yes).

Breast cancer: It is a dataset that can be used to predict the stage of breast cancer. It contains 31 predictor attributes that describe the features of a fine needle aspirate (FNA of a breast mass) that presents in the image. The dataset has 569 instances and a total of 32 attributes.

Phishing: This dataset collects different features related to legitimate and phishing websites. The dataset contains 1353 instances and 10 attributes. 9 attributes are continuous predictor variables and one categorical variable. The categorical variables present three categories: legitimate: the web is safe, dangerous: the web is not safe, and suspicious: the web can be either phyphy or legitimate

All the dataset is obtained from *UCI repository*. The information of two experiment designed is shown below:

- Experiment 1: how accuracy, time and number of parameters varies using different datasets and CGNs classifiers?

  - Test 1: Accuracy test.
    - Dataset used: vertebral column, breast cancer, mammography, phishing.
    - Algorithm used: NB, TAN-CL, TAN-HC, TAN-HCSP, KDB, FSSJ and BSEJ
    - Process: draw a line chart where y is the accuracy, x is the dataset and list by different algorithms

  - Test 2: Computational time test.
    - Dataset used: vertebral column, breast cancer, mammography, phishing.
    - Algorithm used: NB, TAN-CL, TAN-HC, TAN-HCSP, KDB, FSSJ and BSEJ Process: draw a line chart where y is the time used to compute the structure, x is the dataset and list by different algorithms

  - Test 3: Number of parameters time test.
    - Dataset used: vertebral column, breast cancer, mammography, phishing.
    - Algorithm used: NB, TAN-CL, TAN-HC, TAN-HCSP, KDB, FSSJ and BSEJ

- Process: draw a line chart where y is the total number of parameters in the structure, x is the dataset and list by different algorithms.

❖ Experiment 2: how does the number of parameters vary if I use different methods to process the data set?

- Test 1: comparison of parameter numbers test.

  - Dataset used: mammographic
  - Algorithm used: NB, TAN-CL, TAN-HC, TAN-HCSP, KDB, FSSJ and BSEJ
  - Method used: Gaussian distribution and discretization.
  - Process: draw a bar chart where y is the number of parameters, x is different algorithm used, and group by Gaussian method and discretization method.

5.1 Experiment 1

In this subsection, I analyze the relationship between accuracy-dataset, time-dataset, and parameter-dataset. The efficiency of the classifiers using different datasets is also analyzed in this subsection.

- Test 1: accuracy test

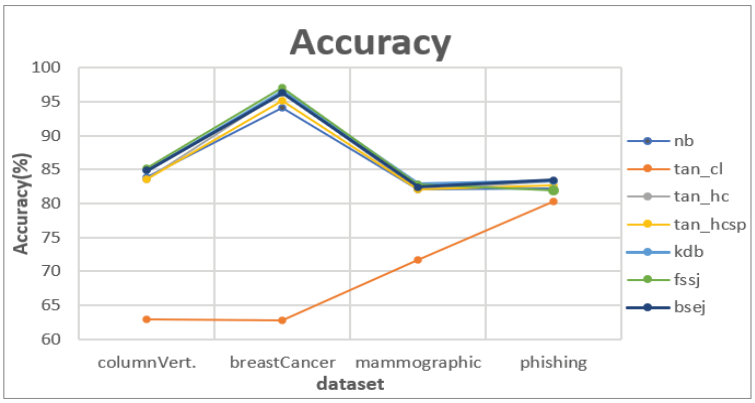

Figure 21. Experiment 1 test 1, accuracy test: estimated accuracy with 2-fold cross-validation, epsilon = 0 and k = 2 for kdb

Figure 21 presents the accuracy obtained using different datasets. The column vertebral data contains a few rows and columns. The breast cancer data presents more features compared to the first data. The mammography data has a greater number of instances than the first one. The last data, phishing, it contains more rows than mammography, therefore, more instances compared to column vertebral.

We can see in the graph:

- The accuracy changed smoothly among the classifiers except for the tree augmented network-ChowLiu.

In order to improve the TAN-CL accuracy, two options are added in the program: AIC and BIC. Both methods penalized the conditional mutual information score.

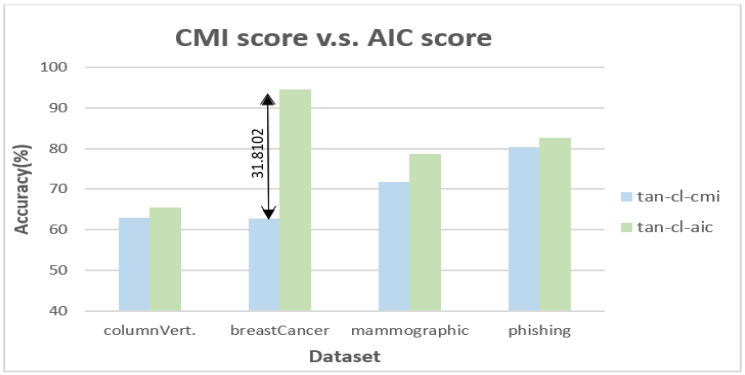

Figure 22. tan-cl penalized by AIC method

As we can see in the figure above, all the accuracy is improved and the best improvement is obtained with the dataset breast cancer, 31.8102%.

- Test 2: computational time test

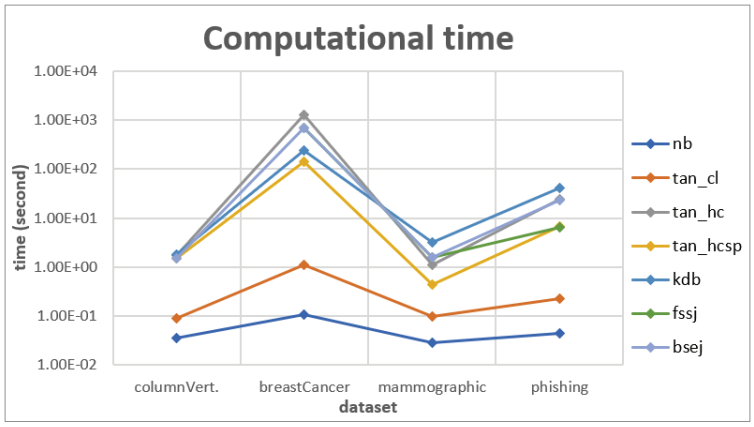

Figure 23. Experiment 1, test2, time test: time used to construct the structure. 2-fold cross-validation, epsilon = 0 and k = 2 for k-DB

As we can see in Figure 23, the classifier that has the least running time is Bayes network. This result makes sense with the theory since the Naive Bayes assume all the predictor variables are conditionally independent, so no searching procedure is required to construct the structure. On the other hand, the classifiers, TAN-HC, TAN-HCSP, k-DB, FSSJ and BSEJ require a lot of time to accomplish the task, which is due to the learning method used. These classifiers search the best variable by estimating the accuracy of all possible structures in the space. For example, suppose we need to construct a TAN-HC classifier with the dataset, breast cancer, the number of the possible structures to be estimated in the first iteration is approximately 300.

In general, the classifier that takes the most time is TAN-HC. As I mentioned in the section 2.3.2, the complexity of this algorithm is O(n^3). Despite the complexity, another Tree-Augment-Naïve classifier called TAN-HCSP is also introduced, and the complexity of this structure is O(n^2). The results obtained after running the program do make sense with the theory, TAN-HCSP uses less computational time than TAN-HC. We can see in Figure 23, TAN-HCSP reached 1120 seconds faster using the data breast cancer.

- Test 3: Number of parameters test

The number of parameters is the sum of the quantity of the coefficient and the deviation parameters of each nodes that exist in the structure. The equation to calculate the parameter number is:

$$parameters = \sum_{i=1}^{k} r * (n_i) + r \qquad (19$$

Where i presents the variables x_i exists in the structure, r is the number of categories exists in the class variable. n_i, is the number of parents of the variable. For example, the number of parameters of Figure 18 is 33.

We can observe two things in Figure 24:

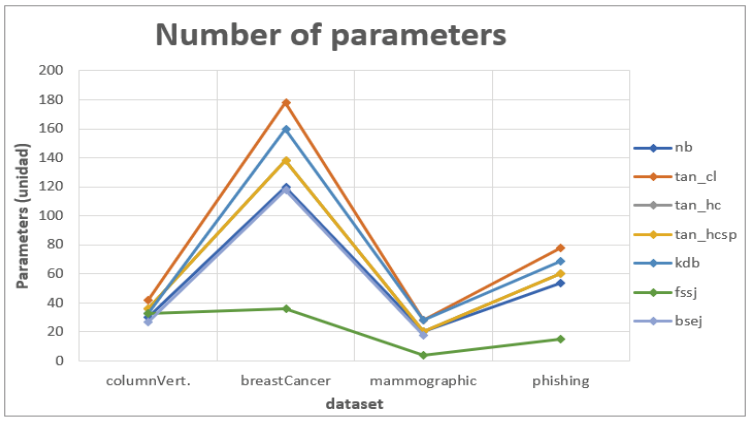

Figure 24.Experiment 1, test3, time test: estimated accuracy with 2-fold cross-validation, epsilon = 0 and k = 2 for kdb

- The number of parameters grows depending on the number of predictor variables in a dataset. A dataset with many attributes constructs a complex Bayesian structure, so a greater number of the parameter is computed for each node in the structure.

- The lowest number of parameters is obtained with FSSJ algorithm. The reason is that it starts with an empty structure and stops the searching process when it considers that there are no improvements, therefore, the number of nodes that form the structure can be lower than other Bayes network classifiers. Naive Bayes classifier also has low parameters. The reason is that the independence

assumption makes all number of the nodes are independent with each other, so the number of parent nodes is equal to 1.

To finish this experiment, we can conclude:

If a data set contains many attributes, more structure options are generated during searching process. It should be noted that if there are more possible structures, the computation time will increase. If there are many nodes in the structure, the number of parameters will increase as well.

5.2 Experiment 2

- Test 1: Comparison of parameter numbers test.

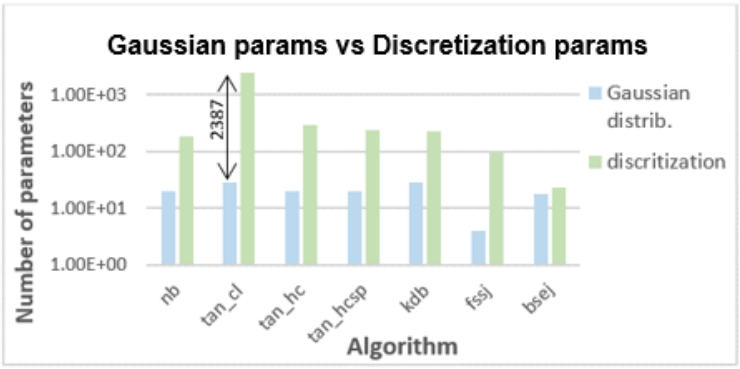

Figure 25. Experiment 1, test3, param comparison test: 2-fold cross-validation, epsilon = 0 and k = 2 for kdb, dataset = mammographic

The blue and green bar presents the parameter numbers obtained using the Gaussian distribution and discretization method. The number of parameters obtained with Gaussian distribution is significantly lower compared to discretization. In the case of TAN_CL, the difference between them is 2387.

6. Conclusions and future work

In this work, I extend the functionality of the bnclassify package allowing the state-of-the-art algorithms can deal continuous predictor variables. I adapted the BMNs classifiers to CGNs assuming all variables follow the Gaussian distribution. The adapted classifiers are: Naive Bayes, Tree-Augmented-Naïve (ChowLiu, Hill-Climbing and Hill-Climbing Super Parent), k-Dependencies Bayesian classifier, forward sequential selection and joining (FSSJ) and backward sequential elimination and joining (BSEJ). The AIC and BIC which penalize the score according to the structure was adapted as well.

In order to adapt the BMNs to CGNs, a new function was created and it is merged into the bnclassify package. This function allows us to obtain the Gaussian parameter of the nodes that exist in the current classifier. Once we have the structure and parameter learned, another new function was designed to estimate the accuracy. This function is also merged into the bnclassify package.

The experiments designed in this work measure the efficiency of the classifiers, analyze the relationship between different parameters, and how the parameter number change when we apply Gaussian implementation. Four data sets are used in the experiments. The conclusions are shown below:

A complex dataset that contains many attributes can get a good accuracy but the cost is the increased computation time since more attributes, more possible structure can be estimated so more time will be used. For example, Tree-Augmented network with Hill-Climbing score looks for all possible structures in the space and selects the one with the highest accuracy. Obviously, it will take a lot of computation time in the early iteration. On the other hand, the classifiers that do not use any kind of searching process, such as Naive Bayes, the impact caused by a complex data set could be ignored, so the computation time is low. A data set with multiple attributes makes the number of parameters larger since the optimal structure can have multiples arcs and a complex relationship between the nodes.

The experiment also showed that the number of parameters obtained using Gaussian distribution is less than the number obtained by discretizing the continuous predictor variables.

The future work line can be:

1. Improve speed and code robustness. As we can see in some classifiers, for example, tree augmented network with Hill-Climbing, it takes a long time to construct the network, so a possible future work is using the existing C++ libraries for graphs and Bayesian networks to refactoring the package and makes it faster

2. Due to the limited time of this work, the function that I have implemented only works for a dataset where the predictor variables are continuous and it will not work if the predictor variables are a mixture of continuous and categorical. However, the function that learns Gaussian parameters is able to work with mixed datasets. Therefore, the work remains only in designing a learning structure from a dataset which the attributes are mixture of continuous and categorical variables

7. Annex

7.1 Data tables

In this subsection, you can find the data obtained during the experiment

Method	Gaussian distribution		
Dataset	Mammographic		
Parameters	2-fold, epsilon = 0, score = 'loglik', kdbk = 2		
Algorithm	Accuracy (%)	Time (second)	Number of parameters (unit)
Naïve Bayes	82.02654	0.028410	20
Tan-ChowLiu	71.65259	0.099206	28
Tan-Hill Climbing	82.02654	1.104673	20
Tan-CL Super parent	82.02654	0.488030	20
k-dependency Bayes	82.87000	3.185832	28
Fssj	82.75000	0.197170	4
Bsej	82.38442	1.560817	18

Table 4. Data. Method: Gaussian distribution, dataset: mammographic

Method	Gaussian distribution		
Dataset	Breast cancer		
Parameters	2-fold, epsilon = 0, score = 'loglik', kdbk = 2		
Algorithm	Accuracy (%)	Time (second)	Number of parameters (unit)
Naïve Bayes	94.02460	0.106743	120
Tan-ChowLiu	62.74165	1.098292	178
Tan-Hill Climbing	96.48506	1260.000	160
Tan-CL Super parent	95.07909	140.4000	138
k-dependency Bayes	96.48506	240.4300	160
Fssj	97.01230	26.74600	38
Bsej	96.30931	684.0000	118

Table 5. Data. Method: Gaussian distribution, dataset: Breast cancer

Method	Gaussian distribution		
Dataset	Phishing		
Parameters	2-fold, epsilon = 0, score = 'loglik', kdbk = 2		
Algorithm	Accuracy (%)	Time (second)	Number of parameters (unit)
Naïve Bayes	82.26164	0.043884	120
Tan-ChowLiu	80.33999	0.226593	178
Tan-Hill Climbing	83.29638	23.92000	160
Tan-CL Super parent	82.70510	6.682658	138
k-dependency Bayes	83.29638	41.246430	160
Fssj	81.89209	6.487310	38
Bsej	83.44420	23.287450	118

Table 6. Data. Method: Gaussian distribution, dataset: Phishing

Method	Gaussian distribution		
Dataset	Column vertebral		
Parameters	2-fold, epsilon = 0, score = 'loglik', kdbk = 2		
Algorithm	Accuracy (%)	Time (second)	Number of parameters (unit)
Naïve Bayes	83.87097	0.035463	30
Tan-ChowLiu	62.90323	0.090539	42
Tan-Hill Climbing	83.54839	1.6845100	36
Tan-CL Super parent	83.54839	1.5243880	36
k-dependency Bayes	84.83871	1.8005670	33
Fssj	85.16129	1.2003910	33
Bsej	84.83871	1.4923510	27

Table 7. Data. Method: Gaussian distribution, dataset: column vertebral

Method	Discretization		
Dataset	Mammographic		
Parameters	2-fold, epsilon = 0, score = 'loglik', kdbk = 2		
Algorithm	Accuracy (%)	Time (second)	Number of parameters (unit)
Naïve Bayes	---	---	185
Tan-ChowLiu	---	---	2415
Tan-Hill Climbing	---	---	295
Tan-CL Super parent	---	---	241
k-dependency Bayes	---	---	225
Fssj	---	---	95
Bsej	---	---	23

Table 8. Data. Method: Discretization, dataset: mammographic

8. Referencies

[1] A. Pérez, I. Inza and P. Larraña, "Supervised classification with conditional Gaussian networksL Increasing the structure complexity from naive Bayes," *Science Direct,* 13 02 2006.

[2] M. Bojan, L. Pedro y B. Concha, «bnclassify: Learning Bayesian Network Classifiers,» *The R journal,* vol. 10/2, 2018.

[3] W. Song, "Aliyun," alibaba, 14 03 2018. [Online]. Available: https://developer.aliyun.com/article/537884. [Accessed 15 06 2020].

[4] icaoys, "One example to clarify (a priori/posteriori distribution/likelihood estimation)," CSDN, 17 10 2017. [Online]. Available: https://blog.csdn.net/qq_23947237/article/details/78265026. [Accessed 22 05 2020].

[5] shipipi, "Bayesian network and related knowledge," CSDN, 11 07 28018. [Online]. Available: https://blog.csdn.net/Pancheng1/article/details/81001459. [Accessed 16 07 2020].

[6] M.Minsky, «Steps toward Artificial Intelligence,» *Transaction on Intuitive of Radios Engineers,* vol. 49, pp. 8-30, 1961.

[7] LYHao, "jianshu," 09 12 2018. [Online]. Available: https://www.jianshu.com/p/a4ddf754357b. [Accessed 12 05 2020].

[8] . N. Friedma, D. Geiger y M. Goldszmidt, «Bayesian network classifiers,» *Mach Learn,* vol. 29, nº 2, pp. 131-163, 1997.

[9] Padmanaban and Harini, "Comparative Analysis of Naive Bayes and Tree Augmented Naive Bayes Models," 2014. [Online]. Available: https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1350&context=etd_projects. [Accessed 20 05 2020].

[10] E. Keogh y M. Pazzaini, «Learning the structure of augmented Bayesian classifiers,» *International Journal on Artificial Intelligence Tools,* pp. 455,456,459, 2002.

[11] J. Leogh y J. P. Eaon, «Algorithms for Learning Augmented Bayessian Classifiers,» *International Journal on Artificial Intelligence Tools,* pp. 587-601, 2002.

[12] M.Sahami, «Learning limited dependence Bayesian classifiers,» *In proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining,* vol. 96, pp. 335-338, 1996.

[13] B. Rosa, M. Marisa, Q. Jorge y L. Pedro, «Feature selection in bayesian classifiers for the prognosis of survival o cirrhotic patients treated with TIPS,» *science direct,* p. 376'388, 2005.

[14] Wang, «k-Dependence Bayesian Classifier Ensemble,» *entropy,* 2017.

[15] M.J.Pazzani, «Searching for Dependencies in Bayesian Classifiers,» de *Artificial Intelligence and Statistics IV,* New York, Springer-Verlang, 1997.

[16] M.DeGroot, Optimal Statistical Decisions, New York: McGraw-Hill, 1970.

[17] J. Castillo, Expert Systems and Probabilistic Network Models, Springer-Velag, 1997.

[18] D.D.Geiger, «Learning Gaussian networks,» *Technical report, Microsoft Research, Advanced Technology Division,* 1994.

[19] S.G.Bottcher, «Learning Gaussian networks,» *Aalborg Universitet,* 2005.

[20] SFESLY, "Bayesian learning," CSDN, 08 07 2015. [Online]. Available: https://www.cnblogs.com/hustxujinkang/p/4629175.html.. [Accessed 24 05 2020].

[21] V. Liu, "Bayesian parameter learning," zhihu, 03 08 2018. [Online]. Available: https://zhuanlan.zhihu.com/p/41235124. [Accessed 21 05 2020].