Trabajo Fin de Máster

A new multivariate data analysis model: constrained Naïve Bayes

Presented by:
María Remedios Sillero Denamiel

Supervisors:
Dr. Rafael Blanquero Bravo, Universidad de Sevilla
Dr. Emilio Carrizosa Priego, Universidad de Sevilla
External Supervisor:
Dr. Pepa Ramírez Cobo, Universidad de Cádiz

June 24, 2016

Agradecimientos

En primer lugar quiero agradecer a los doctores Emilio Carrizosa Priego y Rafael Blanquero Bravo quienes, hace poco más de un año, me brindaron la oportunidad de conocer qué es la investigación matemática, un trabajo que requiere mucha dedicación y constancia, las cuales me han sabido transmitir a la perfección.

El trabajo que aquí se presenta me ha permitido conocer a la doctora Pepa Ramírez Cobo, quien además de volcarse con éste, ha sido un pilar fundamental para mí en este tiempo.

Como alguien diría: "Las personas con las que trabajo son, sin duda, lo mejor de este trabajo".

A mis padres, Andrés y Aurelia, y a mi hermana Rocío, ellos son mi apoyo incondicional y los que me han enseñado a luchar por lo que quiero. Gracias Antonio J. Molero por animarme siempre y por estar siempre ahí.

Gracias a todos por confiar en mí.

Contents

Introduction

The classification problems are in vogue due to the fact that they are necessary in many filds of real life, where the datasets gather so much information. Some examples that reveal the importance of these problems are: the early detection of diseases, the granting of credit to a certain individual,... It seems clear that to work with good features, that is, attributes that are able to distinguish the different classes, is a key point.

Over the years, the classification problems have been studied and many classifiers have been developed. Some examples of them are: Support Vector Machine ([5]), Random Forest ([3]), Naïve Bayes ([6],[11],[8],[10]), between others. Once a classifier is defined, the question that everyone wants to know is how good is a given classification function. In order to solve this question, measures of effectiveness were defined ([16]).

This work focuses on the Naïve Bayes classifier. A thorough study of the behavior of the Naïve Bayes classifier, the effect of assuming independence and the influence of the involved parameters estimation methods is presented. Finally, as a major contribution of this work, a different version of Naïve Bayes classifier is presented, in which the estimation is made by imposing constraints on the effectiveness measures on the obtained classifier.

The structure of the work is described next. In Chapter 1 an introduction to classification is made. In Chapter 2, Naïve Bayes classification is explained in more detail, in addition to introduce a coefficient for the calculation of both linear and nonlinear relationships between variables ([15], [18]). Finally, the different parameters estimation methodologies will be compared. In Chapter 3 a study of the novel approach will be carried out.

Chapter 1

Motivation and Background

1.1 Definition of a classifier and performance measures

A classifier is a function, called f, that associates input variable vectors $\mathbf{x} = (x_1, \ldots, x_p) \in \mathcal{X}$ to output classes $y \in \{C_1, \ldots, C_K\}$, where $\mathcal{X}$ is the space of variables and $x_i \in \mathbb{R}$ although they will be continuous and discrete. Moreover, it is considered that classes are categorical and respectively exclusive. The aim of statistical classification is to learn f from a labeled training dataset of N input-output pairs, $(\mathbf{x}_n, y_n)$, $n = 1 \ldots N$, where $\mathbf{x}_n$ are the individuals and y_n denotes the class for such individual.

Once the classifier has been obtained using the training set, its performance is measured ([16]) by using a test dataset so as to avoid the over-fitting. Note that both samples, training and test, can be assumed to be generated from the same population.

Let the true class for a given individual be denoted x and the class for the same individual indicated by the classifier be denoted y. Although it is expected that they are the same for as many individuals as possible in the test dataset, nevertheless, the classifier will make errors. The probabilities that the classifier will return correct results on each of the classes are estimated in a frequentist manner by dividing the number of rightly classified individuals for a given class by the total number of individuals in that class in the test dataset.

The probability $P(y = C_k | x = C_k)$ is understood as the probability that the classifier designates class C_k, given that the individual belongs to class C_k. Now, $P(y = C_k | x = C_l)$, $k, l \in \{1, \ldots, K\}$, $k \neq l$, shows the probability that the classifier fails in classifying an individual actually from class C_l as an individual in class C_k.

The correct classification rate (CCR) and the misclassification rate (MCR) can be obtained from $P(y = C_i|x = C_j)\ \forall i, j \in \{1, \ldots, K\}$. The CCR of a certain class is defined as follows:

$$CCR_{C_k} = \frac{\#\{Correct\ classifications\ of\ C_k\ class\}}{\#\{Individuals\ from\ C_k\}},\ k = 1, \ldots, K. \tag{1.1}$$

And the MCR of a certain class is:

$$MCR_{C_k} = 1 - CCR_{C_k},\ k = 1, \ldots, K.$$

1.2 Overall view of classification methods

As commented in the previous section, the aim of classification methods is to specify a class for each input individual. Given a training dataset of the form $(\mathbf{x}_n, y_n)$, where $\mathbf{x}_n \in \mathcal{X}$ is the n-th individual and $y_n \in \{C_1, \ldots, C_K\}$ is the n-th class the aim is obtaining a model (classifier) f that computes $f(\mathbf{x})$ for a new individual. In this section two kinds of classification approaches are distinguised: binary classification method and multiclass classification method.

Depending on the number of class labels it will be said that we are dealing with a binary classification, when the number of class labels is two, or a multiclass classification, in case more than two classes exist. However, the multiclass classification problem can be viewed as a set of binary classification problems, which can be solved via binary classifiers as is described below.

One-versus-all

This approach consists of reducing the problem of classifying K classes into K binary classification problems, where each problem distinguishes a fixed class from the other $K-1$ classes. For this strategy, K binary classifiers are needed, where the k-th classifier is trained under positive individuals from class k and negative individuals belonging to the other $K-1$ classes. When an unknown individual is classified, the most voted class is considered the winner, and this class is assigned to such individual.

One-versus-one

In this approach, each class is contrasted against each other. A binary classifier is constructed to separate each pair of classes, while ignoring the remaining classes. This strategy demands building $\frac{K(K-1)}{2}$ binary classifiers. When it classifies a new individual, a voting procedure is performed between the classifiers and the class with the

maximum number of votes is the winner.

1.3 Benchmark classifiers

A number of classifiers have been proposed in the literature. Among them, we should highlight the Support Vector Machine, the Random Forest and the Naïve Bayes classifier because of their wide range of applications, tractability and computational reasons.

1.3.1 Support Vector Machine

Support Vector Machine (SVM) was introduced by Vapnik and co-workers ([5]). This subsection will briefly review the method. The classic linear SVM behaves as follows. Given a new observation $\mathbf{x}$, the classifier is defined as

$$f(\mathbf{x}) = \beta^t \mathbf{x} + \beta_0,$$

where $\beta \in \mathbb{R}^p$, $\beta_0 \in \mathbb{R}$ and such that if $f(\mathbf{x}) > 0$, the observation is assigned to class $+1$. In other case, the class -1 is selected. But the aim is to find a classifier (or separation function) that classifies correctly between individuals of each class and that it fails as little as possible in a certain given dataset. In other words, it is sought that all positive points ($y = 1$) in the training dataset are classified to class 1 and negative points ($y = -1$) are assigned to class -1. This can be formulated as

$$y(\beta^t \mathbf{x} + \beta_0) > 0,$$

for all individual from training dataset. In the case in which $f(\mathbf{x}) = 0$, it is necessary to fix a rule, such as classifying at random.

Depending on the dataset, a linear separation function f will be able to separate both classes, but, in other cases, the dataset will not be linearly separable. Let us study both options.

Linearly Separable Case

This case is characterized because the positive points ($y = 1$) and negative points ($y = -1$) from the training dataset can be separated by a hyperplane:

$$\{\mathbf{x} : f(\mathbf{x}) = \beta \mathbf{x} + \beta_0 = 0\}$$

where β is the weight vector and β_0 is the bias. So, if this hyperplane is able to separate the positive and the negative individuals, then, the positive and the negative hyperplanes are, respectively the following:

$$H^+ : \beta^t \mathbf{x} + \beta_0 > 0, \text{if } y = 1$$
$$H^- : \beta^t \mathbf{x} + \beta_0 < 0, \text{if } y = -1.$$

Let us notice that, for separable datasets, lots of separating hyplerplanes can be built. Fixed one of them, it is defined d^+ as the shortest distance from the separating hyperplane to the nearest positive data individual, and d^- as the shortest distance from the separating hyperplane to the nearest negative data individual. Moreover, the hyperplane that maximizes the distance between it and the closest observation is called the optimal separating hyperplane.

The optimization problem that returns the optimal separating hyperplane, using the euclidean distance, is the following one:

$$\min_{\beta,\beta_0} \beta^t \beta$$
$$s.t. : y_n(\beta_t \mathbf{x}_n + \beta_0) \geq 1, \forall n = 1, \ldots, N.$$
$$\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}.$$

where N is the cardinal of the training dataset.

Nonlinearly Separable Case

The previous case is not common in real applications, that is, it will be completely likely to find the classes to superimpose. It seems clear that this kind of datasets are more difficult to classifier with linearly hyperplanes because of it will be problems for classification, especially in overlapping individuals.

If it is considered ε as a perturbation, it is possible to include a sum in the objective funtion to control this perturbation, that is:

$$\min_{\beta,\beta_0} \beta^t \beta + C(\|\varepsilon\|_p)^p$$
$$s.t. : y_n(\beta_t \mathbf{x}_n + \beta_0) + \varepsilon_n \geq 1, \forall n = 1, \ldots, N.$$
$$\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}.$$
$$\varepsilon_n \geq 0, \forall n = 1, \ldots, N.$$

where N is the cardinal of the training dataset and $C > 0$ is a regularization parameter.

Finally, for the case when none of the above cases can be used, they can be combined with a technique that performs, using 'kernels', a non-linear mapping to the feature space. That is, SVM finds a non-linear hyperplane in feature space which corresponds with a non-linear decision in the training dataset (see [7]).

1.3.2 Random Forest

Random Forest (RF) was introduced by Breiman in 2001 and consists of a combination of tree classifiers. The goal of classification trees ([17]) is to create a model that predicts the value of a response variable based on several predictor variables arranged as a tree. Each interior node of tree classifiers corresponds to one of the predictor variables and, moreover, in some cases there are other edges towards daughter nodes depending on the value of that predictor variable.
Each sheet represents a value (class) of the response variable given the values of the predictor variables represented by the pathway from the root node to the sheet.

Each tree learns by splitting the full set of attributes into subsets of features. This process is repeated on each subset in a recursive manner, and the recursion finishes when the subset at a node has all the same value of the response variable, or when splitting does not add value to the predictions.

Improvements in classification results can be obtained from generating groups of trees and making that they select the most popular class. In order to create these groups of trees, random vectors of features are used to control the growth of each tree in the group. That is, let us consider the k-th tree, then the random vector Θ_k for this tree is sampled independently but with the same distribution than $\Theta_1, \ldots, \Theta_{k-1}$. The training dataset and Θ_k are necessary to construct the corresponding tree, and this yields a classifier $f(\mathbf{x}, \Theta_k)$, with $\mathbf{x}$ the input vector. In summary, the procedure that RF follows is: firstly a big number of trees is constructed and, later, it is selected the most voted class.

So, let us define a RF as in [3]:

Definition 1.3.1 *A random forest is a classifier consisting of a collection of tree-structured classifiers* $\{f(\mathbf{x}, \Theta_k), k = 1, \ldots\}$ *where the* Θ_k *are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input* $\mathbf{x}$.

In more detail, fixed B, a large number, the RF algorithm can be written as:

1. For $b = 1$ to B:

   - Generate a bootstrap sample $\mathbf{Z}^*$ of size n from the training dataset, where n is the size of the training dataset (a sample obtained by sampling n times with replacement from the training dataset).
   - Construct a random-forest tree T_b to the previous data. This will be done by recursively repeating the following steps for each terminal node of the tree:
     - Select m variables at random from the p variables.
     - Pick the best variable among the m.
     - Split the node into two daughter nodes.
   - Output the ensemble of trees $\{T_b\}_1^B$.

   This will be repeated until the minimum node size n_{min} is reached.

2. And finally, to make a prediction at a new individual $\mathbf{x}$, if $\hat{C}_b(\mathbf{x})$ is the class prediction of the b-th random-forest tree, then

$$\hat{C}_{rf}^B(\mathbf{x}) = \text{majority vote}\{\hat{C}_b(\mathbf{x})\}_1^B.$$

1.3.3 Naïve Bayes

Although in the next chapters this classifier will be explained in more detail, a brief description of Naïve Bayes classifier is made in this first chapter to introduce it.

The Naïve Bayes approach, ([6],[11],[8],[10]) is a classification technique that is mainly proper when the dimension p of the feature space is large, making joint density estimation disagreeable.

The Naïve Bayes classification is based on Bayes' rule and works as follows. Name the vector of measurements of a individual by $\mathbf{x} = (x_1, \ldots, x_p)$ and its class by y ($y \in \{C_1, \ldots, C_K\}$). Then, to obtain estimates of $p(\mathbf{x}|y)$, the distribution of $\mathbf{x}$ for individuals from class y, and estimates of $\pi(y)$, $y \in \{C_1, \ldots, C_K\}$, the probabilities that a member of class y happens, and then, it merges them using Bayes theorem to give estimates of $p(y|\mathbf{x}) \propto p(\mathbf{x}|y)\pi(y)$.

Now, it is necessary to estimate the multivariate class-conditional distributions of $\mathbf{x}$. The Naïve Bayes Classifier presupposes that $p(\mathbf{x}|y)$ factorizes into a product of its univariate marginals, and it is selected the class y that maximizes this value over all the classes.

Chapter 2

Naïve Bayes classifier

In this chapter, the Naïve Bayes classifier (NB) is described. Since the tractability of this algorithm is due to the hypothesis of conditional independence of the features to the class, a discussion about such hypothesis will be given. On the other hand, in order to implement the NB, a statistical approach for estimating the distributions of the features needs to be chosen. Here, two different methodologies will be described and compared on a real dataset: the common, classic, frequentist approach, versus a Bayesian approach based on the Normal Gamma distribution.

2.1 Description

Assume that our classification setting is given by a set of p features $X_1, \dots, X_p$ and K possible classes, $C_1, \dots, C_K$. Given a new observation $\mathbf{x} = (x_1, \dots, x_p)$ the aim is to assign to $\mathbf{x}$ one of the K classes. The NB performs by computing the conditional probabilities $p(C_k \mid \mathbf{x})$ for $k = 1, \dots, K$ and the class $\hat{y} \in \{1, \dots, K\}$ assigned to $\mathbf{x}$ is that maximizes $p(C_k \mid \mathbf{x})$, that is

$$\hat{y} = \underset{k \in \{1,\dots,K\}}{\operatorname{argmax}} \; p(C_k \mid \mathbf{x}). \tag{2.1}$$

The computation of $p(C_k \mid \mathbf{x})$ may be cumbersome if the number of features p is large. However, the use of the Bayes theorem makes the model more tractable since

$$p(C_k \mid \mathbf{x}) = \frac{\pi(C_k) p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}, \tag{2.2}$$

where $\pi(C_k)$ is the prior distribution for the class, $p(\mathbf{x} \mid C_k)$ is the likelihood function of the data and $p(\mathbf{x})$ is the so-called evidence. Since the evidence is constant (it is

independent on the class), in practice, the interest is in computing the numerator, which can be written as the joint distribution

$$p(\mathbf{x}, C_k) = p(x_1 \mid x_2 \dots x_n, C_k)p(x_2 \mid x_3 \dots x_n, C_k) \dots p(x_n \mid C_k)\pi(C_k). \tag{2.3}$$

The key assumption of the NB, which makes it a so tractable classifier even for large values of p, is the independence of the features conditioned to the class, which implies that (2.3) can be simplified to

$$p(\mathbf{x}, C_k) = \pi(C_k) \prod_{i=1}^{p} p(x_i \mid C_k).$$

Finally, the probabilities of interest (2.2) will be computed according to

$$p(C_k \mid \mathbf{x}) \propto \pi(C_k) \prod_{i=1}^{p} p(x_i \mid C_k).$$

Note that in order to implement the NB classifier, a prior distribution $\pi(\cdot)$ for the class as well as a probability distribution for the features conditioned to the class $X_i \mid C_k$ need to be selected. Common choices for the last include the Normal and Multinomial distributions, for the continuous and discrete cases, respectively. Concerning the choice of the prior, it can be based on the researcher's previous knowledge of the problem or, in the case of lack of prior knowledge, it can be set equal for all classes (that is, $\pi(C_k) = 1/K$, for all $k = 1, \dots, K$).

2.2 Discussion about the independence hypothesis

As previously commented, the independence assumption in the NB notably simplifies the computation of the conditional probabilities as in (2.2). However, this assumption is generally not true in practice and violated for many real datasets. The aim of this section is to discuss how this affects the performance of the classifier. As a toy example, consider three features X_1, X_2, X_3 such that X_1 and X_2 are independent, but X_3 is highly correlated with X_1. The assumption of independence leads to decomposing the joint density as the product of the three marginals, and therefore the influence of X_1 ($\simeq X_3$) is different than it should be. Hence, it is natural to think that such miscalculations may possibly deteriorate the classifier's performance. However, the NB has proven to be comparable or superior to many well-known classification alternatives. In [6] this fact is explained as follows:

> "*...although the individual class density estimates may be biased, this bias might not hurt the posterior probabilities as much, especially near the decision regions. In fact, the problem may be able to withstand considerable bias for the savings in variance such a naive assumption earns...*"

Also, [8] provides another possible explanation about the over performance of the NB, which is based on the features' selection process. Due to the measurement of variables is often quite expensive (for example, in biomedical contexts as metabolomics), a previous selection step is undertaken in order to eliminate redundant information. The result of such selection is a set of variables which tend to be weakly correlated.

Next, we explore more in depth the effect of dependent features on the NB performance via a numerical example. Assume that the number of classes is only two (positive and negative), and four features ($M1$, $M2$, $M3$ and $M4$) are simulated according to a multivariate Normal distribution with means vectors given by

$$\mu_+ = (2, 3, 4, 1), \quad \mu_- = (3, 6.5, 2, 3.5)$$

and covariance matrix defined by

$$\begin{array}{c} \\ M1 \\ M2 \\ M3 \\ M4 \end{array} \begin{array}{c} \begin{array}{cccc} M1 & M2 & M3 & M4 \end{array} \\ \begin{pmatrix} 1.0 & 0.9 & 0 & 0 \\ 0.9 & 1.0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{pmatrix} \end{array}$$

Note that features are linearly independent, except for $M1$ and $M2$ which are highly correlated. In order to train the NB classifier (where Normal conditional marginals are assumed), a sample of size equal to 1000 is used. For validation of the method, a sample of 300 observations is considered. In both cases, the proportion of the classes is equal. To implement the NB, the routine **NaiveBayes** from the library **klaR** is used (see the Appendix for the details). Table 2.1 shows the performance values (correct classification rates) for both classes, for all possible combinations of features.

From the table, it can be observed how the results under the selection ($M2, M3, M4$) (that is, when one dependent variable is removed) are better than using the complete information provided by the four variables. This fact points towards the importance of a proper feature selection method that discards noisy information. Because of the independence assumption in the NB, such attributes' selection process should produce a final set of independent features with high significance. Traditionally, the Pearson correlation coefficient has been used to measure the (linear) dependence between random variables. However, other type of correlations (nonlinear) may be present in the data and are ignored by the Pearson coefficient. The next section introduces an index, on which such feature selection process might be based, that takes into account nonlinear associations.

2.2.1 Maximal Information Coefficient (MIC)

The Maximal Information Coefficient (MIC) (see [15],[18]) measures nonlinear relationships among random variables. Its definition arises from that of the concept of *mutual*

Combination	CCR Positive Class	CCR Negative Class
M1	0.741	0.708
M2	0.954	0.96
M3	0.838	0.829
M4	0.888	0.899
M1, M2	0.935	0.952
M1, M3	0.865	0.896
M1, M4	0.905	0.895
M2, M3	0.968	0.993
M2, M4	0.973	0.967
M3, M4	0.935	0.959
M1, M2, M3	0.942	0.979
M1, M2, M4	0.966	0.948
M1, M3, M4	0.953	0.947
M2, M3, M4	**0.987**	**0.993**
M1, M2, M3, M4	0.974	0.98

Table 2.1: Results of each combination of variables.

information (MI) (see, [9]), which quantifies the information about one variable X that is provided by a different variable Y. The MI coefficient performs by exploring if the products of marginal distribution $p(X)p(Y)$ is similar to the joint distribution $p(X, Y)$. Specifically, the MI coefficient between two discrete variables X and Y is defined as

$$MI(X,Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right),$$

where $p(x)$ and $p(y)$ denote the probability mass functions of X and Y, and $p(x, y)$ is the joint probability mass function of X and Y.

Similarly, for the continuous case, the coefficient is given by

$$MI(X,Y) = \int_Y \int_X p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right) dxdy,$$

where now $p(\cdot)$ refers to a density function.

The MI coefficient presents two main problems. The first is from a computational viewpoint, since its calculation depends on a two dimensional smoothing. The second problem is that it is not bounded, which makes its interpretation difficult. After more than 50 years of development of the MI index, the MIC coefficient has been proposed to overcome the drawbacks of the MI. The MIC captures the relationships between two variables using a grid on the scatterplot of these two variables. To calculate the

MIC, all possible grids are explored up to a maximal grid resolution (depending on the sample size), and for each pair of integer values (x, y), the highest possible MI realizable by any x-by-y grid is computed. Then, such MI values are normalized in the interval $[0, 1]$ in order to guarantee the comparison between all grids. To calculate the MIC coefficient, first the matrix $M = (m_{x,y})$ is defined where $m_{x,y}$ is the largest normalized mutual information obtained by any x-by-y grid, so that the maximum value in M will be the value of the MIC coefficient. Formally, let G a grid and let I_G be the mutual information of the probability distribution induced on the boxes of G, with the probability of a box proportional to the number of points dropping inside the box. The (x, y)-th entry of M is denoted by $m_{x,y}$, defined as

$$m_{x,y} = \max I_G / \log \min\{x, y\},$$

where this maximum is calculated over all x-by-y grids G. The MIC index is defined as the maximum of $m_{x,y}$ between all ordered pairs (x, y) satisfying $xy < B$, where B is dependent on the sample size. Note that the elements of M take values between 0 and 1, and consequently, the MIC does, too. Also, the MIC index satisfies the symmetry property, that is, $MIC(X, Y) = MIC(Y, X)$, because of the symmetry of the MI coefficient and because of I_G depends on the rank order of the data. Finally, it should be noted that to compute M, it is necessary to optimize over all grids, so it seems clear that computing tools will be need to calculate the MIC value. In particular, the software R computes the MIC coefficient through the command **mine**, included in the library **miverva** (see the Appendix for the details).

2.3 Parameters' estimation methodologies

As described in Section 2.1., in order to implement the NB classifier, a probability distribution for the features conditioned to the class needs to be selected,

$$X_i \mid C_k \sim F_{\theta_{i,k}}(x).$$

Once such model is set, then it needs to be estimated through sample data. Throughout this work, it will be assumed that $X_i \mid C_k$ follows a Normal distribution with unknown mean and variance denoted by $\theta_{i,k} = \left(\mu_{i,k}, \sigma^2_{i,k}\right)$. In next section we explore two possible alternatives for estimating the model's parameters.

2.3.1 The frequentist versus the Bayesian paradigms

Two main approaches can be considered for statistical inference: the frequentist (or classic) and the Bayesian one. The first approach undertakes estimation just by taking into account the observed sample data and, basing on the concept of frequency, it

derives point estimates for the model parameters by maximizing the likelihood function. Therefore, under a Gaussian likelihood, the frequentist approach gives as parameters' points estimates the sample mean and the sample variance.

The second approach considers instead the possible prior knowledge that the researcher possesses about the problem, and this previous knowledge is updated (via the Bayes formula) by the likelihood function once data are observed. The use of the prior knowledge implies assigning a probability distribution (the so-called *prior distribution*) to the model parameters, which are treated as random variables (instead of unique, unknown values as in the frequentist approach). Such prior distribution will transform (using the likelihood function) into the *posterior distribution*, which is the object of interest for Bayesian statisticians. Formally, the Bayesian inference approach performs as follows. Suppose that before the experiment is undertaken, our prior distribution describing the model parameter θ is $\pi(\theta)$. The data are coming from an assumed model (likelihood) which depends on the parameter and is denoted by $f(x|\theta)$. Bayes theorem updates the prior $\pi(\theta)$ to the posterior by accounting for the data x,

$$\pi(\theta|x) = \frac{h(x,\theta)}{m(x)} = \frac{f(x|\theta)\pi(\theta)}{m(x)},$$

where $m(x)$ is a normalizing constant, $m(x) = \int_\Theta f(x|\theta)\pi(\theta)d\theta$. Some of the advantages of the Bayesian statistics are pointed out next:

- The uncertainty is expressed via the probability distribution. The statistical inference follows a conceptually simple recipe embodied in Bayes' theorem.
- Available prior information is coherently incorporated intro the statistical model describing the data.
- The FDA (US Food and Drug administration) recommends the use of a Bayesian methodology in the design and analysis of clinical trials for medical devices. Some of the reasons are:
  - Valuable prior information is often available.
  - The use of prior information may alleviate the need for a larger sized trial.
  - Bayesian methods allow for great flexibility in dealing with missing data.
  - Bayesian models facilitate meta-analysis.

For a detailed description of the Bayesian paradigm, we refer the reader to [19].

In this work, because of the versatility that priori distributions provide, we will assume a Bayesian framework. The next section introduces the Normal-Gamma model, as a useful tool for Bayesian inference of the Normal distribution.

2.3.2 The Normal-Gamma model

Consider a pair of random variables (X, Y) such that the conditional distribution $X \mid Y$ is normal

$$X \mid Y \sim N\left(\mu, \frac{1}{kY}\right).$$

Also, Y is distributed according to a Gamma probability model,

$$Y \mid (\alpha, \beta) \sim \text{Ga}(\alpha, \beta).$$

Then, it is said that the pair (X, Y) follows a Normal-Gamma distribution, noted by

$$(X, Y) \sim NG(\mu, k, \alpha, \beta).$$

The joint probability density function can be found as

$$f(x, y \mid \mu, k, \alpha, \beta) = \frac{\beta^\alpha \sqrt{k}}{\Gamma(\alpha)\sqrt{2\pi}} y^{\alpha - \frac{1}{2}} e^{-\beta y - \frac{ky(x-\mu)^2}{2}}. \tag{2.4}$$

Figure 2.1 depicts the previous density function for an assortment of parameters' values. Some properties of the model are as follows. Concerning the expected values of the random variables, they are given by

$$E(X) = \mu, \quad E(Y) = \frac{\alpha}{\beta}.$$

Regarding the variance values, they can be found as

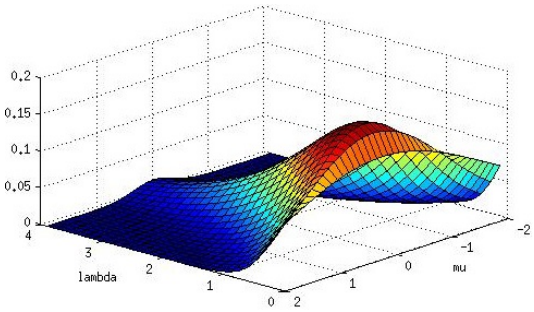
$$V(X) = \frac{\beta}{k(\alpha - 1)}, \quad V(Y) = \frac{\alpha}{\beta^2}.$$

Finally, it can be proven that (2.4) is unimodal with the maximum attained at $\left(\mu, \frac{\alpha - 0.5}{\beta}\right)$.

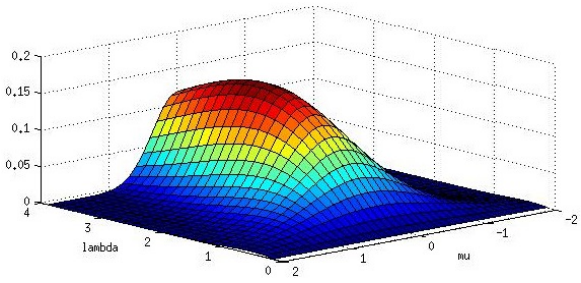
The usefulness of the Normal-Gamma model in Bayesian statistics is that it is a *conjugate* model. This implies that, for a Normal likelihood (like the case considered in this work), a Normal-Gamma prior transforms into a Normal-Gamma posterior distribution, and therefore, the posterior does not need to be numerically calculated. This fact is stated by the next Theorem (see [12]).

Theorem 2.3.1 *Let X follow a Normal distribution, $X \sim N(\mu, \sigma^2)$ and let $\mathbf{x} = (x_1, \ldots, x_n)$ denote a simple random sample of X. Assume that μ and σ^2 follow a Normal-Gamma prior distribution, that is*

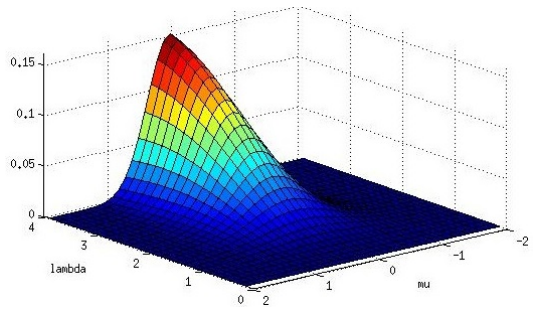
$$\left(\mu, \frac{1}{\sigma^2}\right) \sim NG(\mu_0, k_0, \alpha_0, \beta_0).$$



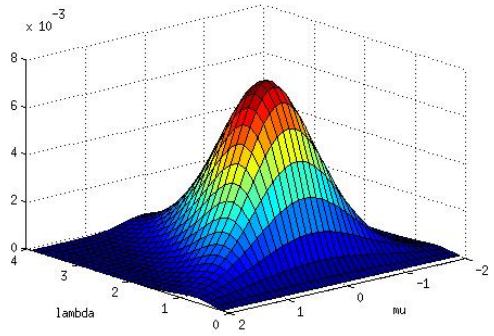
(a) $NG(\mu = 0.1, k = 2, \alpha = 1, \beta = 1)$



(b) $NG(\mu = 0.1, k = 2, \alpha = 3, \beta = 1)$



(c) $NG(\mu = 0.1, k = 2, \alpha = 5, \beta = 1)$



(d) $NG(\mu = 0.1, k = 2, \alpha = 5, \beta = 3)$

Figure 2.1: The probability density function of different Normal-Gamma models

Then, the posterior distribution of X is also a Normal-Gamma distribution, given by

$$\left(\mu, \frac{1}{\sigma^2}\right) \mid \mathbf{x} \sim NG(\mu_n, k_n, \alpha_n, \beta_n),$$

where

$$\mu_n = \frac{k_0\mu_0 + n\bar{\mathbf{x}}}{k_0 + n}$$

$$k_n = k_0 + n$$

$$\alpha_n = \alpha_0 + \frac{n}{2}$$

$$\beta_n = \beta_0 + \frac{1}{2}\sum_{j=1}^{n}(x_j - \bar{\mathbf{x}})^2 + \frac{k_0 n(\bar{\mathbf{x}} - \mu_0)^2}{2(k_0 + n)}.$$

Proof:
The likelihood function is

$$l(\mathbf{x}|\mu, \sigma^2) = \prod_{i=1}^{n} p(x_j|\mu, \sigma^2) =$$

$$\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left\{-\frac{1}{2\sigma^2}\sum_{j=1}^{n}(x_j - \mu)^2\right\}. \tag{2.5}$$

Let λ denote the so-called precision, $\lambda = \frac{1}{\sigma^2}$. Then

$$l(\mathbf{x}|\mu, \sigma^2) = \frac{1}{(2\pi)^{n/2}}\lambda^{n/2}\exp\left\{-\frac{\lambda}{2}\sum_{j=1}^{n}(x_j - \mu)^2\right\}. \tag{2.6}$$

Let $\bar{\mathbf{x}}$ and s^2 denote the empirical mean and variance:

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{j=1}^{n} x_j$$

$$s^2 = \frac{1}{n}\sum_{j=1}^{n}(x_j - \bar{\mathbf{x}})^2.$$

Since

$$\sum_{j=1}^{n}(x_j - \bar{\mathbf{x}})(\mu - \bar{\mathbf{x}}) = (\mu - \bar{\mathbf{x}})\left(\left(\sum_{j=1}^{n} x_j\right) - n\bar{\mathbf{x}}\right) = (\mu - \bar{\mathbf{x}})(n\bar{\mathbf{x}} - n\bar{\mathbf{x}}) = 0.$$

then, it is possible to rewrite the exponential term as follow

$$\sum_{j=1}^{n}(x_j-\mu)^2=\sum_{j=1}^{n}[(x_j-\bar{\mathbf{x}})-(\mu-\bar{\mathbf{x}})]^2=$$

$$=\sum_{j=1}^{n}(x_j-\bar{\mathbf{x}})^2+\sum_{j=1}^{n}(\bar{\mathbf{x}}-\mu)^2-2\sum_{j=1}^{n}(x_j-\bar{\mathbf{x}})(\mu-\bar{\mathbf{x}})=$$

$$=ns^2+n(\bar{\mathbf{x}}-\mu)^2 \tag{2.7}$$

Assume that (μ,λ) follows a Normal-Gamma conjugate prior distribution, then

$$f(\mu,\lambda|\mu_0,k_0,\alpha_0,\beta_0)=\frac{1}{\frac{\Gamma(\alpha_0)}{\beta_0^{\alpha_0}}\left(\frac{2\pi}{k_0}\right)^{\frac{1}{2}}}\lambda^{\alpha_0-\frac{1}{2}}\exp\left(-\frac{\lambda}{2}[k_0(\mu-\mu_0)^2+2\beta_0]\right),$$

where $\mu_0,k_0,\alpha_0,\beta_0$ are assumed to be known. Hence, the posterior density is

$$f(\mu,\lambda|\mathbf{x})\propto f(\mu,\lambda|\mu_0,k_0,\alpha_0,\beta_0)p(\mathbf{x}|\mu,\lambda)$$
$$\propto\lambda^{\frac{1}{2}}e^{-\frac{(k_0\lambda)(\mu-\mu_0)^2}{2}}\lambda^{\alpha_0-1}e^{-\beta_0\lambda}\lambda^{\frac{n}{2}}e^{-\left(\frac{\lambda}{2}\right)\sum_{j=1}^{n}(x_j-\mu)^2}$$

From (2.7)

$$\sum_{j=1}^{n}(x_j-\mu)^2=n(\mu-\bar{\mathbf{x}})^2+\sum_{j=1}^{n}(x_j-\bar{\mathbf{x}})^2.$$

Finally, it can be shown that

$$k_0(\mu-\mu_0)^2+n(\mu-\bar{\mathbf{x}})^2=(k_0+n)(\mu-\mu_n)^2+\frac{k_0n(\bar{\mathbf{x}}-\mu_0)^2}{k_0+n},$$

where

$$\mu_n=\frac{k_0\mu_0+n\bar{\mathbf{x}}}{k_0+n}.$$

Therefore,

$$k_0(\mu-\mu_0)^2+\sum_{j=1}^{n}(x_j-\mu)^2=k_0(\mu-\mu_0)^2+\sum_{j=1}^{n}(x_j-\bar{\mathbf{x}})^2=$$

$$=(k_0+n)(\mu-\mu_n)^2+\frac{k_0n(\bar{\mathbf{x}}-\mu_0)^2}{k_0+n}+\sum_{j=1}^{n}(x_j-\bar{\mathbf{x}})^2.$$

Thus,

$$f(\mu,\lambda|\mathbf{x}) \propto \lambda^{\frac{1}{2}} e^{-\frac{\lambda}{2}(k_0+n)(\mu-\mu_n)^2} \lambda^{\alpha_0+\frac{n}{2}-1} e^{-\beta_0\lambda} e^{-\left(\frac{\lambda}{2}\right)\sum_{j=1}^{n}(x_j-\bar{\mathbf{x}})^2} e^{-\left(\frac{\lambda}{2}\right)\frac{k_0 n(\bar{\mathbf{x}}-\mu_0)^2}{k_0+n}}.$$

In conclusion, the posterior density follows a Normal-Gamma distribution

$$p(\mu,\lambda|\mathbf{x}) = NG(\mu,\lambda|\mu_n,k_n,\alpha_n,\beta_n)$$

with

$$\mu_n = \frac{k_0\mu_0 + n\bar{\mathbf{x}}}{k_0+n}$$

$$k_n = k_0 + n$$

$$\alpha_n = \alpha_0 + \frac{n}{2}$$

$$\beta_n = \beta_0 + \frac{1}{2}\sum_{j=1}^{n}(x_j-\bar{\mathbf{x}})^2 + \frac{k_0 n(\bar{\mathbf{x}}-\mu_0)^2}{2(k_0+n)}.$$

■

2.3.3 A simulation example

Here, we estimate from a Bayesian viewpoint and using the Normal-Gamma model described in the previous section, the probability density in a Gaussian NB classification context. The model parameters are estimated by maximizing the posterior density (*Maximum a posteriori* estimates). In this case, the parameters are Normal-Gamma distributed a priori, and consequently, according to Theorem 2.3.1, they follow the same probability model (with different parameters) a posteriori.

The maximum a posteriori estimate

The maximum a posteriori (MAP) estimates are parameters' point estimates obtained as the mode of the posterior distribution. Therefore, once the posterior distribution is known, then it needs to me maximized. Here, a simplification of the posterior density is found, under the Normal-Gamma setting. Assume first that feature X_i conditioned to the class C_k is Gaussian distributed

$$X_i \mid C_k \sim N(\mu_{i,k}, \sigma^2_{i,k}),$$

where $i = 1, \ldots, p$ and $k = 1, \ldots, K$. Define $\Theta = \{\theta_{i,k}\}_{i=1,\ldots,p,\ k=1,\ldots,K}$, where $\theta_{i,k} = \left(\mu_{i,k}, \sigma^2_{i,k}\right)$. Given a random sample $\mathbf{x}_1, \ldots, \mathbf{x}_p$, then due to the independence assumption of the NB, the posterior distribution will be given by

$$f(\Theta \mid \mathbf{x}_1, \ldots, \mathbf{x}_p) = \prod_{k=1}^{K} \prod_{i=1}^{p} \pi(\theta_{i,k}) f(\mathbf{x}_i^{(k)} | \theta_{i,k}), \tag{2.8}$$

where $\mathbf{x}_i^{(k)}$ is the vector of observations of the feature i for the class k, and $\pi(\cdot)$ and $f(x \mid \cdot)$ denote the prior and likelihood functions, respectively. Under the assumption that the prior distribution of $\theta_{i,k}$ follows a Normal-Gamma model with parameters $\left(\mu_{0,i}^{(k)}, k_{0,i}^{(k)}, \alpha_{0,i}^{(k)}, \beta_{0,i}^{(k)}\right)$ then,

$$\prod_{k=1}^{K} \prod_{i=1}^{p} \pi(\theta_{i,k}) p(\mathbf{x}_i^{(k)} | \theta_{i,k}) =$$

$$= \prod_{k=1}^{K} \prod_{i=1}^{p} \lambda_{i,k}^{\frac{1}{2}} e^{-\frac{\lambda_{i,k}}{2} k_{n_{ik},i}^{(k)} (\mu_{i,k} - \mu_{n_{ik},i}^{(k)})^2} \lambda_{i,k}^{\alpha_{n_{ik},i}^{(k)} - 1} e^{-\lambda_{i,k} \beta_{n_{ik},i}^{(k)}} =$$

$$= \prod_{k=1}^{K} \prod_{i=1}^{p} \lambda_{i,k}^{\alpha_{n_{ik},i}^{(k)} - \frac{1}{2}} e^{\left(-\lambda_{i,k} \left[\frac{k_{n_{ik},i}^{(k)} (\mu_{i,k} - \mu_{n_{ik},i}^{(k)})^2}{2} + \beta_{n_{ik},i}^{(k)} \right] \right)}$$

where n_{ik} denotes the sample size of $\mathbf{x}_i^{(k)}$ and

$$\mu_{n_{ik},i}^{(k)} = \frac{k_{0,i}^{(k)} \mu_{0,i}^{(k)} + n_{ik} \bar{\mathbf{x}}_i^{(k)}}{k_{0,i}^{(k)} + n_{ik}}$$

$$k_{n_{ik},i}^{(k)} = k_{0,i}^{(k)} + n_{ik}$$

$$\alpha_{n_{ik},i}^{(k)} = \alpha_{0,i}^{(k)} + \frac{n_{ik}}{2}$$

$$\beta_{n_{ik},i}^{(k)} = \beta_{0,i}^{(k)} \frac{1}{2} \sum_{j=1}^{n_{ik}} \left(x_i^{(k)}(j) - \bar{\mathbf{x}}_i^{(k)} \right)^2 + \frac{k_{0,i}^{(k)} n_{ik} \left(\bar{\mathbf{x}}_i^{(k)} - \mu_{0,i}^{(k)} \right)^2}{2 \left(k_{0,i}^{(k)} + n_{ik} \right)}$$

The posterior (2.8) can be simplified if logarithms are taken

$$\sum_{k=1}^{K} \sum_{i=1}^{p} \left(\log \left(\lambda_{i,k}^{\alpha_{n_{ik},i}^{(k)} - \frac{1}{2}} \right) - \lambda_{i,k} \left[\frac{k_{n_{ik},i}^{(k)} (\mu_{i,k} - \mu_{n_{ik},i}^{(k)})^2}{2} + \beta_{n_{ik},i}^{(k)} \right] \right). \tag{2.9}$$

Finally, the MAP estimates are defined as

$$\{\hat{\theta}_{i,k}\} = \underset{(\mathbb{R}\times\mathbb{R}^+)^{p\times K}}{\operatorname{argmax}} \sum_{k=1}^{K}\sum_{i=1}^{p}\left(\log\left(\lambda_{i,k}^{\alpha_{n_{ik},i}^{(k)}-\frac{1}{2}}\right) - \lambda_{i,k}\left[\frac{k_{n_{ik},i}^{(k)}(\mu_{i,k}-\mu_{n_{ik},i}^{(k)})^2}{2} + \beta_{n_{ik},i}^{(k)}\right]\right).$$

Implementation with R

For this example, all calculations will be implemented in R and in particular, the maximization of the objective function (2.9) will be done via the **nmkb** function from the **dfoptim** library (see the Appendix for the details). Such function performs a Nelder-Mead algorithm ([13]) for derivative-free optimization which is described next. Assume that a function of p variables is to be minimized without constraints. Then, first the algorithm considers $(p+1)$ p-dimensional initial points, called $P_0, P_1, \ldots, P_p$. Let y_i be the objective function value at P_i, and y_h and y_l the maximum and the minimum values reached, respectively. Let $\overline{P}$ be the centroid of the points with $i \neq h$ and $[P_iP_j]$ the distance from P_i to P_j. Then, at each phase in the process, P_h is substituted by a new point using one of the next operations: reflection, contraction and expansion.In a first place, the algorithm performs a reflection P^* and it checks if y^* have lied between y_h and y_l. In this case, P_h will be replace by P^* and the algorithm starts again. If $y^* < y_l$, that is, a new minimum have been produced by the reflection, it will be perform a expansion P^{**} of P_h and it will check if $y^{**} < y_l$. In this case, P_h will be replace by P^{**} and the algorithm starts again. In other case, P_h is replaced by P^* and the algorithm restarts. Note that, when $y^* > y_i$ $i \neq h$, the algorithm made a contraction, and in the case in which the contraction does not return the expected results, the algorithm replaces all $P_i's$ by $\frac{(P_i+P_l)}{2}$, and the procedure begins again. Finally, when the variation of the values of y is smaller than a certain tolerance, the algorithm ends and it returns the best solution that has found.

Results

A multivariate sample of size 1000 of three independent normal distributions with two classes (positive and negative) is simulated. The vectors of means are given by $(2, 3, 4)$ and $(1.5, 5.4, 2)$ for the positive and negative classes, respectively. The variances are

all set equal to one. That is the parameters of the generating model are given by:

$$\begin{aligned}
\theta_{V1,+} &= (2,1) \\
\theta_{V2,+} &= (3,1) \\
\theta_{V3,+} &= (4,1) \\
\theta_{V1,-} &= (1.5,1) \\
\theta_{V2,-} &= (5.4,1) \\
\theta_{V3,-} &= (2,1)
\end{aligned}$$

We show next how the parameters are correctly estimated by maximizing the posterior density given by (2.9). The prior and posterior parameters for the Normal-Gamma model are found in Tables 2.2-2.5. Note how the parameters change from the prior to the posterior, due to the effect of the likelihood function, especially for k and β.

	V1	V2	V3
Positive	$\mu_0 = 1.003$ $\mu_n = 1.937$	$\mu_0 = 0.223$ $\mu_n = 2.963$	$\mu_0 = 0.654$ $\mu_n = 4.02$
Negative	$\mu_0 = 0.213$ $\mu_n = 1.495$	$\mu_0 = 1.1$ $\mu_n = 5.355$	$\mu_0 = 0.34$ $\mu_n = 2.025$

Table 2.2: Comparison between μ_0 and μ_n.

	V1	V2	V3
Positive	$k_0 = 7$ $k_n = 1007$	$k_0 = 2$ $k_n = 1002$	$k_0 = 5$ $k_n = 1005$
Negative	$k_0 = 8$ $k_n = 1008$	$k_0 = 5$ $k_n = 1005$	$k_0 = 4$ $k_n = 1004$

Table 2.3: Comparison between k_0 and k_n.

	V1	V2	V3
Positive	$\alpha_0 = 6$ $\alpha_n = 506$	$\alpha_0 = 4$ $\alpha_n = 504$	$\alpha_0 = 3$ $\alpha_n = 503$
Negative	$\alpha_0 = 9$ $\alpha_n = 509$	$\alpha_0 = 3$ $\alpha_n = 503$	$\alpha_0 = 5$ $\alpha_n = 505$

Table 2.4: Comparison between α_0 and α_n.

	V1	V2	V3
Positive	$\beta_0 = 0.1$ $\beta_n = 511.543$	$\beta_0 = 0.2$ $\beta_n = 512.756$	$\beta_0 = 0.044$ $\beta_n = 552.540$
Negative	$\beta_0 = 0.994$ $\beta_n = 521.799$	$\beta_0 = 0.6$ $\beta_n = 586.808$	$\beta_0 = 0.1$ $\beta_n = 486.041$

Table 2.5: Comparison between β_0 and β_n.

Under randomly chosen starting points, the final solution is given by

$$\begin{aligned}
\hat{\theta}_{V1,+} &= (1.93, 1.01) \\
\hat{\theta}_{V2,+} &= (2.96, 1.01) \\
\hat{\theta}_{V3,+} &= (4.02, 1.09) \\
\hat{\theta}_{V1,-} &= (1.49, 1.026) \\
\hat{\theta}_{V2,-} &= (5.35, 1.16) \\
\hat{\theta}_{V3,-} &= (2.02, 0.96)
\end{aligned}$$

It should be noted here that the solutions to the problem of maximizing the posterior density, shown previously, are very close to the true modes of the posterior, known to be given by $\left(\mu_{n_{ik}}, \frac{\alpha_{n_{ik}}-0.5}{\beta_{n_{ik}}}\right)$.

2.4 A real dataset example

In this section we implement the Gaussian NB classifier using the well-known Iris dataset. After a short description of the dataset where some statistical properties are shown, a comparison between the frequentist and the Bayesian (in terms of the Normal-Gamma model) estimation methods are given.

2.4.1 Univariate and bivariate analysis

The (Fisher's or Anderson's) iris dataset contains the measurements in centimeters of the variables Sepal Length, Sepal Width, Petal Length and Petal Width, respectively, for 50 flowers from each of 3 species of iris. The species are Setosa, Versicolor, and Virginica, and this dataset contains 50 individuals of each class. In order to better understand the classification results, it may be of interest to study some marginal descriptive statistics for each one of the variables in the set. Table (2.6) shows the maximum and minimum values of each variable as well as the median, mean and coefficient of variation. The results point out to the discriminating power of the variable

related to the petal, since they are more disperse than the other two variables (due to their range is larger). The same behavior can be observed from the marginal histograms in Figure 2.2.

Variable	Minimum	Median	Mean	Maximum	Variation Coefficient
Sepal Length	4.3	5.8	5.843	7.9	0.142
Sepal Width	2	3	3.057	4.4	0.143
Petal Length	1	4.35	1.758	6.9	1
Petal Length	0.1	1.3	1.199	2.5	0.636

Table 2.6: Some descriptive statistics of the Iris dataset.

Figure 2.3 depicts the two-dimensional projections of the multidimensional data where the red points represent the Setosa class, the green points the Versicolor class and the blue points the Virginica class. It looks like most of the variables could be used to predict the species, although it can be observed that the variables Sepal Length and Sepal Width can not separate Virginica and Versicolor classes as well as the others set of variables.

2.4.2 Comparison between the frequentist and Bayesian approaches

The previous sections described the frequentist versus the Bayesian approach (in particular, this last one) in the context of the NB classifier. It seems clear that different estimation approaches will lead to different classifiers, and it is the purpose of this section to explore this issue more in depth.

Design of experiments

The performance results of the NB using the Iris dataset will be calculated under the frequentist and Bayesian inference methods previously discussed. That is, the parameters for the Gaussian model for the features conditioned to the classes will be estimated by both the sample moments, and via the Normal-Gamma conjugate model.

Since the NB considers assumes features, we first computed the MIC coefficients of such features (per classes). The results are shown by the next matrices for the classes Setosa , Virginica and Versicolor, respectively, where SL, SW, PL and PL stand for Sepal Length, Sepal Width, Petal Length and Petal Width.

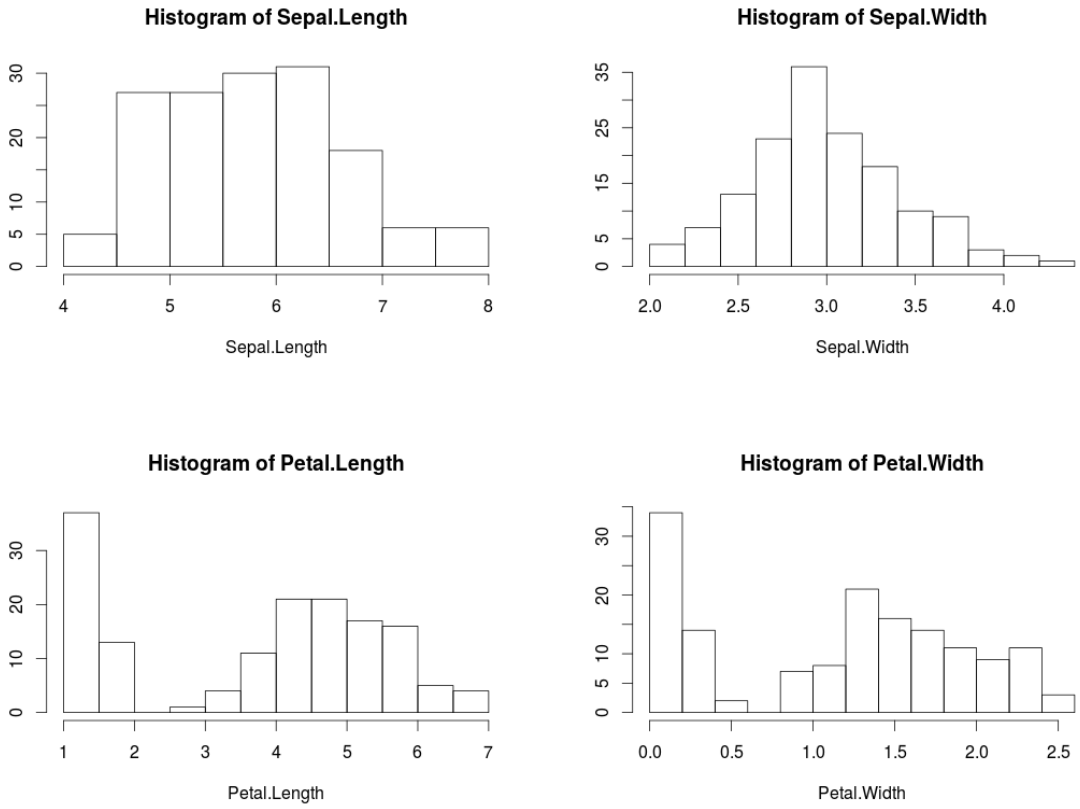


Figure 2.2: Marginal histograms in the Iris data set

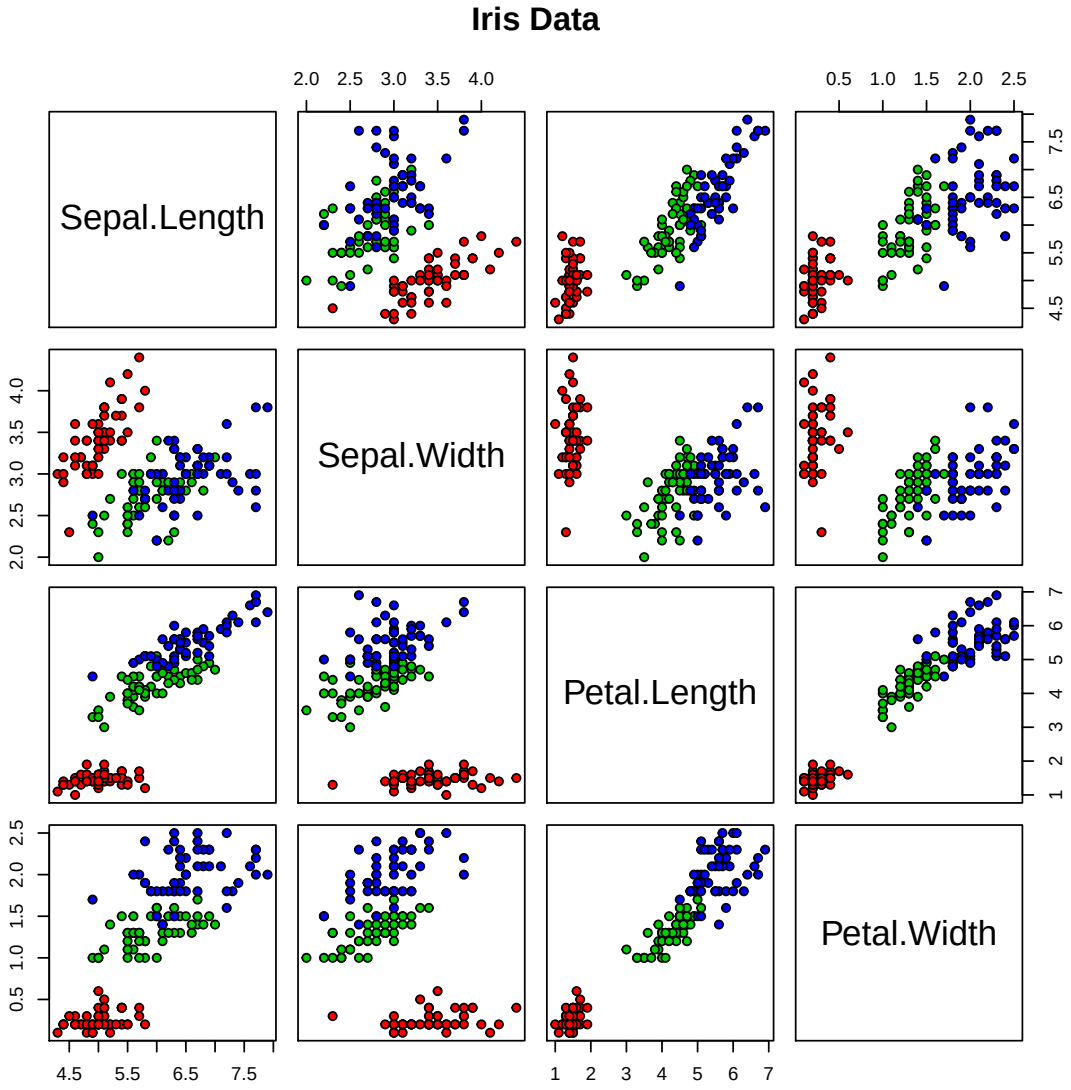


Figure 2.3: Scatterplots for each pair of variables in the Iris dataset

$$\begin{array}{c} \\ SL \\ SW \\ PL \\ PW \end{array} \begin{array}{c} \begin{array}{cccc} SL & SW & PL & PW \end{array} \\ \begin{pmatrix} 0.99 & 0.66 & 0.23 & 0.19 \\ 0.66 & 1 & 0.3 & 0.22 \\ 0.23 & 0.3 & 1 & 0.14 \\ 0.19 & 0.22 & 0.14 & 0.9 \end{pmatrix} \end{array}$$

$$\begin{array}{c} \\ SL \\ SW \\ PL \\ PW \end{array} \begin{array}{c} \begin{array}{cccc} SL & SW & PL & PW \end{array} \\ \begin{pmatrix} 0.99 & 0.40 & 0.62 & 0.29 \\ 0.40 & 0.99 & 0.22 & 0.3 \\ 0.62 & 0.22 & 1 & 0.28 \\ 0.29 & 0.3 & 0.28 & 1 \end{pmatrix} \end{array}$$

$$\begin{array}{c} \\ SL \\ SW \\ PL \\ PW \end{array} \begin{array}{c} \begin{array}{cccc} SL & SW & PL & PW \end{array} \\ \begin{pmatrix} 0.99 & 0.36 & 0.60 & 0.50 \\ 0.36 & 1 & 0.40 & 0.38 \\ 0.6 & 0.39 & 1 & 0.52 \\ 0.50 & 0.38 & 0.52 & 0.99 \end{pmatrix} \end{array}$$

Note from the MIC values how the hypothesis of independence becomes unrealistic in this example (especially for some pairs of variables).

Some details are given next, related to how the numerical experiment will be conducted. First, all possible combinations of variables will be considered. For each combination, we will proceed as follows.

- The full dataset is separated in training dataset and validation samples with samples sizes given by the Table 2.7.

	Setosa	Virginica	Versicolor
Training	35	35	35
Validation	15	15	15

Table 2.7: Distribution of samples in training and validation.

- In order to get reliable results and to ensure that the results do not depend on a certain choice of the training set and validation set, the previous step will be repeated 10 times in a random way, so that the final performance measure will be the average/median of the measures for each fold on the validation set. Then,

– in the case of Classic (frequentist) NB classifier, as the parameters estimates are the sample mean and sample variance, we get them from the training sample. Once these estimations are set, then the correct classification rate for each class will be calculated on the validation dataset. The R command **NaiveBayes** will be used for this case.

– in the case of the Bayesian NB classifier, where the parameters estimates are found as the solution to the maximization of the objective function (2.9) using the training sample, then the correct classification rate for each class will be calculated on the validation dataset. Here, similarly as in Section 2.3., the R command **nmkb** will be used to maximize the posterior density.

Results

We focus now on the classification rates under both estimation methods, shown by Tables 2.8-2.11. In view of the results obtained, it seems that, for the Iris dataset, in most of variable sets the average behavior is more or less similar under both approaches, although the frequentist estimation of parameters of the NB classifier gets more balanced values of correct classification rate for the three classes, while the Bayesian Inference of the parameters returns some cases that are more unbalanced in this aspect. As another observation, the best accuracy has been obtained, in both cases, for the combination *Petal Width* and *Petal Length*. It is important to note that these are independent in the three classes, because of the value 0.52 is the highest value that they take. This clearly points out to the already commented importance of the attributes selection process.

Figures 2.4 and 2.5 depict the boxplots of the estimations for the precision parameter under the frequentist and Bayesian approaches, respectively, for Petal Length and Petal Width variables. Note from the first figure how the classic method leads to more variable results. Note also, how the Bayesian one seems more robust and its results are not very affected by the choice of the starting point in the maximization algorithm. Figures 2.6-2.7 are the analogous to the previous ones for the means. In this case, both estimation methods look more similar.

We found it of interest to visually show how the prior density transformed into the posterior. Figures 2.8-2.9 depict such results for the case of the Petal Length variable conditioned to the Setosa class. Even though the densities belong to the same family of distributions it is clear the change in the modes.

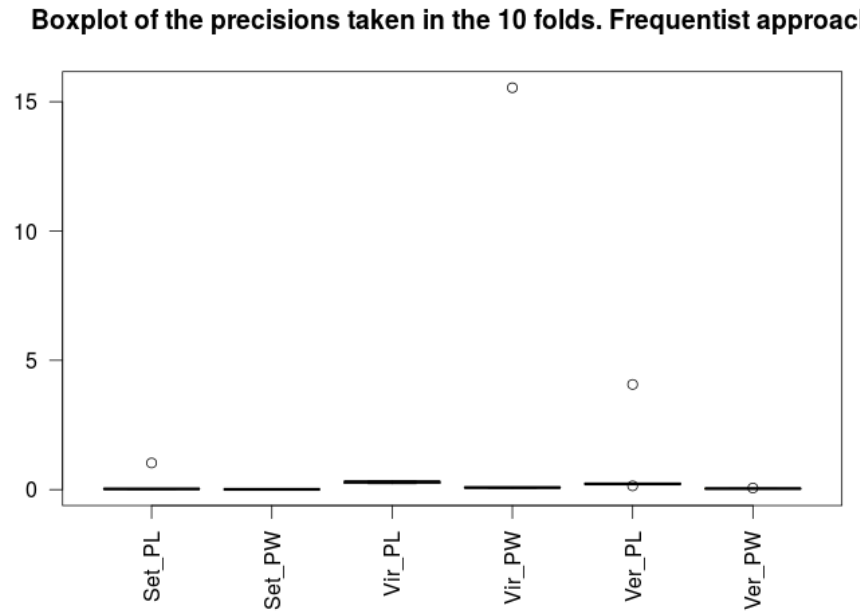


Figure 2.4

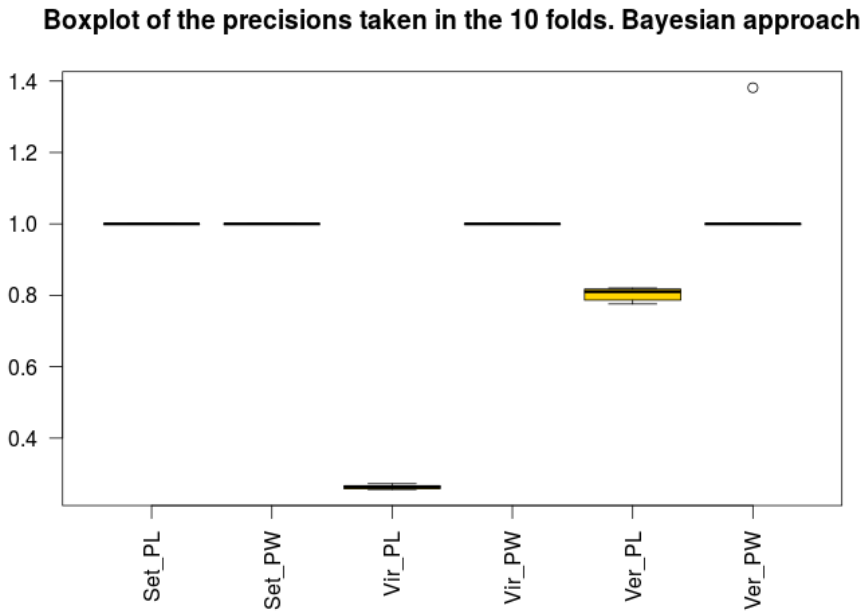


Figure 2.5

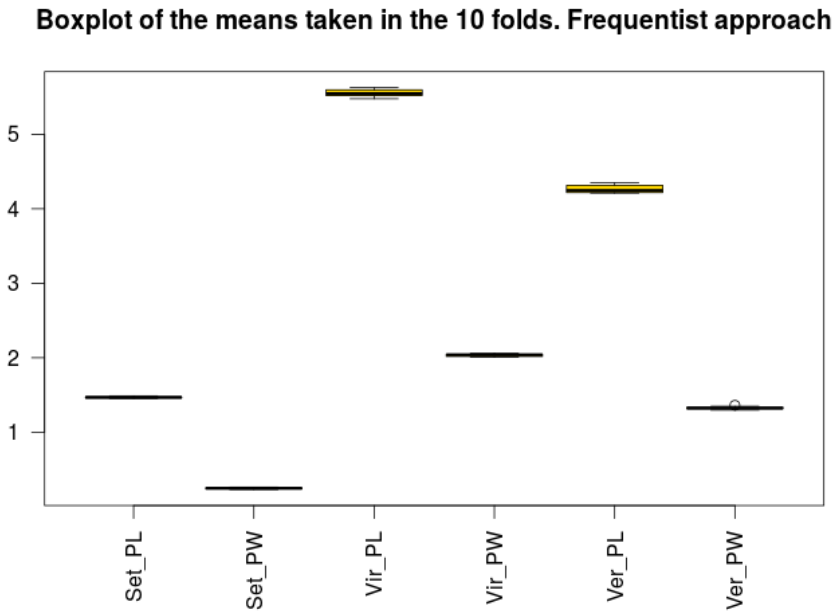


Figure 2.6

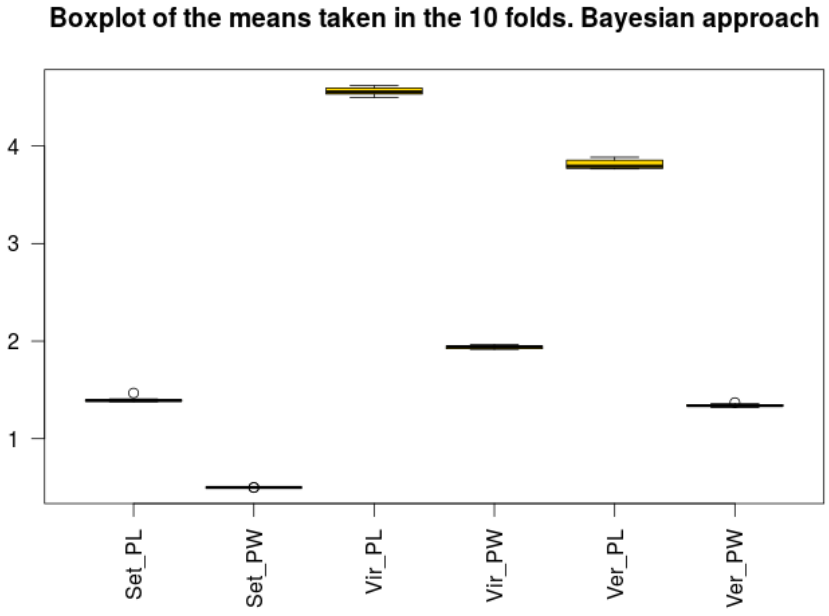


Figure 2.7

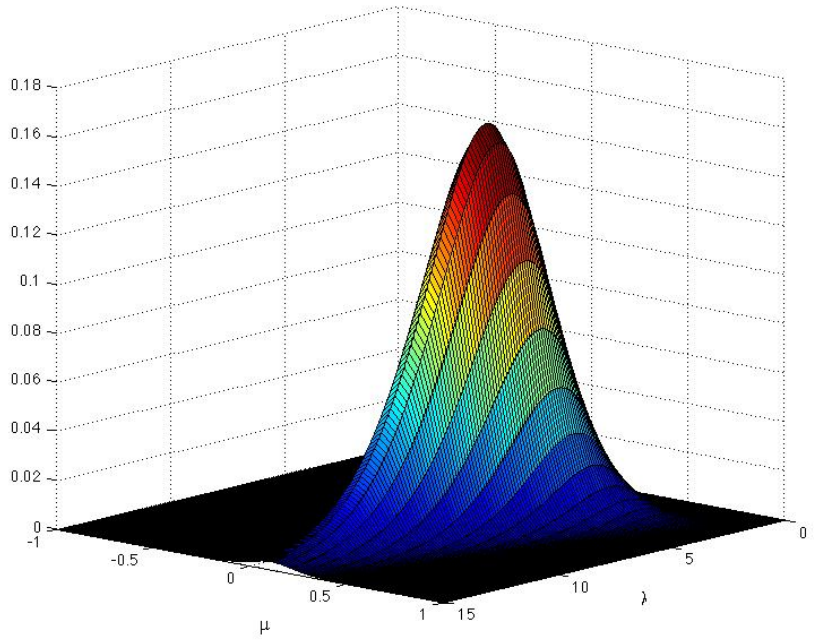


Figure 2.8: Prior density for the Gaussian parameters of the Petal Length variable conditioned to the class Setosa.

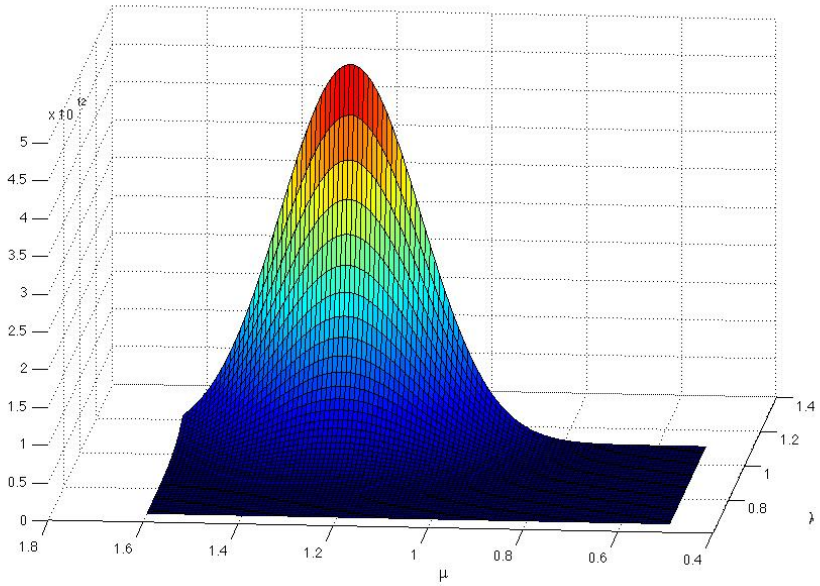


Figure 2.9: Posterior density for the Gaussian parameters of the Petal Length variable conditioned to the class Setosa.

Sepal Length				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	86.66	62	62.67	70.44
Median	86.67	60	66.67	71.11
Variation coefficient	0.14	0.18	0.18	0.08
Sepal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	71.33	42.67	56	56.67
Median	73.33	43.33	56.67	57.78
Variation coefficient	0.13	0.17	0.22	0.1
Petal Length				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	92	93.33	95.11
Median	100	93.33	93.33	94.44
Variation coefficient	0	0.75	0.07	0.02
Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	98	91.33	98	95.78
Median	100	90	100	95.56
Variation coefficient	0.03	0.06	0.03	0.03
Sepal Length & Sepal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	98.67	63.33	74.67	78.89
Median	100	60	76.67	80
Variation Coefficient	0.03	0.14	0.14	0.04
Sepal Length & Petal Length				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	80	87.33	89.11
Median	100	86.67	86.67	87.78
Variation coefficient	0	0.16	0.12	0.04
Sepal Length & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	99.33	91.33	95.33	95.33
Median	100	90	93.33	95.56
Variation coefficient	0.02	0.06	0.03	0.02
Sepal Width & Petal Length				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	82	89.33	90.44
Median	100	83.33	93.33	91.11
Variation coefficient	0	0.13	0.1	0.04
Sepal Width & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	99.33	89.33	94	94.22
Median	100	86.67	96.67	94.44
Variation coefficient	0.02	0.08	0.07	0.03

Table 2.8: Classic Naïve Bayes.

Petal Length & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	93.33	96.67	96.67
Median	100	93.33	100	96.67
Variation coefficient	0	0.07	0.05	0.03
Sepal Length & Sepal Width & Petal Length				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	72.67	87.33	86.67
Median	100	80	86.67	86.67
Variation coefficient	0	0.15	0.12	0.05
Sepal Length & Sepal Width & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	91.33	88.67	93.33
Median	100	90	86.67	94.44
Variation coefficient	0	0.06	0.09	0.03
Sepal Length & Petal Length & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	93.33	93.33	95.56
Median	100	93.33	93.33	95.56
Variation coefficient	0	0.07	0.06	0.02
Sepal Width & Petal Length & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	92.67	96.67	96.44
Median	100	90	100	96.67
Variation coefficient	0	0.07	0.05	0.03
Sepal Length & Sepal Width & Petal Length & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	92.67	94.67	95.78
Median	100	90	93.33	95.56
Variation coefficient	0	0.07	0.04	0.02

Table 2.9: Classic Naïve Bayes

Sepal Length				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	44	25.33	98	55.78
Median	46.67	26.67	100	56.67
Variation coefficient	0.3	0.35	0.05	0.09
Sepal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	98	0	40	46
Median	100	0	36.67	44.44
Variation coefficient	0.03	0	0.02	0.06
Petal Length				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	83.33	98.67	94
Median	100	86.67	100	94.44
Variation coefficient	0	0.14	0.03	0.04
Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	98	69.33	89.11
Median	100	100	73.33	88.89
Variation coefficient	0	0.03	0.12	0.03
Sepal Length & Sepal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	97.33	42	91.33	76.89
Median	100	40	93.33	75.56
Variation coefficient	0.04	0.2	0.08	0.05
Sepal Length & Petal Length				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	94	86.67	93.56
Median	100	93.33	86.67	93.33
Variation coefficient	0	0.07	0.11	0.03
Sepal Length & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	60	100	86.67
Median	100	60	100	86.67
Variation coefficient	0	0.1	0	0.02
Sepal Width & Petal Length				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	99.33	62.67	87.33
Median	100	100	63.33	87.78
Variation coefficient	0	0.02	0.3	0.07
Sepal Width & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	90	98	96
Median	100	86.67	100	95.56
Variation coefficient	0	0.07	0.03	0.03

Table 2.10: Naïve Bayes and Bayesian Inference.

Petal Length & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	88.67	99.33	96
Median	100	93.33	100	96.67
Variation coefficient	0	0.1	0.02	0.03
Sepal Length & Sepal Width & Petal Length				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	68	90.67	86.22
Median	100	66.67	90	86.67
Variation coefficient	0	0.14	0.09	0.02
Sepal Length & Sepal Width & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	69.33	83.33	84.22
Median	100	70	80	84.44
Variation coefficient	0	0.09	0.14	0.03
Sepal Length & Petal Length & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	98	81.33	93.11
Median	100	100	76.67	92.22
Variation coefficient	0	0.03	0.14	0.04
Sepal Width & Petal Length & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	100	60.67	86.89
Median	100	100	60	86.67
Variation coefficient	0	0	0.26	0.06
Sepal Length & Sepal Width & Petal Length & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	92.67	85.33	92.67
Median	100	96.67	83.33	92.22
Variation coefficient	0	0.1	0.12	0.02

Table 2.11: Naïve Bayes and Bayesian Inference

Chapter 3

The constrained Naïve Bayes

3.1 Motivation and formulation of the optimization problem

There is a number of approaches in the literature modifying standard classification methods, but there is a lack of methodologies allowing the user to control simultaneously the different performance measures of interest.

The application of mathematical optimization tools, as described in [4], seems to be a promising and not fully explored option: one overall criterion is to be optimized, while constraints are introduced in the model to impose acceptable values for the estimates of the many performance measures under consideration.

NB is especially appropriate to accommodate constraints. Deterministic performance constraints (e.g. for each class, the correct classification rate on a given independent sample must be above a threshold value) are formulated and it would allow us not only to obtain competitive estimates of the performance measures, but also to control the achievement in the different individual performance measures under consideration.

We analyze how to pose the associated optimization problem if a NB is used as base classifier. It will be of interest to identify how estimates of the parameters densities are to be derived if, as mentioned above, performance constraints are included in the model.

Hence in this work we propose to obtain bayesian estimates of the model parameters $\Theta = \{\theta_{i,k}\}$, $i = 1, \ldots, p$, $k = 1, \ldots, K$ is such a way that the performance of the NB when classifying new observations is increased. Specifically, let us consider a set of

features, assumed be independent (by a previous clustering based on MIC criterion) and given a training sample $\mathbf{x}_1, \ldots, \mathbf{x}_p$ (each $\mathbf{x}_i$ is a vector of observations of the feature i), we propose to obtain Θ as the solution of the following optimization problem with constraints concerning the performance measures:

$$\max_{\Theta} \ \pi(\Theta)p(\mathbf{x}_1, \ldots, \mathbf{x}_p|\Theta)$$
$$\text{s.t.} \begin{cases} m_1(\mathbf{x}, \Theta) \geq a_1, \\ \vdots \\ m_J(\mathbf{x}, \Theta) \geq a_J \end{cases} \tag{3.1}$$

where

$$\pi(\Theta)p(\mathbf{x}_1, \ldots, \mathbf{x}_p|\Theta) = \prod_{k=1}^{K}\prod_{i=1}^{p} \pi(\theta_{i,k})p(\mathbf{x}_i^{(k)}|\theta_{i,k}) =$$
$$= \prod_{k=1}^{K}\prod_{i=1}^{p} \lambda_{i,k}^{\alpha_{n_{ik},i}^{(k)}-\frac{1}{2}} e^{\left(-\lambda_{i,k}\left[\frac{k_{n_{ik},i}^{(k)}(\mu_{i,k}-\mu_{n_{ik},i}^{(k)})^2}{2}+\beta_{n_{ik},i}^{(k)}\right]\right)}$$

with

$$n_{ik} \equiv \text{number of observations of } \mathbf{x}_i \text{ beloging to the class } k.$$

$$\mu_{n_{ik},i}^{(k)} = \frac{k_{0,i}^{(k)}\mu_{0,i}^{(k)} + n_{ik}\bar{\mathbf{x}}_i^{(k)}}{k_{0,i}^{(k)} + n_{ik}}$$

$$k_{n_{ik},i}^{(k)} = k_{0,i}^{(k)} + n_{ik}$$

$$\alpha_{n_{ik},i}^{(k)} = \alpha_{0,i}^{(k)} + \frac{n_{ik}}{2}$$

$$\beta_{n_{ik},i}^{(k)} = \beta_{0,i}^{(k)}\frac{1}{2}\sum_{j=1}^{n_{ik}}(x_i^{(k)}(j) - \bar{\mathbf{x}}_i^{(k)})^2 + \frac{k_{0,i}^{(k)}n_{ik}(\bar{\mathbf{x}}_i^{(k)} - \mu_{0,i}^{(k)})^2}{2(k_{0,i}^{(k)} + n_{ik})}$$

and with $k_{0,i}^{(k)}$, $\alpha_{0,i}^{(k)}$, $\beta_{0,i}^{(k)}$ and $\mu_{0,i}^{(k)}$ known $\forall i$, $\forall k$.
That is, our objective function is a product of $NG(\mu_{i,k}, \lambda_{i,k}|\mu_{n_{ik},i}^{(k)}, k_{n_{ik},i}^{(k)}, \alpha_{n_{ik},i}^{(k)}, \beta_{n_{ik},i}^{(k)})$, and the performance measure is controlled through the constraints.

Note that the posterior density is constructed by multiplying a selected prior density with the likelihood function defined from the observed dataset. So the obtained

solution (estimation of Θ) is a MAP (maximum a posteriori) estimator, in the constrained space. The sequence m_j, $j = 1, \ldots, J$ denotes a set of performance measures (some of those discussed above) which are functions of θ and are evaluated at sample $\mathbf{x}$. Finally, the values $a_1, \ldots, a_J$ are fixed lower-bounds of the performance measures.

The choice of the constraints for our model may vary depending on the aim that is pursued.

It is important to note that the constraints considered depend on the classifier, that is, these constraints must be calculated using the classifier in question, or similarly, expressions for m_j $\forall j \in 1, \ldots, J$ dependent on the Naïve classifier.

3.2 An algorithm to solve the optimization problem

As in the previous simulation, in order to obtain estimates of the parameters, it is again necessary to optimize the objective function, but in this case taking into account the constraints.

For this type of optimization problem the **snomadr** function from **crs** library in R will be used, this function is an implementation of the Mesh Adaptative Direct Search algorithm that is designed for optimization problems with blackbox functions as constraints. For more details concerning these commands, see the Appendix.

The Mesh Adaptative Direct Search algorithm (MADS), see [2], is a class of algorithms minimizing nonsmooth functions f,

$$f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\},$$

under general constraints $x \in \Omega \neq \emptyset \subseteq \mathbb{R}^n$, where Ω is the feasible region, it is characterized because it does not need neither calculate nor approximate the derivatives of f. Then, this algorithm is useful when the derivative of the objective function can not be derived or estimated. Moreover, the feasible region can be defined using blackbox constraints given by an oracle that returns if a certain point is feasible or not. Note that this last fact is very useful in our case, because of we will work with blackbox constraints.
So, given $x_0 \in \Omega$, an initial point, MADS algorithm tries to find a minimum of f over Ω by computing f at a set of trial points from Ω. MADS is an iterative algorithm. At each iteration, a finite set of trial points are selected and evaluated in the objective function f. Once these calculations have been made, they are compared with the best feasible objective function value found so far. The trial points have selected on the

mesh of the iteration l, constructed from a finite set of scaled directions on $\mathbb{R}^n$ that are fixed according to some restrictions.

3.3 Numerical example

We will design the following experiment in the same conditions as those explained in Section 2.4.2: for each combination of variables, we perform the NB classification with Bayesian inference of the parameters including constraints on the training dataset, and, finally, we will compare the results obtained in these experiments on the validation dataset with the results obtained in tables 2.10 and 2.11, that is, with the results that NB classifier with Bayesian inference of the parameters but without constraints.

Let us introduce some constraints in order to study how the constrained NB yields different results according to the requirements imposed. It is very important to note that the values of a_i, for i from one to the number of constraints, that have to be fixed in the optimization problem (3.1) may make the problem infeasible. Note again that the results that will be achieved from this optimization problem not only will depend on the values of a_i, but also of the parameters used in Normal-Gamma prior distributions.

In this example we wanted to classify very well one of the two classes that seems to be more mixed, that is, we have imposed a correct classification rate (see formula (1.1)) of Virginica class of 0.95, although we admit a correct classification rate of Versicolor class of 0.60. And, for Setosa class we want to exceed the 0.75. Note that 0.95, 0.60 and 0.75 are the values of a_i $i = 1, 2, 3$, so, these constraints can be written as follows:

$$CCR_{Virginica} = \frac{\#\{Correct\ classifications\ of\ Virginica\ class\}}{\#\{Individuals\ from\ Virginica\ class\}} \geq 0.95 \qquad (3.2)$$

$$CCR_{Versicolor} = \frac{\#\{Correct\ classifications\ of\ Versicolor\ class\}}{\#\{Individuals\ from\ Versicolor\ class\}} \geq 0.60 \qquad (3.3)$$

$$CCR_{Setosa} = \frac{\#\{Correct\ classifications\ of\ Setosa\ class\}}{\#\{Individuals\ from\ Setosa\ class\}} \geq 0.75 \qquad (3.4)$$

Now, if we observe the results of tables 3.1 and 3.2 and, if we compare with the previous results obtained in tables 2.10 and 2.11, it is clear that, in these last results, the Virginica class is better classified, and the results obtained satisfy the imposed constraints, although there are some sets of variables yielding infeasible problems (Sepal

Length, Sepal Width and both together).

Hence, the best results in terms of overall mean or median accuracy satisfying (3.2)-(3.4) are obtained if, instead of the four variables, only the variables Petal Length and Petal Width are used.

Sepal Length				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	-	-	-	-
Median	-	-	-	-
Variation coefficient	-	-	-	-
Sepal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	-	-	-	-
Median	-	-	-	-
Variation coefficient	-	-	-	-
Petal Length				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	96	88	94.67
Median	100	100	86.67	93.33
Variation coefficient	0	0.07	0.08	0.02
Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	98	69.33	89.11
Median	100	100	73.33	88.89
Variation coefficient	0	0.03	0.12	0.03
Sepal Length & Sepal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	-	-	-	-
Median	-	-	-	-
Variation coefficient	-	-	-	-
Sepal Length & Petal Length				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	98	76.67	91.56
Median	100	100	80	93.33
Variation coefficient	0	0.03	0.2	0.05
Sepal Length & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	87.33	96.67	60.67	81.56
Median	90	100	66.67	82.22
Variation coefficient	0.16	0.05	0.18	0.07
Sepal Width & Petal Length				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	98.67	72.67	90.44
Median	100	100	73.33	91.11
Variation coefficient	0	0.03	0.25	0.06
Sepal Width & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	92	98	63.33	84.44
Median	100	100	66.67	85.56
Variation coefficient	0.19	0.03	0.3	0.09

Table 3.1: Constrained Naïve Bayes and Bayesian Inference.

Petal Length & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	98	88	95.33
Median	100	100	90	95.56
Variation coefficient	0	0.05	0.13	0.03
Sepal Length & Sepal Width & Petal Length				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	96	72	89.33
Median	100	96.67	73.33	88.89
Variation coefficient	0	0.05	0.22	0.06
Sepal Length & Sepal Width & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	90.67	92.67	54.67	79.33
Median	96.67	93.33	56.67	78.89
Variation coefficient	0.15	0.11	0.26	0.05
Sepal Length & Petal Length & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	98	58.67	85.56
Median	100	100	60	85.56
Variation coefficient	0	0.03	0.29	0.07
Sepal Width & Petal Length & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	99.33	98	73.33	90.22
Median	100	100	70	90
Variation coefficient	0.02	0.05	0.2	0.05
Sepal Length & Sepal Width & Petal Length & Petal Width				
	CCR Setosa	CCR Virginica	CCR Versicolor	Accuracy
Mean	100	96.67	65.33	87.33
Median	100	100	63.33	86.67
Variation coefficient	0	0.05	0.28	0.06

Table 3.2: Constrained Naïve Bayes and Bayesian Inference.

Conclusions

Three main conclusions emerge from the study on the NB classifier that has been carried out in this work. First, NB classifier assumes independence of variables. When it is applied with dependent variables, the full set of variables may not yield the best results in terms of accuracy. This is illustrated by the example of the Iris dataset. In fact, Feature Selection, that has not been considered in this work beyond the complete enumeration of set of variables in the same example, is concluded to be necessary.

Another main conclusion is that the different parameters estimation methods induce, of course, different classifiers. This work does not have as an objective to indicate which one is right (if the frequentist approach, if the Bayesian approach), but to show with an example that different methods induce different classifiers. Although, regarding the Bayesian approach, it remains to be done an exhaustive study of the sensitivity priors, because different a priori estimates will lead to different estimates values and therefore to different classifiers.

The novelty of this work is the inclusion of constraints on the classical Bayesian approach. These constraints have provided a different classifier, with the difference that this new approach allows to control the performance measures, which is fundamental in many realworld classification problems.

Appendix: R routines

Naïve Bayes routines

The language and environment R (see [14]) provides the following two packages relating to Naïve Bayes classifier:

- The package **e1071**, that provides the functions
  - *naiveBayes*: to perform Naïve Bayes classification.
  - *predict.naiveBayes*: to obtain the predictions.
- The package **klaR** that has the following function to perform Naïve Bayes classification and to predict
  - *NaiveBayes*
  - *predict.NaiveBayes*

In next subsections, let us explain in more detail the previous functions.

Package "e1071"

Function *naiveBayes* can be called in two different ways, that is, with different type of arguments. The first one would be:

naiveBayes(x, y, laplace=0, subset, na.action=na.pass)

- **x**: Numerical matrix or data-frame of numeric variables and/or categorical.
- **y**: Vector of classes.
- **laplace**: Laplace smoothing parameter to improve the estimates of the probabilities in the case of categorical data (by default, it does not apply).

- **subset**: Vector of indices specifying instances of the sample learning to use.
- **na.action**: Action to perform in the presence of missing values (NA).

And the second one is:

naiveBayes(formula, data, laplace=0, subset, na.action=na.pass)

- **formula**: Formula in the way $class \sim x_1 + x_2 + \ldots$
- **data**: Learning sample data-frame or contingency table with the frequencies resulting from the count.
- **laplace**,**subset**, **na.action**: As before.

On the other hand, the function *naiveBayes* returns an object with the next components:

- **apriori**: Probabities of each class of the response variable.
- **tables**: List of tables, one for each predictor variable. According the type of it, the information obtained is different:
  - *Qualitative*: the table contains the probabilities of each class of the output variable conditional on the different modalities of the predictor variables.
  - *Quantitative*: for each class the output variable, the table contains the mean and standard deviation of the predictor variable conditioned to that class.
- **levels**: Output variable classes.
- **call**: Sequence of the call to the *naiveBayes* function.

Finally, once the classifier has been constructed according to the previous functions, predictions can be made:

predict(object, newdata, type=class,threshold=0.001, eps=0)

where

- **object**: Object generated by the *naiveBayes* function.
- **newdata**: Data-frame with the instances to predict.

- **type**: If *raw* is selected, the function returns the probability of each class of output variable (for each instance to predict). If *class* is selected, it only indicates the class with maximum probability for each instance to predict.
- **threshold**: Value for which the probabilities are replaced below the value specified in **eps**.
- **eps**: The lower probabilities than this value will be replaced by **threshold**.

Package "klaR"

There exists another function called *NaiveBayes* from **klaR**. The main differences between the *naiveBayes* function (**e1071**) and the *NaiveBayes* function (**klaR**) are the following:

- *NaiveBayes* allows one to specify prior probabilities of classes of the response variable. In *naiveBayes* they are estimated from the learning sample.
- *NaiveBayes* lets one use kernel function to estimate the density function of continuous variables. In *naiveBayes* always it is assumed that this kind of variables follows a Normal distribution.
- *NaiveBayes* requires learning sample given as matrix or data-frame (it does not support contingency table).

MIC routine

R provides a function that performs the calculation of the MIC. The library **miverva** has a function called *mine*, which returns the calculation of the MIC for a data set given.

The principal argument needed is **x**, a numeric vector, matrix or data frame (which is coerced to matrix). The rest of arguments are optional and they are explained in the manual provided by R. This function returns, among other outputs, the Maximal Information Coefficient (MIC).

Optimization Routines

In this section two different functions from two distinct packages will be explained. First, it will be described a function that is useful when unconstrained optimization problems are carried out. The second one allows one to include constraints to the

optimization problem and, moreover, both constraints as the objective function can be given as blackbox functions.

Package "dfoptim"

The library **dfoptim** has a function called *nmkb*. This function is an implementation of the Nelder-Mead algorithm for derivative-free optimization. It allows bounds over the parameters. The arguments of this function are:

nmkb(par, fn, lower=-Inf, upper=Inf, control = list(), ...)

where

- **par**: A initial vector of parameters. Note that it must be between lower and upper bounds (if they exist).
- **fn**: The nonlinear objective function to be optimized.
- **lower** and **upper**: lower and upper bounds on the parameters.
- **control**: a list with the control parameters such as tolerance, maximum number of objective function evaluations allowed, ...
- **...**: auxiliary arguments of the objective function.

Package "crs"

The library **crs** has a function called *snomadr*. This last function *snomadr* is an R interface to NOMAD (Nonsmooth Optimization by Mesh Adaptive Direct Search) ([1]), an open source software C++ implementation of the Mesh Adaptive Direct Search algorithm designed for constrained optimization of blackbox functions.
This function needs the following input arguments:

- **eval.f**: the function that returns the value of the objective function and the value of the constraints.
- **n**: the number of variables.
- **bbin**: a vector that indicates the type of the variables.
- **bbout**: a vector that fixed how the constraints have to be considered during the optimization process.

- **x0**: the initial vector of parameters.
- **lb** and **ub**: vectors with lower and upper bounds of the controls, respectively.
- **...**: auxiliary arguments that will be needed to the objective and constraints functions.

Note that this function has some additional arguments which are explained in the help of R.

Bibliography

[1] C. Audet, S. Le Digabel, and C. Tribes. NOMAD user guide. Technical Report G-2009-37, Les cahiers du GERAD, 2009.

[2] Charles Audet and John E Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on optimization*, 17(1):188–217, 2006.

[3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] Emilio Carrizosa and Dolores Romero Morales. Supervised classification and mathematical optimization. *Computers & Operations Research*, 40(1):150–165, 2013.

[5] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[6] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[7] Terrence S Furey, Nello Cristianini, Nigel Duffy, David W Bednarski, Michel Schummer, and David Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.

[8] David J Hand and Keming Yu. Idiot's bayes - not so stupid after all? *International statistical review*, 69(3):385–398, 2001.

[9] EH Linfoot. An informational measure of correlation. *Information and control*, 1(1):85–89, 1957.

[10] Neha Mehra and Surendra Gupta. Survey on multiclass classification methods. 2013.

[11] Kevin P Murphy. Naive bayes classifiers. *University of British Columbia*, 2006.

[12] Kevin P Murphy. Conjugate bayesian analysis of the gaussian distribution. Technical report, The University of British Columbia, 2007.

[13] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[14] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[15] David N Reshef, Yakir A Reshef, Hilary K Finucane, Sharon R Grossman, Gilean McVean, Peter J Turnbaugh, Eric S Lander, Michael Mitzenmacher, and Pardis C Sabeti. Detecting novel associations in large data sets. *science*, 334(6062):1518–1524, 2011.

[16] John A Richards. Classifier performance and map accuracy. *Remote Sensing of Environment*, 57(3):161–166, 1996.

[17] Lior Rokach and Oded Maimon. *Data mining with decision trees: theory and applications*. World scientific, 2014.

[18] Terry Speed et al. A correlation for the 21st century. *Science*, 334(6062):1502–1503, 2011.

[19] Brani Vidakovic. *Statistics for bioengineering sciences: with MATLAB and WinBUGS support*. Springer Science & Business Media, 2011.