Naïve Bayes classifier: normally distributed continuous attributes versus the discretized version of those attributes

Bachelor thesis 7.5 ECTS

I. Majoor
5516013

Supervisor: dr. S. Renooij
Second reviewer: dr. ir. J.M. Broersen

Bachelor Artificial Intelligence
Utrecht University
30 – 06 – 2017

Abstract

In this paper, the differences between training a Naïve Bayes classifier on normally distributed continuous attributes and training a Naïve Bayes classifier on the discretized version of those continuous attributes have been examined. First, the methods that have been used in the experiment have been chosen carefully. To test if an attribute has a normal distribution, the Shapiro-Wilk test was executed. The discretization has taken place with the unsupervised method equal frequency and a supervised method using the minimum description length principle. Monte-Carlo cross validation was used to get three means of the percentages wrongly classified unseen instances per dataset after 25 runs: when using the continuous attributes themselves and when the attributes were discretized using the two discretization methods. In the results, the means have been tested with the paired t-test. The conclusion is to keep the continuous attributes when dealing with a larger dataset (2280 till 5000 instances) as there were less unseen instances wrongly classified and the true difference between the means was significant. When dealing with a smaller dataset (114 till 250 instances), the true difference between the means was not significant. Only when a smaller dataset has an unbalanced split of number of instances per different classification with a ratio of 1:2.5, keeping the normally distributed continuous attributes resulted in a better accuracy.

Content

1. Introduction

More and more companies make use of artificial intelligence to make the decisions that have to be taken based on a lot of data. With machine learning, it is possible to calculate the best option in a particular situation. Take for example a bank. A person will only get a loan if it is likely that he can repay it. Otherwise, the risk is too high and the bank can lose a lot of money. Classification is a good method to check if the person's risk is too high. Based on the person's data, like income, debt and home-situation, and historical data of all the customers who received a loan in the past (along with an indication if the customer could repay the loan or not), it can be calculated which situation has a greater chance of happening: the person will be able to repay his new loan or he won't be able to repay his new loan. As a result, the bank will now be able to get the most out of their money.

One classifier to do this, is the Naïve Bayes classifier. The goal of the classifier is to assign a classification to a new instance. A classification is the category to which an instance belongs. If we want to classify the sex of a person based on his/her height and weight, the classification can be "male" or "female". To get this classification we use the attributes "height" and "weight": the properties that belongs to the instance. The person itself is the instance: an observation that has its own values for all the attributes. If you take multiple instances, including their corresponding classifications, you will get a dataset the Naïve Bayes classifier can train on. Now, whenever you receive a new instance without a classification, you can use the trained Naïve Bayes classifier to get the classification the new instance most likely belongs to. The Naïve Bayes classifier is useful for problems in real-life, like recognizing end-user transactions [1]. As the Naïve Bayes classifier is a useful classification method, it is good to know how to use it in different situations to get the best results.

1.1 Continuous vs Categorical

The type of the attributes the Naïve Bayes classifier trains on may differ. It could be numeric or categorical. A numeric attribute could contain an infinite number of values, continuous, or a finite number of values, discrete. A categorical attribute has a value that describes what it is: "low" or "high", "A" or "B" or "C", and so on. A categorical attribute does not have to be a word or letter, it can also be numeric like the categories 1 to 5. It is possible to replace the infinite number of continuous values with a finite number of categorical values using discretization. For example, instead of having a different height for each person, the top twenty percent can be classified as "tall", the bottom twenty percent as "short" and the rest as "average".

Using discretization on continuous attributes can lead to wrong results as there is data loss in most cases [2]. For continuous attributes, a normal distribution must be assumed when using the Naïve Bayes classifier, even when it is verified that an attribute does not have a normal distribution. There has been an experiment in which a comparison has been made between continuous attributes and a discretized version of those attributes, to find out which one has the highest classification accuracy for unseen instances using the Naïve Bayes classifier. As discretization can be done in many ways, they discretized the attributes with not one, but four different methods. The results have shown that the discretized attributes, no matter which method has been used, had a higher accuracy than the continuous values [3]. The authors are

dealing with both the data loss due to discretization and the obligatory assumption of having continuous attributes. It was not verified that the continuous attributes of the used datasets were normally distributed. In fact, the authors have said that normally distributed assumption was “inappropriate”. This could be a reason that the accuracy of the Naïve Bayes classifier on the datasets with the continuous attributes is lower than on the datasets with the discretized attributes as a wrong assumption may lead to wrong results.

In this paper, it will be clear if the data loss due to discretization is suddenly much more prominent when the normal distribution assumption is true. There will also be a comparison between continuous attributes and the discretized version of those attributes, to find out which one has the highest classification accuracy for unseen instances using Naïve Bayes. However, it will not be assumed that the attributes have a normal distribution; before a data set is used, it will be verified that the distribution is normal. Our research question is: if it is verified that the continuous attributes of a dataset have a normal distribution, what will have a higher accuracy: using the Naïve Bayes classifier on the continuous attributes or using the Naïve Bayes classifier on the discretized version of those attributes? Now, only the data loss due to discretization can influence the results.

In the following section, we shall talk about the relevance of this paper and an explanation for the methods we shall use during the experiment. In section 3, we will go deeper into the experiment itself: the datasets we will use, the set-up and the results. The conclusion and possible future research are discussed in the final section.

2. Preliminaries

2.1 Central Limit Theorem

The conclusion to this experiment will only be relevant to the datasets that contain normally distributed continuous attributes. The next question follows: does a normal distribution really occur in real life? Because, if no normally distributed continuous attributes appear in real life data, none of the future datasets will benefit from our conclusion. Except for datasets that have specifically been generated to have a normal distribution of course. To get an explanation for the occurrence of normal distributions in real life data, we can use the central limit theorem: the distribution of the sum (or the mean) of a lot of independent, identically distributed variables will be approximately normal. The variables are independent if they do not influence each other: if you change the value of one of the variables, the values of the other variables should remain the same. The variables are identically distributed if every instance has the same probability distribution for all of the variables [4].

In [5], an example of the central limit theorem is given. The weight of loaves of bread will vary according to the normal distribution if you use the same recipe for all the loaves of bread. When you make 100 loaves of bread, it is impossible to follow the recipe exactly the same every single time. Maybe you have added 201 grams of flour to the first loaf, but only 198 grams to the next one. Also, more milk can remain in the measuring cup after you have poured, than remained last time. The sum of those variables is the weight of the loaf. The distribution of all those sums (weights) will approach the normal distribution. This shows us that we face normally distributed data in real life that could benefit from our

experiment. If we would bake 100 loaves of bread from not one, but two different recipes, we could use a classifier to find out if an unknown bread is baked according to recipe one or two. After we have an answer to our research question, we should be able to decide whether to use the classifier with the continuous values for attribute weight or with a discretized version of those continuous values.

Not only the weight of baked food follows a normal distribution, many continuous attributes closely follow a normal distribution. The data set obtained from [6], contains the height and weight of 25.000 humans. After using our normality test (see Normally Distributed), we can assume that human heights are normally distributed. The minimum value is 60.3 inch (153.1 cm), the maximum value is 75.2 inch (190.9 cm), the mean is 68.0 inch (172.7 cm) and the standard deviation is 1.9 inch (4.8 cm). There are a lot of factors, the variables, which influence your height, the sum of those variables [7]. Not only your height is normally distributed. For example: birthweights, IQ scores and SAT scores also follow this distribution [8]. This means that our experiment will be relevant for the Naïve Bayes classifier in practice.

2.2 The Naïve Bayes classifier

The Naïve Bayes classifier is a statistical classifier. A trained Naïve Bayes classifier can predict the classification of an unseen instance based on the instance's values for the attributes. The prediction will be based on the probability for each of the possible classifications based on the values of the attributes for this instance [9]. One way for deciding which classification most likely belongs to the unseen instance, and what we will use in this experiment, is by choosing the classification with the highest probability: the winner takes all.

The Naïve Bayes classifier uses Bayes' theorem: if we use vector $(x_1 \dots x_n)$ as the values for the attributes for an unseen instance and C_i for a possible classification, the chance of $P(C_i \mid x_1 \dots x_n)$ is equal to $P(x_1 \dots x_n \mid C_i)P(C_i) / P(x_1 \dots x_n)$. This chance is called the posterior probability and can be calculated for all possible classifications. We can leave out the division by $P(x_1 \dots x_n)$, since for every possible classification C_i, $P(x_1 \dots x_n)$ will have the same value: it is a constant that can be used if you want to normalize the probabilities. This leaves us with the calculation of $P(x_1 \dots x_n \mid C_i)P(C_i)$. When a Naïve Bayes classifier is used, the number of instances in the dataset is normally not enough to be able to calculate $P(x_1 \dots x_n \mid C_i)$. To be able to calculate $P(x_1 \dots x_n \mid C_i)$, you need every possible combination of values for the attributes. Even if you have all the possible combinations, it takes a lot of time to calculate. That is why the classifier has the naïve assumption, hence the name Naïve Bayes, that all the attributes are independent given the class value [9]. But even if there are dependent attributes, the Naïve Bayes classifier has a high accuracy [10]. If you take the naïve assumption into account, it is possible to substitute $P(x_1 \dots x_n \mid C_i)$ [9]:

$$P(x_1 \dots x_n \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

The $P(x_k \mid C_i)$ can be easily calculated from the dataset: the total number of times value x_k for the corresponding attribute occurs in all the instances with classification C_i divided by the total number of instances with classification C_i. To make the formula complete, we still have to add the $P(C_i)$: the total number of instances with classification C_i divided by the total

number of instances. Due to this naïve assumption, the formula without normalizing can be calculated easily [9]:

$$\mathrm{P}(\mathrm{C_i} \mid \mathrm{x_1} \ \ldots \ \mathrm{x_n}) \propto P(x_1 \ \ldots \ x_n \mid C_i)P(C_i) = P(C_i)\prod_{k=1}^{n} P(x_k \mid C_i)$$

Or with the constant to normalize:

$$\mathrm{P}(\mathrm{C_i} \mid \mathrm{x_1} \ \ldots \ \mathrm{x_n}) = \frac{P(x_1 \ \ldots \ x_n \mid C_i)P(C_i)}{P(x_1 \ \ldots \ x_n)} = \frac{1}{P(x_1 \ \ldots \ x_n)} P(C_i)\prod_{k=1}^{n} P(x_k \mid C_i)$$

An example: with the given instances found in Table 1, we will calculate the posterior probability of both the possible classifications for a new instance. This new instance has the value “sour” for attribute Taste and “false” for attribute colorRed.

*P(apple|taste=sour, colorRed = false) ∝ P(apple) * P(taste=sour|apple) * P(colorRed=false|apple) = 1/3 * 1/3 * 1/3 = 0.03704*

*P(orange|taste=sour, colorRed = false) ∝ P(orange) * P(taste=sour|orange) * P(colorRed=false|orange) = 1/3 * 2/3 * 2/3 = 0.14815*

Taste	ColorRed	Fruit
sour	true	apple
sour	false	orange
sweet	false	apple
sweet	true	apple
sour	true	orange
sweet	false	orange

Table 1: Example dataset

For this example, the classification “orange” is chosen for the new instance. As we did not normalize, the values do not add up to 1. But, because we choose the classification with the highest posterior probability, this does not matter. With normalization, the posterior probability of “orange” will still be higher than the posterior probability of “apple”, as both posterior probabilities will be multiplied by the same constant.

The formula that assumes that the variables are independent works with categorical attributes, but continuous attributes are also often found in datasets. As the number of possible values for a continuous attribute is infinite, it is possible that the posterior probabilities of all the classifications is zero, since it is very likely that none of the instances in the training set share the same continuous value as the unseen instance. We need another formula for the calculation of $P(x_k \mid C_i)$ when using continuous attributes. The Naïve Bayes classifier assumes that a continuous attribute has a normal distribution. Because a normally distributed continuous attribute has a mean, μ, and the variance, σ, the following formula can be used [11]:

$$P(x_k \mid C_i) \ = \ \frac{1}{\sqrt{2\pi\sigma_{C_i}^2}} exp - \frac{\left(x_k - \mu_{C_i}\right)^2}{2\sigma_{C_i}^2}$$

2.2.1 Accuracy

How can the accuracy of a classifier on a dataset be calculated? We are using supervised learning, so for all the instances in a dataset, the correct classification is given. We split the dataset in two parts: a training set and a test set. The instances in Table 1 can be seen as a training set. Based on the values of those instances, the classification of an unseen instance is predicted. The unseen instances can be seen as the test set. They do not share their attribute values during the training of the classifier. Since for all the instances in the dataset the correct

classification is given, and the test set is a part of the dataset, all the instances in the test set also have a correct classification. The accuracy of the classifier can now be calculated by comparing the predicted classification and the correct classification of all the instances in the test set.

2.2.1 The Naïve Bayes classifier in our experiment

We now know how the Naïve Bayes classifier works and how it handles categorical and continuous attributes. In our experiment, we used package "caret" in R as our Naïve Bayes classifier [12]. The code we used can be found in detail in Appendix A. First of all, we had to train the Naïve Bayes classifier using the training set. Function *train(x, y, method="nb")* returns the model on which we can use the test set. Parameter *x* represents the attributes of the training set and parameter *y* represents the vector with the classifications of the training set. As method, we gave *"nb"* which stands for Naïve Bayes. When using this function, tuning parameters *fL*, *adjust* and *useKernel* can be used [13]. In this experiment, the tuning parameters *fL* and *adjust* had default value 0 and parameter *useKernel* had default value false: we did not use any tuning.

Using the function *predict(model, x)* with parameter *model* as the result of the train function and parameter *x* as all the attributes of the test set, a vector with the predicted classifications is returned. It is also possible to return the posterior probability for every classification per instance when parameter *type="prob"* is used instead of the default *type="raw*. The predicted classification is always the one with the highest posterior probability, as seen when comparing the results of using *type="prob"* and *type="raw"* (see Appendix B). In this experiment, the default *type="raw"* is used.

To get a nice overview of the predictions and the actual classifications of the test set, method *table(pred, y)* is used. Parameter *pred* is the vector with predicted classifications returned from the predict function and *y* is the vector with the actual classifications of the test set. The accuracy, the total number of correctly classified instances in the test set, can be calculated from this table. In the table, the top row stands for the actual classifications of the instances and the left column for the predicted classifications of the instances by the Naïve Bayes classifier. The numbers in the diagonal from the top left till the bottom right are the correctly classified instances. The other numbers are the wrongly classified instances. For example, in Table 2, one instance was predicted to have a classification of "3", but its actual classification is "1". The accuracy in this case is the total number of correctly predicted instances divided by the total number of instances times 100%: (11 + 17 + 11) / (11 + 17 + 11 + 2 + 1) * 100% = 92.86%.

	actual		
predicted	*1*	*2*	*3*
1	***11***	*2*	*0*
2	*0*	***17***	*0*
3	*1*	*0*	***11***

Table 2: Example of a table returned by method table(pred,y); bold numbers are the correctly predicted instances

2.3 Normally Distributed

Our experiment focuses on normally distributed continuous attributes. So, before we can use a continuous attribute of a dataset, we have to verify that the attribute has a normal distribution. What do we mean exactly by "an attribute has a normal distribution"? For all possible classifications, $C_1 \ldots C_n$, in the dataset, continuous attribute X has a normal distribution if and only if $P(X|Ci)$ has a normal distribution with $i = \{1, \ldots, n\}$. For example, if we want to check if attribute weight has a normal distribution and the dataset of which attribute weight is part has possible classifications "male" and "female", than attribute weight is normally distributed if all the instances with classification "male" as well as all the instances with classification "female" have a normal distribution for attribute weight.

We can test an attribute with graphical methods, such as boxplots, histograms and QQ plots, to see if the attributes has a normal distribtion. Those graphical methods make it easier for someone to get an idea of the distribution at a glance, since most people understand a visual representation faster than an array of numbers. But even if you conclude with the help of those methods that an attribute has a normal distribution, this does not have to be correct [14]. To get a more reliable answer, normality tests are often used. There are multiple tests available, such as the Jarque-Bera test and the Lilliefors test, but studies have shown that the Shapiro-Wilk test is the most reliable [15,16]. But, the Shapiro-Wilk test is only reliable to use if the sample size is bigger than thirty [16]. So, to prevent attributes classified as having a normal distribution even if they have not, we will only use the datasets in which all the classifications have more than thirty instances.

2.3.1 Shapiro-Wilk

The Shapiro-Wilk test will be used on one attribute at the time and separately for all possible classifications. For example, if we want to test if the loaves of bread from the example in section 2.1 have a normal distribution for the continuous attribute weight, we will first use the Shapiro-Wilk test on attribute weight for all the loaves of recipe one and then on all the loaves of recipe two. We will only use attribute weight for our Naïve Bayes classifier if both tests succeed and there are at least thirty loaves of bread per recipe.

There are two hypotheses. The null hypothesis assumes that the continuous attribute forms a normal distribution and the alternative hypothesis the opposite: the attribute is not normally distributed. The test returns a probability value, a p-value, which we can use to decide if we have to reject or cannot reject the null hypothesis. If the p-value is smaller or equal to 0.05, we can assume that the attribute is not normally distributed as there is a only very small chance that it is: we have to reject the null hypothesis. The null hypothesis does not have to be rejected if the p-value is bigger than 0.05, as you cannot claim that the attribute is not normally distributed [17]. We will run this test on all the attributes in the dataset for every classification individually and only keep the attributes of which, it cannot be claimed, to not be normally distributed. An example of how we used the Shapiro-Wilk test has been given in Appendix A.

2.4 Discretization

The goal is to use the Naïve Bayes classifier on normally distributed datasets to see if the continuous attributes themselves or the categorical version of those attributes will be

classified wrong less often. To get those categorical values, we will have to find a way to divide the continuous attributes in multiple categories: we have to discretize the attributes. There are multiple methods to achieve our goal. We can use a supervised technique like the minimum-entropy-based discretization by Fayyad and Irani or an unsupervised technique like equal frequency or equal width. Six discretization methods for the Naïve Bayes classifier have been compared in terms of mean error (divided into bias and variance) and accuracy. Equal frequency with 10 bins has the highest average accuracy and the lowest mean error [18]. Due to this conclusion, the unsupervised method equal frequency with ten bins will be used in this experiment. A bin can be seen as an interval. Every instance, of which the value of the attribute that is being discretized falls in that interval, should get the category of that bin as a value for the newly discretized attribute. The first bin has an interval that starts with minus infinity and the last bin has an interval that ends with infinity. That equal frequency has the highest average accuracy and the lowest mean error, does not necessarily mean that this unsupervised method will give the highest accuracy in comparison with other discretization methods.

Two unsupervised methods, equal width and equal frequency, and one supervised method, entropy-based discretization, were tested on the same eight datasets. Both the unsupervised method and the supervised method had a better accuracy half the time: four times each [19]. Since this is the case, we will use a supervised and an unsupervised discretization method, instead of only unsupervised method equal frequency. This is to prevent having an incomplete conclusion. After all, by chance it may happen that we will pick datasets that would have higher accuracies if using a supervised method. Then, if only equal frequency has been used, the difference between the accuracy of the continuous and the discretized version will not be conclusive: the continuous attributes may give a better accuracy, but maybe if a supervised method would have been used, the discretized attributes could be more accurate. Supervised methods take, in contrast to unsupervised methods, the classifications of the instances in consideration when discretizing. Using this extra information, the supervised methods try to get the bins that provide the highest accuracy of classifying unseen instances.

In this experiment, the supervised method to calculate the boundaries of the bins will be by using the minimum description length principle. As more bins increases the chance of overfitting and decreases generalization [20], this supervised method is extra useful next to equal frequency. The minimum description length principle calculates the simplest set of cut points with the best accuracy. Generally, this is less than the ten bins we use for the equal frequency discretization. When a simple model is used, there is less chance of overfitting on the noise in data [21]. As we use two different discretization methods that complement each other (supervised vs unsupervised and difference in over- and underfitting), we will get a more reliable answer to our question what the difference is between Naïve Bayes on normally distributed continuous attributes and the discretized version of those attributes: for some datasets unsupervised methods work better than supervised methods [19]. To prevent the possible overfitting of equal frequency, we could use five bins instead of ten. But looking at the variety of sizes, small and big, of the datasets used in [18] and the results of using ten bins in compared to the five bins equal frequency, ten bins is still preferable to fewer bins.

Before the discretization can take place, the dataset has to be split into a training set and a test set. The training set will contain 80% of the available instances and the test set the other 20%. The reason for splitting before discretizing is because you may not use any data from the test set to influence the training of the Naïve Bayes classifier. Because the dataset is split before discretization and the Naïve Bayes classifier only trains on the training set, the test set has no influence on the results.

As stated previously, the first bin has an interval that starts with minus infinity and the last bin ends with infinity. The use of infinities is necessary due to the instances in the test set. Those instances will be using the same bins as the training set. If the minus infinity and infinity were not being used, it could happen that a value of a test set instance does not belong to any bin (the value could be smaller than the smallest value in the training set) and we may not change the bin boundaries according to the smallest and largest value of the whole dataset (only the training set). This leaves us with using the infinities.

2.4.1 Equal frequency

To get the ten bins we want for equal frequency, we need to know nine more values since we already know a boundary of the first and last bin (the infinities). To get the other nine boundaries, the instances in the training set should be ordered from small to large according to the value of the attribute that is being discretized. Then, a formula can be used to get the sought values. For $i = 0.1, 0.2 \ldots 0.9$, the value of the attribute at place "*total number of instances in the training set * i"* should become a boundary. If the total number of instances is not easily divided by ten, the *"total number of instances in the training set * i"* should be rounded up. This ensures the best distribution of instances in all the bins. Now the eleven boundaries are known and can be used to set the intervals of the ten bins.

2.4.2 Minimum description length principle

The minimum description length principle (MDLP) assumes that the best, most compact representation of data best represents the data [22]. We will use the function *mdlp()* of the "discretization" package in R. This function uses the entropy criterion and has the MDLP as the stopping rule [23]. With a training set with less than 50 instances, the result can be that the MLDP does not return any boundaries, since it has a problem with finding enough useful boundaries due to the small training set size [24]. That is why we will make sure that the training sets have at least 50% more instances: more than 75.

Entropy-based discretization works the following way: using a formula, the optimal boundary is calculated by picking the value of an attribute with the highest gain. Now, the data is split in two on that value. For the smaller sets the new value with the highest gain can be calculated again and the corresponding set can be split again. This goes on and on until a stop criterium is reached. The gain is calculated with the formula $Gain(A, T; S) = Ent(S) - E(A, T; S)$. S is the set we want to split and A is the attribute which we want to split on the value T. Ent(S) is the class entropy of the set and $E(A, T; S)$ is the weighted average of the entropies of the two sets S splits in when the split is made on value T: sets S_1 and S_2. Set S_1 has all the instances with a value smaller or equal to T for attribute A and set S_2 has all the other instances [25].

$$Ent(S) = -\sum_{i=1}^{k} P(C_i, S) log_2(P(C_i, S))$$

$$E(A, T; S) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2)$$

Now we know how the optimal boundaries are calculated using entropy. What is the stop criterium exactly? When using MDLP as stop criterium, a split on value T for attribute A is accepted for set S if the following formula is true, otherwise there will not be a split on value T. N is the number of instances in set S and k_i the number of classifications in the S_i [25].

$$Gain(A, T; S) > \frac{log_2(N-1)}{N} + \frac{\Delta(A, T; S)}{N}$$

$$\Delta(A, T; S) = log_2(3^k - 2) - [kEnt(S) - k_1 Ent(S_1) - k_2 Ent(S_2)]$$

The worst case scenario is only having one bin as a result: minus infinity through to infinity. This scenario comes up when there is never a gain high enough so a split can take place. That is why there were datasets with zero bins in [24].

2.5 Validation

The results will consist of the percentage of wrongly classified unseen instances. To get those percentages, we are going to use Monte-Carlo cross validation:

The instances of a dataset are randomly shuffled and divided into two separated groups: the training data (the first 80%) and the test data (the last 20%). The training and test set both contain random instances, since we randomly shuffled before we split. Every instance occurs only once in either the training set or the test set. This specific partition is used three times: once to train the Naïve Bayes on the continuous attributes, once to train the Naïve Bayes on the discretized version of those continuous attributes using equal frequency and once on the discretized version of those continuous attributes using the MLDP. Of the three training sets, two sets, and their corresponding test sets, are being discretized with the two discretization methods. After training the three Naïve Bayes classifiers on the training datasets to get the models, the models are used on the corresponding test datasets and the results, the percentage of wrongly evaluated instances, is calculated.

This algorithm, from the shuffling and splitting of the dataset to the calculations of the percentage of wrongly evaluated instances, is done 25 times. That means we have 25 percentages for all three tested attributes (continuous, discretized by equal frequency and discretized by MLDP): for every run one. Those percentages are then returned in a table so the three means and three standard deviations of the mean can be calculated. All of this was done in R. The code of the Monte-Carlo cross validation, including the discretization of the datasets, can be found in Appendix A.

The reason for using the percentage of wrongly classified unseen instances instead of the generally used accuracy, the percentage of correctly classified unseen instances, is because the goal of this experiment is not to make a classifier that is as good as possible. The accuracy will not be as high as could be achieved when using all attributes of a dataset instead of only the normally distributed continuous attributes. The goal is to check whether it is better to

discretize or to keep normally distributed continuous attributes. The one that has the smallest percentage of wrongly classified unseen instances is in that case better, if the true difference between means is significant.

3. Experiments

3.1 Used Data Sets

In the past sections, the methods this experiment will use have been explained. But before the actual experiment can take place, the datasets are needed. Four datasets have been picked carefully, since those datasets had normally distributed continuous attributes. It seems like we are doing feature selection, but this is not the case. We do not want to make a classifier that is as good as possible. We want to have a clear picture of the difference between normally distributed continuous attributes and the discretized version of those. The best way to make sure that no other attributes influence the results, is to only use the normally distributed continuous attributes to train the Naïve Bayes classifier. None of the datasets used in this experiment have missing values. Each dataset will be explained briefly and then some information about the artificial datasets will be given. The attribute numbers emphasized between brackets represent the attribute number in the data files found on the UCI repository [26].

Planning Relax

"The dataset concerns with the classification of two mental stages from recorded EEG signals: Planning (during imagination of motor act) and Relax state." – [27]

In this dataset, a total of 13 attributes are available: 12 continuous and 1 for the classification. Because we want to look at normally distributed data, all the attributes have been tested with the Shapiro-Wilk test. Only six continuous attributes passed the test (2, 3, 4, 5, 6 and 8). The 182 instances are divided into two classes: 130 are classified as "1" and the other 52 as "2". This split is unbalanced and it will happen that an unseen instance will be classified as "1" more often. The more data available of class value "1" in comparison with class value "2" leads to having a higher probability of having class value "1" in general. It is interesting to see what the difference in results is if the continuous attributes of this dataset are normally distributed.

Kernels

"Measurements of geometrical properties of kernels belonging to three different varieties of wheat."- [28]

This dataset only has four attributes that passed all the criteria.: the perimeter of the wheat kernel (2), the compactness (3), the length of the kernel (4) and the length of the kernel groove (7). For each of the three different varieties of wheat, Kama, Rosa and Canadian, seventy randomly selected kernels were chosen: a total amount of 210 instances.

Wine

"These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars." – [26]

The 178 instances consist of three types of wines: 59 of type "1", 71 of type "2 " and 48 of type "3". Even though there are only 48 wines with classification "3", this is still more than the

thirty instances required per class to be able to use the Shapiro-Wilk test. The four normal distributed attributes indicate the quantity of the following constituents: alcohol (2), ash (4), alkalinity of ash (5) and OD280/OD315 of diluted wines (13).

Wimbledon
This dataset contains the match statistics for the men at the Wimbledon tennis tournament of the year 2013. – [26]

For this dataset the classifier tries to determine if player one or player two has won. The classifications are "0" if player one has lost and "1" if he managed to win. Of the 114 matches, player one lost 59 times and won 55 times. The attributes mean the following: first serve percentage for player one (7), second serve percentage for player one (9), winners earned by player one (13), first serve percentage for player two (25) and second serve percentage for player two (27).

Artificial datasets
Not only those four datasets have been used in this paper. In real life, it is possible that there is more data available than the hundred or two hundred instances as in the datasets above. To verify that the end results will remain the same when the number of instances increase, artificial datasets have been generated. Using the knowledge that the attributes are normally distributed, the mean and standard deviation per attribute per class of the datasets we found on the UCI repository (overview in Appendix C) can be calculated. Based on those values, as many instances as wanted can be made for a new dataset. Keeping the same ratio of instances with different classifications as in the original dataset, all four datasets have been made twenty times bigger. The twenty times as big newly generated datasets have been tested the same way as the original datasets: tested with the Shapiro-Wilk test to verify that the continuous attributes are normally distributed, discretized with the two methods and validated with Monte-Carlo cross validation.

Next to those four artificial datasets, two extra datasets have been made. One dataset with random means and standard deviation for five attributes and two classifications: class "1" has 100 instances and class "2" has 150 instances. This represents a random possible real life dataset. The second has the same means and standard deviations but there are twenty times more instances: 2000 for class "1" and 3000 for "2". We have given this dataset an unbalanced number of instances per classification as a real life dataset is not always balanced and we want to simulate that.

3.2 Experimental set-up

Now that it has been decided which methods and datasets to use, the actual experiment can begin. First of all, the chosen datasets have at least one continuous attribute. All of the continuous attributes in the chosen datasets were tested with the Shapiro-Wilk test to see if the continuous attributes had a normal distribution. Of all the attributes in the dataset, only the normally distributed continuous attributes will be used in this test. The p-values, the result of the Shapiro-Wilk test, for all the continuous attributes used in this experiment, are written down in Appendix D. Then, per dataset, the Monte-Carlo cross validation method was used to get a table with 25 rows: one for each run. Every row contains the three percentages of wrongly classified unseen instances: one for the continuous attributes and one for each of the

two discretized attributes. Those tables can be found in Appendix E. After the Monte-Carlo cross validation was finished, the means and the corresponding standard deviation of each column in the table were calculated.

3.3 Results & discussion

Dataset	# of attributes	# of class values	# of instances	Continuous		Equal Frequency		MDLP	
				Mean	SD	Mean	SD	Mean	SD
Planning Relax	6	2	182	29%	7.82	38.78%	6.39	-	-
Planning Relax (x20)	6	2	3640	28.46%	0.85	28.69%	0.95	-	-
Kernels	4	3	210	9.52%	3.37	10.1%	3.91	11.05%	3.82
Kernels (x20)	4	3	4200	3.50%	0.54	3.94%	0.54	3.97%	0.64
Wine	4	3	178	8.22%	5.04	9.44%	5.13	7.67%	6.02
Wine (x20)	4	3	3560	5.65%	0.85	6.69%	0.69	6.69%	0.83
Wimbledon	5	2	114	44.35%	8.87	43.83%	8.51	-	-
Wimbledon (x20)	5	2	2280	31.41%	2.03	32.04%	2.18	-	-
Artificial set	5	2	250	5.84%	3.05	6.64%	3.30	8.40%	3.74
Artificial set (x20)	5	2	5000	6.69%	0.60	7.48%	0.73	7.45%	0.69

Table 3: Percentage wrongly classified unseen instances of using Naïve Bayes with two discretization methods

Using only the normally distributed attributes of the datasets as attributes for the Naïve Bayes classifier, the percentage of wrongly classified unseen instances has been calculated. In Table 3 the results are shown, including the standard deviation of the mean, for the three tests: Naïve Bayes using the continuous values themselves and using the values discretized with equal frequency and with the MDLP method. The hyphens under MDLP mean that one (or more) of the attributes of the dataset only has one bin. For every dataset, the number of class values and instances are given. Also, how many attributes of the dataset we used: the number of normally distributed continuous variables in the dataset. The results show that whenever an attribute in a dataset only gets one bin using MDLP, the dataset that is twenty times bigger also only gets one bin. For those datasets, there is never a gain high enough so a split will take place.

In eight of the ten datasets, using the continuous attributes results in the lowest percentage of wrongly classified instances. But just looking at the different means and standard deviation does not provide an accurate conclusion. Using the paired t-test, we can calculate if the differences in mean are significance. If the p-value is bigger than our alpha, 0.05, we will accept our null hypothesis: the means are equal. If the p-value is smaller than our alpha, we will reject our null hypothesis: the means are not equal. We used the t.test function in R to calculate the t-values. The t.test function expects a column with the percentages per run, instead of the mean we calculated for Table 3. Those percentages per run are available in Appendix E. The results of the paired t-tests are shown in Table 4. An example of how we used the t.test function is given in Appendix A.

Dataset	p-value between continuous and equal frequency	p-value between continuous and MLDP	p-value between equal frequency and MLDP
Planning Relax	**1.009e-05**	-	-
Planning Relax (x20)	**0.003938**	-	-
Kernels	0.4775	0.07261	0.2178
Kernels (x20)	**3.889e-06**	**5.006e-05**	0.7934
Wine	0.1487	0.5934	0.126
Wine (x20)	**1.302e-09**	**9.128e-07**	0.9956
Wimbledon	0.8115	-	-
Wimbledon (x20)	**0.005419**	-	-
Artificial set	0.1059	**0.0008102**	**0.02775**
Artificial set (x20)	**2.18e-11**	**1.969e-07**	0.7936

Table 4: t-values between the means of the percentages of the different trained Naïve Bayes classifier; bold numbers indicate significant differences

For all the datasets that are twenty times larger, there is a true difference between the mean when using the continuous attributes and the mean when using the attributes discretized with equal frequency or MLDP. For all of those datasets, the difference between using the attributes discretized with equal frequency and the attributes discretized with MLDP is not significant. If you compare the percentages of wrongly classified unseen instances of those datasets, you can see that for all those datasets, the datasets that are twenty times larger, the mean when using the continuous attributes is lower than the means of the other two. A lower mean means less wrongly classified unseen instances.

For the original datasets Kernels, Wine and Wimbledon, the difference between the mean when using the continuous attributes and the means when using the discretized attributes is not significant. It does not matter which discretization method is used, as the difference between their means is not significant either.

The mean when using continuous attributes and the mean when using the attributes that are discretized using equal frequency of the small artificial dataset are not significantly different either. But the other two paired t-tests, between continuous attributes and the attributes discretized with MDLP, and between the two discretization methods, show a significant difference. The means of the continuous attributes and the attributes discretized with equal frequency are lower than the mean when using the attributes using MLDP.

The original dataset Planning Relax with the unbalanced split of number of instances per different classification is an exception for the smaller datasets. There is a significant difference between using the continuous attributes themselves and using the discretized attributes using equal frequency.

4. Conclusion & Future Research

Our experiment gives an answer to our question: if we verify that the attributes of a dataset are normally distributed, which will give a higher accuracy using Naïve Bayes: the continuous attributes themselves or the discretized version of those? Using the paired t-test we have investigated if the difference of the means is significant. We have smaller datasets, between 114 and 250 instances, and larger datasets, between 2280 and 5000 instances. For the larger datasets, using the continuous attributes themselves is in our experiment always better than using discretized attributes. The discretization method does not matter in this experiment as

the difference between the means of the supervised and unsupervised method is not significant. So, if we verify that the attributes of a dataset are normally distributed and there are a large number of instances, it is better to use the continuous attributes themselves.

For the smaller datasets, the answer to our question will be different. For most datasets, it does not matter if we use the continuous attributes themselves or if we use the discretized attributes as the difference between their means is not significant. Only if the split of the number of instances per class value is unbalanced, is it better to use the continuous attributes themselves. The ratio of the split for this dataset is 2.5:1: for every instance with classification "2", there are 2.5 instances with classification "1". That it is better to use the continuous attributes themselves is probably because, for the discretized values, a lot of bins have more instances of classification "1" than of classification "2". The chance that an unseen instance has classification "1", is higher in that case. For the continuous attributes, the probability of a classification is calculated with a formula in which the instances of the other classifications are not included. The formula could result in a high probability for classification "2" even if a lot of instances of classification "1" have around the same values for the attributes as the unseen instance. In a future experiment, different ratios can be tried to find the ratio for which the difference in means are not significant anymore for smaller datasets.

The answers above are useful to know for future classifications. As [3] has concluded that discretization of continuous attributes provides the best accuracy of unseen instances and we have just concluded that it is best for the normally distributed continuous attributes to keep them as they are for large datasets, there is a possibility that a combined version of those conclusions is even better to use. A new experiment that discretizes all continuous attributes, except for the normally distributed continuous attributes, will presumably get even higher accuracies! If that is the case, companies can adjust their current classifications and get better results. For example the bank that wants to know when to give a loan and when the risk is too high. If the new experiment shows that the combination of our experiment and the experiment in [3] improves the classifications, the bank will lose even less money. Mixing continuous attributes and categorical attributes is possible for the Naïve Bayes classifier if you extend it to handle both [29]. This is only useful for large datasets as the smaller datasets can just be discretized along with the not normally distributed continuous attributes. On the other hand, small datasets with an unbalanced split for number of instances per classification can maybe profit when, like large datasets, using the extended Naïve Bayes classifier so that the normally distributed continuous attributes do not have to be discretized. Once again, there should be an experiment first to see for which ratios this is true or if maybe it is better to replace the winner takes all rule with a threshold.

In our experiment we only used one unsupervised and one supervised method to discretize. Those methods were selected very carefully using the results of other papers, but those papers have not verified that they were using normally distributed continuous attributes. It is possible that other discretization methods work more accurately with verified normally distributed values. In another experiment this could be investigated. We do have seen that the two discretization methods do not have a significant difference in the means. Only once, using equal frequency gave less wrongly classified unseen instances. If you have to choose

between using equal frequency and MLDP for discretization, it could be better to use equal frequency when dealing with normally distributed continuous attributes.

References

[1] Hellerstein, J. L., Jayram, T. S., & Rish, I. (2000). Recognizing End-User Transactions in Performance Management. Austin, Texas: *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence.*

[2] Jin, R., Breitbart, Y., & Muoh, C. (2007). Data Discretization Unification. *Seventh IEEE International Conference on Data Mining (ICDM 2007).*

[3] Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. *Proceedings of the Twelfth International Conference on Machine Learning,* 194-202.

[4] Hildebrand, A.J. (2008, April 9) The Central Limit Theorem. Retrieved from http://www.math.uiuc.edu/~ajh/408/408clt.pdf

[5] Mlodinow, L. (2009). The Drunkards walk: how randomness rules our lives. New York: Vintage Books.

[6] SOCR Data Dinov 020108 HeightsWeights. (2008, February 1). Retrieved from http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights

[7] Goldstein, H. (1971). Factors influencing the height of seven year old children—results from the national child development study. *Human Biology, 43*(1), 92-111.

[8] Mahumud, R.A., Sultana, M. & Sarker, A.R. (2017). Distribution and Determinants of Low Birth Weight in Developing Counties. *Journal of Preventive Medicine & Public Health*, *50*(1), 18-28.

[9] Leung, K. M. (2007, November 28). Naive Bayesian Classifier. Retrieved from http://cis.poly.edu/~mleung/FRE7851/f07/naiveBayesianClassifier.pdf

[10] Domingos, P., & Pazzani, M. (1997). On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, *29*, 103-130.

[11] John, G.H., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. San Mateo, California: *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*.

[12] Kuhn, M. (2017, April 18). Package "caret". Retrieved from https://cran.r-project.org/web/packages/caret/caret.pdf

[13] Kuhn, M. (2016, November 29). The caret Package. Retrieved from http://topepo.github.io/caret/available-models.html

[14] Ghasemi, A., & Zahediasl, S. (2012). Normality tests for statistical analysis: a guide for non-statisticians. *International Journal of Endocrinology and Metabolism, 10*(2), 486-489

[15] Mendes, M., & Pala, A. (2003). Type I error rate and power of three normality tests*. Pakistan Journal of Information and Technology, 2*(2), 135-139.

[16] Razali, N. M., & Wah, Y. B. (2011). Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests . *Journal of Statistical Modeling and Analytics, 2*(1), 21-33.

[17] Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika, 52*(3-4), 591-611.

[18] Flores, M. J., Gámez, J. A., Martínez, A. M., & Puerta, J. M. (2011). Handling numeric attributes when comparing Bayesian network classifiers: does the discretization method matter? *Applied Intelligence, 34*(3), 372-385.

[19] Ibrahim, M. H., & Hacibeyoğlu, M. (2016). Comparison of the effect of unsupervised and supervised discretization methods on classification process. *International Journal of Intelligent Systems and Applications in Engineering, 4*(Special Issue), 105-108

[20] Bosman, P.A.N. & Thierens, D. (2006). Numerical optimization with real-values estimation-of-distribution algorithms. *Studies in Computational Intelligence*, 33, 91-120

[21] Grünwlad, P.D. (2007). The minimum description length principle. Cambridge, Massachusetts: The MIT Press.

[22] Rissanen, J. (1978). Modeling by shortest data description. *Automatica, 14*(5), 465-471

[23] Kim, H. (2015, February 19). Package 'discretization'. Retrieved from https://cran.r-project.org/web/packages/discretization/discretization.pdf

[24] An, A. & Cercone, N. (1999). Discretization of continuous attributes for learning classification rules. *PAKDD 1999: Methodologies for Knowledge Discovery and Data Mining*, 509-514

[25] Fayyad, U. M. & Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning*, Proceedings of the International Joint Conference on Uncertainty in AI*, 1022-1027

[26] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science

[27] Rajen Bhatt, 'Planning-Relax Dataset for Automatic Classification of EEG Signals', UCI Machine Learning Repository

[28] Placed on the UCI Repository [26] by the Institute of Agrophysics of the Polish Academy of Sciences in Lublin.

[29] Hsu, C., Huang, Y. & Chang, K. (2008). Extended Naïve Bayes classifier for mixed data. *Expert Systems with Applications, 35*(3), 1080-1083

Appendix A: R-code

In this section we will go deeper into some used code with some explanation of how we interpreted the results. The libraries we have used are: “gridExtra”, “dplyr”, “ggpubr”, “caret”, “klaR” and “discretization”.

Monte-Carlo cross validation (including discretization)

We will use the wine dataset as an example. First of all, we need a place to store the results for every run and import the data:

```
mydata = read.csv("bankk.txt", header = FALSE)
mydata = mydata[c("V2","V4","V5","V13","V1")]
colnames(mydata) = c("x1", "x2", "x3", "x4", "y")
mydata["y"] <- lapply(mydata["y"], factor)
results <- matrix(ncol=3, nrow=25)
```

Then, we have a loop that runs over the following code 25 times, and adds the results to the matrix above. First of all, we have to make the new attributes so we are able to save the discretized values somewhere:

```
mydata = mydata[sample(nrow(mydata)),] #shuffle data

divider = round(nrow(mydata)*0.2) #20% testdata, 80% trainingdata

#Continuous attribute
test = mydata[0:divider,]
training = mydata[-(0:divider),]

#Discretized attribute - equal frequency
test2 = mydata[0:divider,]
training2 = mydata[-(0:divider),]

#Discretized attribute - MLDP
test3 = mydata[0:divider,]
training3 = mydata[-(0:divider),]
```

For equal frequency we have to calculate the ten bin boundaries and give the right category per instance in the training and test set. The following code should be executed for every attribute:

```
number = nrow(training2)

ordered = training2[order(training2$x1),]
c1 = -Inf
c2 = ordered[ceiling(number * 0.1),]$x1
c3 = ordered[ceiling(number * 0.2),]$x1
c4 = ordered[ceiling(number * 0.3),]$x1
c5 = ordered[ceiling(number * 0.4),]$x1
c6 = ordered[ceiling(number * 0.5),]$x1
c7 = ordered[ceiling(number * 0.6),]$x1
c8 = ordered[ceiling(number * 0.7),]$x1
c9 = ordered[ceiling(number * 0.8),]$x1
c10 = ordered[ceiling(number * 0.9),]$x1
c11 = Inf
training2$x1<-cut(training2$x1,
c(c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11), right=FALSE, labels=c(1:10))
```

```
test2$x1<-cut(test2$x1, c(c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11),
right=FALSE, labels=c(1:10))

training2$x1 = as.factor(training2$x1)
test2$x1 = as.factor(test2$x1)
```

For MLDP, we have to calculate the boundaries of the bins. We use the function mdlp() from the library “discretization”. For all continuous attributes the following code should be executed:

```
training3 = mdlp(training3)
k = c(-Inf, training3$cutp[[1]], Inf)
test3$x1<-cut(test3$x1, k, right=FALSE, labels=(c(1:(length(k) -
1))))
```

Next, the Naïve Bayes classifier can be trained using the training sets. We want to have a table with the classifications of the unseen instances in the test set. That table can be used to calculate the percentage wrongly classified instances:

```
# Results Continuous and Discretized attributes
model = train(training[,1:4], training[,5], method="nb")
pred = predict(model, test[,1:4])
table1 = table(pred, test[,5], dnn=list('predicted', 'actual'))

model2 = train(training2[,1:4], training2[,5], method="nb")
pred2 = predict(model2, test2[,1:4])
table2 = table(pred2, test2[,5], dnn=list('predicted', 'actual'))

model3 = train(training3[,1:4], training3[,5], method="nb")
pred3 = predict(model3, test3[,1:4])
table3 = table(pred3, test3[,5], dnn=list('predicted', 'actual'))

# Put results in a matrix
totaldata =  nrow(test2)
results[i, 1]  = (totaldata - table1[1, "1"] - table1[2, "2"] -
table1[3, "3"]) / totaldata * 100
results[i, 2]  = (totaldata - table2[1, "1"] - table2[2, "2"] -
table2[3, "3"]) / totaldata * 100
results[i, 3]  = (totaldata - table3[1, "1"] - table3[2, "2"] -
table3[3, "3"]) / totaldata * 100
```

A table looks like this:

```
         actual
predicted  1  2  3
        1 12  0  0
        2  2 12  1
        3  0  2  7
```

When this has run 25 times (i = 1 … 25), the results should be printed like the tables in Appendix D:

```
#Print final results
dfResults = data.frame(results)

mean1 = mean(dfResults[["X1"]])
```

```
mean2 = mean(dfResults[["X2"]])
mean3 = mean(dfResults[["X3"]])
sd1= sd(dfResults[["X1"]], na.rm = FALSE)
sd2= sd(dfResults[["X2"]], na.rm = FALSE)
sd3= sd(dfResults[["X3"]], na.rm = FALSE)

dfResults = round(dfResults,2)
dfResults = cbind(Row.Names = rownames(dfResults), dfResults)
colnames(dfResults) =
c("Attempt","Continuous","EqualFrequency","MDLP")
dfResults$Continuous = paste(dfResults$Continuous, "%", sep="")
dfResults$EqualFrequency= paste(dfResults$EqualFrequency, "%",
sep="")
dfResults$MDLP = paste(dfResults$MDLP, "%", sep="")

png("wine_4attributes_table.png", height = 570, width = 300)
grid.table(dfResults, rows = NULL)
dev.off()
```

The mean and standard deviation of the mean have also been calculated so we were able to put it in the result section:

```
mean1 = round(mean1,2)
mean1 = paste(mean1, "%", sep="")
mean2 = round(mean2,2)
mean2 = paste(mean2, "%", sep="")
mean3 = round(mean3,2)
mean3 = paste(mean3, "%", sep="")
means = data.frame(mean1, mean2, mean3)
sd1 = round(sd1, 2)
sd2 = round(sd2, 2)
sd3 = round(sd3, 2)
sd1 = as.character(sd1)
sd2 = as.character(sd2)
sd3 = as.character(sd3)
sd = data.frame(sd1, sd2, sd3)
sd = cbind(" " = "Standard Deviation", sd)
means = cbind(" " = "Mean", means)
colnames(means) = c(" ", "Continuous","EqualFrequency","MDLP")
colnames(sd) = c(" ", "Continuous","EqualFrequency","MDLP")
means = rbind(means, sd)

png("150madexx_5attributes_means.png", height = 80, width = 360)
grid.table(means, rows = NULL)
dev.off()
```

Artificial data and the Shapiro-Wilk test

In the following code, it is shown how the artificial data is being generated in this experiment and the Shapiro-Wilk test is executed on the newly generated data. The Shapiro-Wilk test on the real life datasets is done in the same way. The newly generated data is generated based on the information in the following table:

Attribute	Class value	Sample Size	Mean	Standard deviation
X1	1	2000	6	6.25
	2	2000	3	3.6
X2	1	2000	1	0.75
	2	2000	2	0.25

```
#generate new attribute values
x11 = rnorm(2000,6,6.25)
shapiro.test(x11)
x12 = rnorm(2000,3.3,6)
shapiro.test(x12)
x1 = c(x11,x12)

x21 = rnorm(2000,1,0.75)
shapiro.test(x21)
x22 = rnorm(2000,2,0.25)
shapiro.test(x22)
x2 = c(x21,x22)

# combine the newly generated attribute values with the class value
("1" or "2") in a dataframe
y1 = rep(1,2000)
y2 = rep(2,2000)
y = c(y1,y2)
df = data.frame(x1, x2, y)
```

The Shapiro-Wilk test output will contain a p-value. This p-value will be used to determine if the attribute is normally distributed for that particular classification. For example, the `shapiro.test(x11)` returns a p-value of 0.5387. This is more than 0.05, so we can assume that the attribute is normally distributed for this classification:

```
  Shapiro-Wilk normality test

data:  x11
W = 0.99919, p-value = 0.5387
```

An example of an attribute that is nog normally distributed for a classification can be given with the line: `shapiro.test(runif(2000)).Runif(x)` generates x random values between 0 and 1. This vector has the same amount of instances as "x11", but the p-value is smaller than 2.2e-16, so we cannot say that the attribute is normally distributed in this case:

```
      Shapiro-Wilk normality test

data:  runif(2000)
W = 0.9541, p-value < 2.2e-16
```

t.test

Using the information from the tables in Appendix D, we can execute our paired t-test to see if the different means are significant. For the artificial dataset, the following code is used:

```
# artificial
i1 = c(2, 4, 2, 4, 12, 8, 8, 2, 4, 6,
      12, 6, 6, 6, 2, 4, 6, 10, 6, 4,
      8, 8, 2, 10, 4) #continuous
i2 = c(4, 6, 2, 4, 12, 10, 8, 2, 8, 4,
      14, 12, 8, 6, 6, 2, 6, 10, 4, 8,
      8, 6, 2, 6, 8) #equal frequency
i3 = c(2, 4, 4, 10, 10, 12, 16, 4, 10, 8,
      14, 10, 10, 4, 4, 10, 6, 12, 8, 8,
```

```
      6, 14, 12, 6, 6) #MLDP
t.test(i1,i2,paired=TRUE)
t.test(i1,i3,paired=TRUE)
t.test(i2,i3,paired=TRUE)
```

The results show the means of the normally distributed continuous attributes and of the discretized attributes using equal frequency are not significant different. The p-value of 0.1059 is bigger than our alpha of 0.05, what leads us to the acceptance of the null hypothesis: the true difference in means is equal. Two means that are not equal are the mean of the normally distributed continuous attributes and the discretized attributes using MLDP. That p-value of 0.0008102 is smaller than our alpha, so we have to reject the null hypothesis and accept the alternative hypothesis: the difference in means is significant enough. The results of the t.test function follow:

```
        Paired t-test

data:  i1 and i2
t = -1.6803, df = 24, p-value = 0.1059
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.7826123  0.1826123
sample estimates:
mean of the differences
                  -0.8


        Paired t-test

data:  i1 and i3
t = -3.8293, df = 24, p-value = 0.0008102
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.93978 -1.18022
sample estimates:
mean of the differences
                 -2.56
```

Appendix B: prediction

To find out what prediction the predict method returned, the following data has been used. On the left, the predict method has been used with parameter type=”prob”. This returns the posterior probability per class per instance in the test set. On that same test set, the predict method has been used again. This time, the parameter type had value “raw”. This returned a vector with all the predictions for every instance.

We have made the highest posterior probability bold for every instance. Comparing the bold probabilities, with the predicted classification, we came to the conclusion that every predicted classification was the classification with the highest posterior probability.

Predict(model, test attributes, type=”prob”)

	1	2
250	4.998912e-05	**9.999500e-01**
51	**8.229313e-01**	1.770687e-01
8	**9.686041e-01**	3.139591e-02
85	**9.999992e-01**	7.578852e-07
25	**6.451755e-01**	3.548245e-01
174	7.110436e-06	**9.999929e-01**
182	1.280581e-01	**8.719419e-01**
135	8.326408e-07	**9.999992e-01**
131	7.635828e-03	**9.923642e-01**
139	9.910268e-02	**9.008973e-01**
150	2.747544e-03	**9.972525e-01**
246	3.081964e-03	**9.969180e-01**
94	**9.112790e-01**	8.872102e-02
78	**9.648195e-01**	3.518052e-02
180	4.333802e-03	**9.956662e-01**
72	**7.883276e-01**	2.116724e-01
59	**9.485187e-01**	5.148130e-02
242	1.046420e-02	**9.895358e-01**
32	**9.922288e-01**	7.771175e-03
161	6.579077e-04	**9.993421e-01**
70	**9.997493e-01**	2.507043e-04
5	**9.982993e-01**	1.700690e-03
74	**9.897583e-01**	1.024168e-02
71	**9.999995e-01**	4.567310e-07
122	1.021879e-13	**1.000000e+00**
13	**8.279950e-01**	1.720050e-01
222	1.422757e-03	**9.985772e-01**
186	5.269493e-11	**1.000000e+00**
65	**9.951235e-01**	4.876550e-03
117	5.477719e-06	**9.999945e-01**
171	1.082382e-03	**9.989176e-01**
89	**9.999459e-01**	5.407778e-05
31	**8.617059e-01**	1.382941e-01
28	**9.957744e-01**	4.225559e-03
91	**9.994952e-01**	5.047785e-04
136	1.829980e-05	**9.999817e-01**

Predict(model, test attributes, type=”raw”)

2
1
1
1
1
2
2
2
2
2
2
2
1
1
2
1
1
2
1
2
1
1
1
2
1
2
2
1
2
2
1
1
1
1
2

7	**9.998079e-01**	1.920546e-04	1
3	**9.516224e-01**	4.837761e-02	1
175	2.122911e-06	**9.999979e-01**	2
214	1.490735e-12	**1.000000e+00**	2
118	4.930508e-01	**5.069492e-01**	2
197	4.587606e-03	**9.954124e-01**	2
185	5.992491e-05	**9.999401e-01**	2
169	3.664085e-04	**9.996336e-01**	2
49	**8.605017e-01**	1.394983e-01	1
37	**8.985189e-01**	1.014811e-01	1
63	**9.441253e-01**	5.587467e-02	1
245	5.641984e-07	**9.999994e-01**	2
109	1.234364e-11	**1.000000e+00**	2
240	1.137882e-01	**8.862118e-01**	2

Appendix C: means and standard deviations per attribute

In the following tables, one for each dataset, the mean and standard deviation per normally distributed attribute per class are shown. The attribute numbers represent the attribute number in the data files found on the UCI repository [26].

Planning Relax			
Attribute	Class value	Mean	Standard deviation
2	1	-0.00742	0.399
	2	0.00697	0.347
3	1	-0.00566	0.347
	2	-0.01317	0.337
4	1	-0.02563	0.406
	2	0.02153	0.434
5	1	0.01208	0.481
	2	-0.03700	0.468
6	1	-0.01426	0.418
	2	0.01808	0.371
8	1	0.00247	0.382
	2	0.00010	0.402

Kernels			
Attribute	Class value	Mean	Standard deviation
2	1	14.29	0.577
	2	16.14	0.617
	3	13.25	0.340
3	1	0.88	0.016
	2	0.88	0.016
	3	0.85	0.022
4	1	5.51	0.232
	2	6.15	0.268
	3	5.23	0.138
7	1	5.09	0.264
	2	6.02	0.254
	3	5.12	0.162

Wine			
Attribute	Class value	Mean	Standard deviation
2	1	13.74	0.462
	2	12.28	0.538
	3	13.15	0.530
4	1	2.46	0.227
	2	2.24	0.315
	3	2.44	0.185
5	1	17.04	2.546
	2	20.24	3.350
	3	21.42	2.258
13	1	3.16	0.357

	2	2.79	0.497
	3	1.68	0.272

Wimbledon			
Attribute	Class value	Mean	Standard deviation
7	0	62.69	6.50
	1	65.91	6.01
9	0	37.31	6.50
	1	34.09	6.01
13	0	36.69	14.84
	1	45.84	14.26
25	0	64.29	6.49
	1	63.6	6.22
27	0	35.71	6.49
	1	36.4	6.22

Artificial			
Attribute	Class value	Mean	Standard deviation
1	1	40	14
	2	35	7
2	1	6	1.2
	2	9	2.4
3	1	12	8
	2	8	4
4	1	14	0.5
	2	15	0.8
5	1	90	65
	2	36	52

Appendix D: Shapiro-Wilk test results

For every continuous attributes in the used dataset (including the artificial datasets) the Shapiro-Wilk test is tested to make sure the attributes are normally distributed. The p-values, the result of the Shapiro-Wilk test, are in the tables below per dataset per attribute per class value. The p-values of the attributes in the artificial datasets are the result of the attribute values we generated to have our final results. Based on those p-values we were able to conclude that the distribution is normal.

Planning Relax			
Attribute	Class value	Sample Size	p-value
2	1	130	0.3417
	2	52	0.7537
3	1	130	0.5276
	2	52	0.5632
4	1	130	0.4673
	2	52	0.8806
5	1	130	0.1735
	2	52	0.5318
6	1	130	0.4614
	2	52	0.8794
8	1	130	0.7704
	2	52	0.9336

Planning Relax (x20)			
Attribute	Class value	Sample Size	p-value
2	1	2600	0.3004
	2	1040	0.2218
3	1	2600	0.4988
	2	1040	0.5150
4	1	2600	0.3985
	2	1040	0.5408
5	1	2600	0.7369
	2	1040	0.2358
6	1	2600	0.7437
	2	1040	0.8250
8	1	2600	0.6740
	2	1040	0.3567

Kernels			
Attribute	Class value	Sample Size	p-value
2	1	70	0.5954
	2	70	0.2091
	3	70	0.7436
3	1	70	0.9937
	2	70	0.2251
	3	70	0.5397
4	1	70	0.5367
	2	70	0.521
	3	70	0.8631
7	1	70	0.6602
	2	70	0.1016
	3	70	0.612

Kernels (x20)			
Attribute	Class value	Sample Size	p-value
2	1	1400	0.2770
	2	1400	0.7834
	3	1400	0.8828
3	1	1400	0.8101
	2	1400	0.1183
	3	1400	0.2382
4	1	1400	0.8223
	2	1400	0.9588
	3	1400	0.3096
7	1	1400	0.7404
	2	1400	0.5175
	3	1400	0.3065

Wine			
Attribute	Class value	Sample Size	p-value
2	1	59	0.4791
	2	71	0.114
	3	48	0.6408
4	1	59	0.1556
	2	71	0.6198
	3	48	0.1092
5	1	59	0.2161
	2	71	0.07397
	3	48	0.09874
13	1	59	0.07745
	2	71	0.08904
	3	48	0.08311

Wine (x20)			
Attribute	Class value	Sample Size	p-value
2	1	1180	0.9571
	2	1420	0.6455
	3	960	0.4241
4	1	1180	0.3908
	2	1420	0.5445
	3	960	0.9325
5	1	1180	0.2788
	2	1420	0.8888
	3	960	0.4877
13	1	1180	0.1279
	2	1420	0.6332
	3	960	0.3468

Wimbledon			
Attribute	Class value	Sample Size	p-value
7	0	59	0.09125
	1	55	0.3946
9	0	59	0.09125
	1	55	0.3946
13	0	59	0.3079
	1	55	0.3309
25	0	59	0.4082
	1	55	0.4213
27	0	59	0.4082
	1	55	0.4213

Wimbledon (x20)			
Attribute	Class value	Sample Size	p-value
7	0	1180	0.1971
	1	1100	0.9794
9	0	1180	0.2016
	1	1100	0.9726
13	0	1180	0.6850
	1	1100	0.8842
25	0	1180	0.1279
	1	1100	0.8491
27	0	1180	0.1141
	1	1100	0.3873

Artificial			
Attribute	Class value	Sample Size	p-value
1	1	100	0.3663
	2	150	0.7725
2	1	100	0.6079
	2	150	0.7293
3	1	100	0.6096
	2	150	0.8214
4	1	100	0.7624
	2	150	0.8493
5	1	100	0.3160
	2	150	0.8708

Artificial (x20)			
Attribute	Class value	Sample Size	p-value
1	1	2000	0.7971
	2	3000	0.5861
2	1	2000	0.8443
	2	3000	0.5076
3	1	2000	0.5103
	2	3000	0.2204
4	1	2000	0.5918
	2	3000	0.9322
5	1	2000	0.3034
	2	3000	0.9332

Appendix E: Monte-Carlo results

The Monte-Carlo cross validation runs 25 times. We get 25 percentages wrongly classified unseen instances per trained Naïve Bayes classifier (one for the continuous attributes, one for the discretized attributes using equal frequency and one for the discretized attributes using MLDP). Those percentage will be shown in the tables below per dataset.

Attempt	Continuous	EqualFrequency
1	19.44%	25%
2	36.11%	36.11%
3	27.78%	36.11%
4	33.33%	36.11%
5	33.33%	33.33%
6	38.89%	41.67%
7	27.78%	41.67%
8	30.56%	33.33%
9	33.33%	47.22%
10	41.67%	44.44%
11	25%	50%
12	22.22%	30.56%
13	47.22%	36.11%
14	19.44%	44.44%
15	19.44%	38.89%
16	25%	33.33%
17	36.11%	44.44%
18	25%	44.44%
19	33.33%	44.44%
20	19.44%	33.33%
21	30.56%	41.67%
22	27.78%	50%
23	30.56%	33.33%
24	27.78%	36.11%
25	13.89%	33.33%

Figure 1: Planning Relax

Attempt	Continuous	EqualFrequency
1	27.61%	27.47%
2	28.57%	28.85%
3	28.98%	29.53%
4	28.02%	28.57%
5	29.26%	29.95%
6	29.26%	29.53%
7	28.16%	28.43%
8	28.85%	28.71%
9	27.88%	28.16%
10	28.3%	28.57%
11	28.3%	28.02%
12	28.71%	28.85%
13	28.85%	29.67%
14	29.67%	29.26%
15	27.34%	27.2%
16	26.37%	26.92%
17	29.12%	28.98%
18	28.02%	27.75%
19	26.92%	26.79%
20	28.02%	28.71%
21	30.08%	30.22%
22	28.16%	28.3%
23	29.12%	29.95%
24	28.98%	29.4%
25	28.85%	29.53%

Figure 2: Planning Relax (x20)

Attempt	Continuous	EqualFrequency	MDLP
1	9.52%	7.14%	9.52%
2	7.14%	2.38%	9.52%
3	11.9%	14.29%	16.67%
4	4.76%	9.52%	14.29%
5	9.52%	11.9%	11.9%
6	14.29%	14.29%	14.29%
7	9.52%	11.9%	16.67%
8	4.76%	9.52%	9.52%
9	14.29%	9.52%	11.9%
10	7.14%	4.76%	16.67%
11	0%	9.52%	4.76%
12	9.52%	4.76%	7.14%
13	11.9%	14.29%	14.29%
14	7.14%	4.76%	7.14%
15	9.52%	14.29%	14.29%
16	7.14%	7.14%	4.76%
17	9.52%	11.9%	9.52%
18	9.52%	4.76%	4.76%
19	11.9%	11.9%	7.14%
20	14.29%	7.14%	9.52%
21	9.52%	9.52%	9.52%
22	11.9%	14.29%	14.29%
23	14.29%	16.67%	14.29%
24	9.52%	11.9%	14.29%
25	9.52%	14.29%	9.52%

Figure 3: Kernels

Attempt	Continuous	EqualFrequency	MDLP
1	3.81%	4.29%	4.05%
2	2.5%	3.33%	3.93%
3	4.64%	4.76%	5%
4	3.69%	3.93%	3.57%
5	3.1%	4.29%	4.05%
6	3.57%	4.52%	3.69%
7	3.81%	3.81%	4.64%
8	2.5%	2.74%	2.74%
9	2.98%	3.69%	3.57%
10	3.21%	3.45%	3.45%
11	3.81%	3.69%	3.69%
12	3.69%	4.17%	4.29%
13	3.21%	4.17%	4.05%
14	2.86%	3.69%	3.1%
15	3.1%	2.98%	4.05%
16	3.21%	3.45%	2.74%
17	3.45%	4.29%	4.52%
18	3.45%	3.57%	3.93%
19	3.93%	4.52%	5.24%
20	3.21%	3.45%	3.21%
21	3.57%	3.93%	4.05%
22	4.4%	4.88%	4.88%
23	3.57%	4.05%	4.17%
24	3.81%	4.64%	4.4%
25	4.4%	4.29%	4.17%

Figure 4: Kernels (x20)

Attempt	Continuous	EqualFrequency	MDLP
1	5.56%	2.78%	11.11%
2	11.11%	13.89%	13.89%
3	5.56%	5.56%	5.56%
4	16.67%	13.89%	30.56%
5	11.11%	13.89%	8.33%
6	13.89%	16.67%	13.89%
7	8.33%	8.33%	8.33%
8	8.33%	11.11%	8.33%
9	0%	2.78%	5.56%
10	5.56%	5.56%	2.78%
11	5.56%	2.78%	5.56%
12	2.78%	5.56%	2.78%
13	0%	5.56%	2.78%
14	5.56%	5.56%	8.33%
15	2.78%	8.33%	5.56%
16	2.78%	8.33%	2.78%
17	5.56%	19.44%	8.33%
18	13.89%	13.89%	8.33%
19	11.11%	5.56%	5.56%
20	11.11%	11.11%	8.33%
21	5.56%	8.33%	2.78%
22	13.89%	8.33%	2.78%
23	19.44%	22.22%	13.89%
24	8.33%	8.33%	2.78%
25	11.11%	8.33%	2.78%

Figure 5: Wine

Attempt	Continuous	EqualFrequency	MDLP
1	5.9%	7.02%	6.6%
2	5.9%	7.02%	7.3%
3	6.46%	7.02%	7.3%
4	6.6%	7.58%	7.3%
5	5.9%	7.16%	5.48%
6	4.35%	6.74%	5.62%
7	6.18%	7.72%	7.44%
8	5.2%	6.32%	6.46%
9	5.48%	5.9%	6.88%
10	5.06%	6.32%	6.04%
11	4.78%	5.9%	6.04%
12	5.06%	5.9%	5.9%
13	6.04%	6.18%	6.6%
14	7.16%	8.01%	8.15%
15	3.93%	5.34%	5.34%
16	7.3%	7.44%	7.3%
17	5.9%	7.02%	5.2%
18	6.18%	6.88%	7.16%
19	5.48%	7.44%	7.87%
20	5.48%	6.04%	5.9%
21	5.48%	6.18%	6.6%
22	6.74%	7.3%	7.44%
23	4.35%	6.32%	6.88%
24	5.06%	6.32%	7.87%
25	5.34%	6.18%	6.6%

Figure 6: Wine (x20)

Attempt	Continuous	EqualFrequency
1	52.17%	34.78%
2	39.13%	34.78%
3	47.83%	52.17%
4	60.87%	39.13%
5	47.83%	43.48%
6	39.13%	47.83%
7	39.13%	34.78%
8	34.78%	47.83%
9	43.48%	34.78%
10	52.17%	52.17%
11	60.87%	65.22%
12	56.52%	52.17%
13	30.43%	30.43%
14	47.83%	47.83%
15	47.83%	39.13%
16	43.48%	43.48%
17	34.78%	43.48%
18	30.43%	52.17%
19	39.13%	34.78%
20	52.17%	34.78%
21	43.48%	52.17%
22	30.43%	52.17%
23	52.17%	47.83%
24	43.48%	34.78%
25	39.13%	43.48%

Figure 7: Wimbledon

Attempt	Continuous	EqualFrequency
1	35.09%	35.09%
2	28.51%	28.95%
3	32.02%	33.55%
4	31.8%	33.55%
5	30.92%	31.58%
6	32.46%	32.46%
7	30.26%	30.48%
8	33.33%	32.68%
9	33.33%	35.09%
10	33.77%	36.18%
11	26.75%	27.63%
12	32.02%	31.14%
13	31.8%	31.58%
14	33.55%	32.68%
15	28.73%	28.95%
16	29.82%	29.39%
17	28.95%	31.8%
18	30.26%	31.36%
19	30.7%	31.36%
20	28.95%	29.82%
21	32.68%	32.89%
22	34.21%	35.31%
23	32.02%	34.43%
24	31.8%	31.8%
25	31.58%	31.36%

Figure 8: Wimbledon (x20)

Attempt	Continuous	EqualFrequency	MDLP
1	2%	4%	2%
2	4%	6%	4%
3	2%	2%	4%
4	4%	4%	10%
5	12%	12%	10%
6	8%	10%	12%
7	8%	8%	16%
8	2%	2%	4%
9	4%	8%	10%
10	6%	4%	8%
11	12%	14%	14%
12	6%	12%	10%
13	6%	8%	10%
14	6%	6%	4%
15	2%	6%	4%
16	4%	2%	10%
17	6%	6%	6%
18	10%	10%	12%
19	6%	4%	8%
20	4%	8%	8%
21	8%	8%	6%
22	8%	6%	14%
23	2%	2%	12%
24	10%	6%	6%
25	4%	8%	6%

Figure 9: Artificial

Attempt	Continuous	EqualFrequency	MDLP
1	7%	7.4%	8.2%
2	6.9%	7.3%	6.9%
3	7%	7.4%	7.5%
4	7%	7.7%	8.1%
5	6%	6.7%	7.1%
6	7.4%	8%	7.8%
7	6.7%	7.6%	8.3%
8	7.9%	9%	9.1%
9	6.6%	7.4%	7.3%
10	6.3%	6.6%	7.3%
11	6.9%	7.6%	7%
12	6.2%	7%	7.5%
13	6%	7.2%	7.3%
14	5.6%	6.6%	6.3%
15	7.3%	8.2%	7%
16	5.9%	6.7%	7.4%
17	5.8%	6.5%	6%
18	7.3%	8.4%	7.6%
19	7.2%	8.9%	8.4%
20	6.9%	7.9%	7.9%
21	6.4%	6.4%	7.6%
22	7.4%	8.2%	7.7%
23	6.7%	7.6%	7.6%
24	5.9%	6.8%	6.4%
25	6.9%	7.9%	6.9%

Figure 10: Artificial (x20)