ADM: Algorithms for Data Mining

Discretization for Naive Bayes

Marc Maynou Yélamos

March 20, 2022

Abstract

The goal of this paper is to explore the hypotheses stated in Yang and Webb 2008 [1], which essentially describe the benefits of using discretization methodologies to transform continuous variables into categorical ones when performing classification tasks. This procedure is proposed in opposition to fitting a probability function for the continuous variables and generating a prediction through the likelihood of each point belonging to the distribution. Although the optimizations described are specifically tested for the Naive Bayes classifier, the conclusions obtained can have a wider range of utilities, given that the categorical transformation of numerical variables is a fairly common process. More precisely, this document will serve as an overview of the Naive Bayes algorithm and the main discretization techniques used (both the most common in the machine learning field as well as the proposed specifically for this experimentation). The improvements will be tested with practical tests in order to get empirical results to support the claims stated.

1 Naive Bayes algorithm

The Naive Bayes algorithm is based on the Bayes' theorem[4] used in probability theory and statistics, which describes the probability of an event happening based on prior knowledge of the characteristics of the event. For instance, if the probability of a person having lung cancer increases with the amount of cigarettes smoked, then Bayes' theorem allows the assessment of the risk of having lung cancer to be more accurate by conditioning it on how much the person has smoked. This is in opposition to just assuming the individual does not present a deviation from the population (i.e. there is no difference with regards to the mean person). The formula that describes this behavior is the following one:

$$P(A|B) = \frac{P(A|B)P(A)}{P(B)}$$

More specifically, by using this equation we can find the probability of A happening, given that B has occurred, with A being the hypothesis and B being the evidence. The Naive Bayes algorithm[3] uses this principle to try to predict a certain result, conditioning the particular outcome to the other characteristics presented by the observation.

This algorithm is labeled as naive because an assumption is made: the features of the dataset are independent of each other. This is, generally speaking, highly unlikely, as at least some of the features of a dataset tend to be related in some way. However, for the sake of simplicity, we assume that these are not related, reducing the amount of calculations and its complexity, hence the naive connotation. Through this simplification the algorithm is made easier to implement and compute while still being fairly effective as a classification tool.

2 Naive Bayes variables

There are a lot of algorithms used to perform data mining, which widely vary in a lot of properties, such as the mathematical foundation they are based on, the kind of learning (supervised or unsupervised) and the typology of the conclusions they obtain (classification, regression, data exploration, etc.), just to name a few. One of these aspects that changes from algorithm to algorithm is the kind of data they use. Naive Bayes, as stated before, is built over the computation of the conditional probabilities

of each individual instance. This means that each individual observation in the test set is classified into a category based on what is most probable given the observations analyzed in the training set. The usage of these conditional probabilities imply that the attributes used need to have a specific probability that can be worked with. As a result, numerical variables need to be transformed, as the probability of belonging to a given label is not meaningful when there is a high number of possible categories (one per numerical value).

As highlighted in Yang and Webb 2008[1], the initial approach used in machine learning techniques was to fit a distribution to predict the data. That is, a probability function was used to describe the distribution of the value for each individual label and, therefore, calculate the specific probability of that value belonging to the category. This is useful as it allows the model to work with specific probabilities: when a data point arrives, the values it presents are evaluated with the functions, which return the likelihood of that observation belonging to that distribution. There is, however, a crucial problem with this approach that questions the efficiency of this methodology as a generalized procedure: the data might not strictly follow a distribution or this same distribution may be unknown. This poses a big issue as, in order to proceed with the analysis, a distribution has to be arbitrarily chosen. In most cases the normal distribution is used to describe the data given that is the most common and a lot of both natural and artificial phenomena seem to describe it.

However, in latter years, discretization techniques started to appear as solid alternatives, as these avoided the need to make unsafe assumptions regarding the data. The continuous variable is divided into bins, each of them containing a given number of instances grouped under a single label. This approach reduces the number of numerical values from (potentially) thousands to just a few, which makes the probability of each possible label meaningful. However, it is important to take into account that qualitative data has a lower level of measurement than quantitative data. This is obvious given that the possible number of values that the attribute can take is reduced as a result of organizing them into bins. Consequently, discretization may lead to information loss, which will affect the classification's variance and bias. Essentially, the system is losing precision in the data it works with in order not to make unsafe assumptions regarding its distribution.

3 Discretization techniques

The most common discretization strategies nowadays are EFD (Equal-Frequency Discretization) and EWD (Equal-Width Discretization), given that they are simple and easy to use while giving decent results. EWD discretizes by creating bins that have equal width. That is, each bins represents an equal range of the original variable values, no matter how many observations are in each bin. EWD creates n number of bins following this pattern, indicating both a lower and upper threshold for each bin with regards to the specific values seen. This implies that the bins are probably unbalanced, meaning that some of them (generally speaking, those closer to the means) might include considerably more values than some others (generally speaking, those far away from the means). Contrary to this approach, EFD creates n bins that have the same frequency, that is, that contain the same number of observations (if possible). Both methods require the number of bins (n) to be defined by the user. Therefore, in order to get the best result, multiple iterations with different values of n might be needed. An example of both methods is presented next:

- Data: 0,4,12,16,16,18,24,26,30
- Equal-Width Discretization (n = 3)
  - 30 (highest value) - 0 (lowest value) = 30. 30/3 = 10. Each bin holds a range of 10 values. Bear in mind that, as we have to take into account the values at the tails, we are, in fact, working with 31 values, so one bin will hold 11 values.
  - Bin 1 [0,9]: 0,4
  - Bin 2 [10,19]: 12,16,16,18
  - Bin 3 [20,30]: 24,26,30

- Equal-Frequency Discretization (n = 3)
  - There are 9 values that need to be classified. $9/3 = 3$. Each bins contains 3 values
  - Bin 1: 0,4,12
  - Bin 2: 16,16,18
  - Bin 3: 24,26,30

With EWD, clusters are easier to maintain, as the system is not arbitrarily creating a separation point regardless of whether the next value is related or not to the prior bin. This is a problem that EFD has, and that can cause misclassifications given that a group of very related observations might be cut down the middle. On the other hand, EFD generates balanced bins (equal amount of observations), which allow the classifier to have more instances of each label (i.e. all labels will be equally represented. There will not be one label with 100 values and another with just a couple). Both EFD and EWD present some advantages and some inconveniences, and choosing one or the other usually depends on the kind of data at hand. Both, however, present a common problem: they require the number of bins to be fixed before the execution.

The conclusion to which Yang and Webb[1] essentially arrived is that the best way to discretize a variable has to take into account the trade-off between the number of bins and the size of each one (the more intervals, the less data in each, and vice versa). When training the model, it is desirable to have as many bins as possible, as this means that the data is more clearly defined and the learner can better detect the patterns in the associations. However, when predicting values, we want to have as many instances as possible in each bin, as to minimize the chance of wrongly classifying a value. Both the size and the number of bins are relevant, and both EFD and EWD do not take them into account, as the number of bins is arbitrarily decided by the user.

In order to handle this issue, Yang and Webb proposed two need discretization methods:

- The first one is PD (Proportional Distribution), which splits the dataset into n bins, each also containing n instances. For example, a dataset with 10.000 observations would be discretized into 100 bins, each containing 100 observations ($100 \cdot 100 = 10.000$). If the dataset had 20.000 instances, 142 groups would be created, each with 141 or 142 bins (as the result is not an integer). Essentially, the number of bins is the ceiling of the square root of the number of instances.

- The second proposed technique is FFD (Fixed-Frequency Distribution). Similarly to EFD, this method creates bins that have the same frequency (i.e. number of observations in each bin). However, instead of fixing the number of bins and equally distributing the instances, FFD fixates the amount of instances in each bin and then creates as many bins as required to fit all instances. For instance, if we had a dataset of 1.000 observations, EFD with $n = 10$ would create 10 bins, each with 100 observations. However, in FFD, $n = 10$ would indicate that n observations are assigned to each bin. So, for example, if $n = 20$, FFD would create as many bins of 20 observations as necessary (in this case 50 bins; $50 \cdot 20 = 1.000$).

Both methods give the number-size trade off more relevance, structuring the bins around it. PD does so by equating the number of bins to the number of instances in each bin, essentially finding the best balance possible. This means that there are no parameters that can be altered to better tailor the method to a specifics dataset. On the other hand, it will, on average, perform better than FFD. This is so because FFD still has the size of each bin as a parameter, which implies that it can be altered. Finding the best n might require some trial and error, but it also allows the discretization to adapt better once the best value has been found.

4 Experimental part

As a way to complete the development of this paper, and perform a more complete investigation, I decided to compare the discretization implementations in a practical way: through evaluating a prediction mechanism with different discretization techniques, in order to observe if the improvements do appear. It is important to note that the original paper already tests the stated hypotheses, and these tests can be found in the original document. It does so by comparing the efficiency (classification error rate) of the Naive Bayes classifier for different discretization methods for different datasets. Given the

experimental and academic purpose of this document, this analysis will not be performed with such detail, as only two dataset will be tested. Additionally, and given that the goal of this experiment is to test the effects of changing the discretization methods, the preprocessing stage will not be thoroughly developed.

The main comparison will be between a Gaussian Naive Bayes and a multinomial one, as a way to answer the initial hypothesis of the document. To develop a more complete analysis, different discretization techniques will also be applied, with its efficiency tested. This will give an overview of the impact (or lack thereof) of each method. It is important to highlight that by no means I am trying to reach a categorical conclusion with regards to some methods being better than others. The data used may be more suited to apply some kinds of discretization rather than others, which may bias the results. Additionally, the preprocessing steps applied are, as stated beforehand, very general and not specifically tailored to the kind of test I wanted to perform, as this was not the purpose of the experiment. As a result, to get something resembling a definitive conclusion more tests should be performed, over more datasets, with a more detailed preprocessing and with a validation method. Nonetheless, I believe certain conclusions can still be extracted from the results.

Two datasets have been used to develop the tests:

- *Adult income*[1]: it details an individual's annual income based on different factors. This income is presented as a categorical value with two levels: $\leq 50k$ and $>50K$, which is what is being tried to predict. It contains 15 columns, 6 of which are continuous. However, two of them have been eliminated from the model to improve performance, so only 4 variables are discretized.

- *Red wine quality*[2]: this dataset is related to the red variant of the Portuguese *Vinho Verde* wine. It contains 11 different metrics about the wines characteristics and a grade that defines it's quality (from 0 to 10). The model will try to correctly predict the quality of a wine given the values of its characteristics, discretizing all 11 variables.

Both datasets were used to train different Naive Bayes predictors. On the one hand, a Gaussian Naive Bayes was developed as the baseline to test the main hypothesis: *discretizing variables obtains better results than fitting a probability distribution.* On the other hand, multiple multinomial Naive Bayes were built, each with different discretization methods. More specifically:

- *Quantiles* discretization: "basic" discretization that generates 4 bins of equal size (same amount of boundaries), the boundaries of each corresponds to the numerical quantiles of the data.

- EFD (Equal-Frequency Discretization): generates n number of bins of equal size. The method is the same as with the quantile discretization, but it can use other numbers of bins. The most common approach is to use $n = 10$, and, therefore, it will be the one used.

- EWD (Equal Width Discretization): generates n intervals for each variable, each covering an equal amount of values. Then the observations are classified with regards to which interval they fall into. The most common approach is to use $n = 10$, and, therefore, it will be the one used.

- FFD (Fixed-Frequency Discretization): similar to EFD in that it creates a specific number of bins, each with the same number of instances. However, with FFD, the size of the bins is fixed, not the number of bins. The recommended value is 30, so that will be the size used.

- PD (Proportional Discretization): creates a number n of bins, such that each bin also contains n instances.

For the sake of offering a more complete comparison, the null model's accuracy score will also be indicated. This value corresponds to using no model to predict the target, and, instead, assigning the most common label to each one. This allows us to see whether the differences in the accuracy are relevant or not, given that a very unbalanced dataset (i.e. very high null score) might be more difficult to get improvements of. As it can be seen in the tables below, the *Adult income* dataset has a high null score, whereas the *Red wine quality* dataset has a considerably lower one.

Another important distinction has been between the amount of values that have been discretized, being 4 in the first case and 10 in the second. This is, potentially, another analysis point, given that, if

[1] https://www.kaggle.com/wenruliu/adult-income-dataset
[2] https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009

discretizing is better than fitting a probability distribution, the more values are discretized, the better the accuracy score should be. These differences imply that testing both datasets, although insufficient to obtain more solid conclusions, can already indicate important differences.

The code was developed using python notebooks with libraries such as pandas, scikit-learn, numpy or feature-engine for the implementation of the methods. The code can be found in the next address: https://github.com/Minat99/ADM-Deliverable-1/tree/main/Code. The results of the experiments are as follows:

Accuracy in the *Adult income* dataset	
Model/Discretization technique	Accuracy (in %)
Null score	76.01
Gaussian	79.84
Quantiles	80.85
EFD (n = 10)	80.81
EWD (n = 10)	82.13
FFD (n = 30)	79.87
PD	83.76

Table 1: Accuracy comparison of the discretization methods for the *Adult income* dataset

Accuracy in the *Red wine quality* dataset	
Model/Discretization technique	Accuracy (in %)
Null score	43.33
Gaussian	55.21
Quantiles	56.67
EFD (n = 10)	54.79
EWD (n = 10)	59.17
FFD (n = 30)	56.04
PD	56.87

Table 2: Accuracy comparison of the discretization methods for the *Red wine quality* quality dataset

5 Conclusions

The first noticeable fact is that all predictive models perform better than the null model, and in a considerable way. This indicates that the Naive Bayes algorithm does provide a remarkable boost in performance, especially so given that the data has not been thoroughly prepared. The second important point would be that in almost all cases using discretization techniques generates better results than fitting a Gaussian distribution, as there is only one case in which this does not hold true. Related to this second point is that, in general terms once again, the improvements are quite noticeable, averaging a solid 1% increase in the accuracy rate. This confirms, for the limited scope of this experiment, that discretizing continuous variables produces better precision results.

It is interesting to see that the only case where the Gaussian distribution obtains a better result is by using the EFD discretization in the *Red wine quality* dataset. EFD is the most popular discretization technique given the reasons stated in prior sections. However, it does not always produce the best results, which is yet another proof that discretization should not be applied mindlessly when trying to get the best model. Another conclusion worth highlighting is that the two new proposed algorithms perform better than the Gaussian distribution but (i) just slightly and (ii) with clear differences. PD presents a solid improvement in both cases, but FFD is just barely better with the *Adult income* dataset and in the *Red wine quality* dataset it presents the smallest improvement. However, the reason why may be that there are too many bins.

The implementation of FFD used creates bins of 30 instances (as it is the recommended value) and, as a result, with thousands of instances, it creates a lot of bins that may hinder the predictive capabilities. Varying the size of the bins with respect to the dataset is probably the best approach and

one that, if done correctly, may yield better results than other discretization methods. On the other hand, PD presents very good results, generating the best accuracy in the *Red wine quality* dataset and the second best in the *Adult income* dataset. As the name of the technique suggests, the amount of bins scales proportionally to the size of the dataset, reason why it may be more reliable in most cases as it only generates more bins when the data allows it to do so.

On average, the improvements over the *Red wine quality* dataset are higher than with the *Adult income* dataset. This could be attributed to the amount of variables discretized, which was much more elevated in the former case. Therefore, the more variables to discretize the better the results, as the improvements for discretizing each one "add up". However, this may also be because the null accuracy was very poor in the *Red wine quality* dataset, which meant that there was more room to obtain better results and, as such, improvements were easier to achive. This lower base accuracy is mainly due to the *Red wine quality* dataset target presenting more possible labels than the *Adult income* dataset. Once again, with this limited data we can not reach a solid conclusion.

The quantile discretization, while very simple, yields solid improvements, which indicates that in most cases it may be better to just apply a simple and easy discretization rather than both using a Gaussian distribution or a more complex discretization method, specially when we are not sure of which is the bets discretization method to use. As a final remark, the EWD also performs really well in both cases, specially in the *Red wine quality* dataset, where it presents a huge difference with regards to the other methods, evidencing that, even though on paper EFD seems to be a more solid approach, EWD also performs well.

I would like to note that, once again, the scope of this experiment was very limited and in order to get more solid conclusions, more tests should be conducted. One evidence of this is that the *Adult income* dataset was also used in the original paper of Yang and Webb and the results obtained were quite different. Multiple factors can contribute to that, such as the preprocessing steps, the validation methods used and differences in the implementation of methods. There are a lot of discretization procedures that are based on a wide range of mathematical principles. This is highlighted in Salvador et al[2], in which different discretization techniques are used over different dataset with different classificators and, as it can be observed, the accuracy vary substantially Nevertheless, the main hypothesis has been successfully tested and, even though the conclusions can not be regarded as general true statements, the experimental part demonstrates that the conceptual ideas presented hold a certain degree of validity.

References

[1] Y. Yang and G.I. Webb. "Discretization for naive-Bayes learning: managing discretization bias and variance". In: *Mach Learn* 74 (2008), pp. 39–74. DOI: https://doi.org/10.1007/s10994-008-5083-5.

[2] Salvador García et al. "A Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 25 (Feb. 2013). DOI: 10.1109/TKDE.2012.35.

[3] Nagesh Singh Chauhan. *Naïve Bayes Algorithm: Everything you need to know*. [Online; accessed 20-March-2022]. 2020. URL: https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html.

[4] Wikipedia contributors. *Bayes' theorem — Wikipedia, The Free Encyclopedia*. [Online; accessed 20-March-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Bayes%5C%27_theorem&oldid=1077742010.