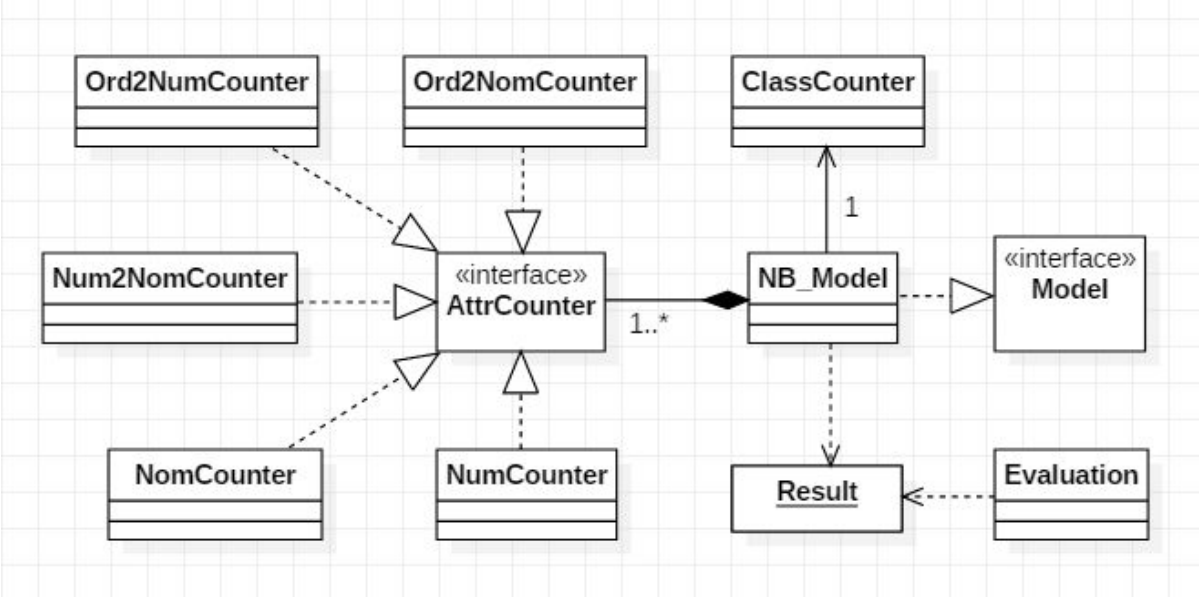
COMP30027 Machine Learning, 2020 Semester 1

Assignment 1: Naive Bayes Classifiers

Student Names: Yue Peng; Tian Qiu
Student IDs: 958289; 988121

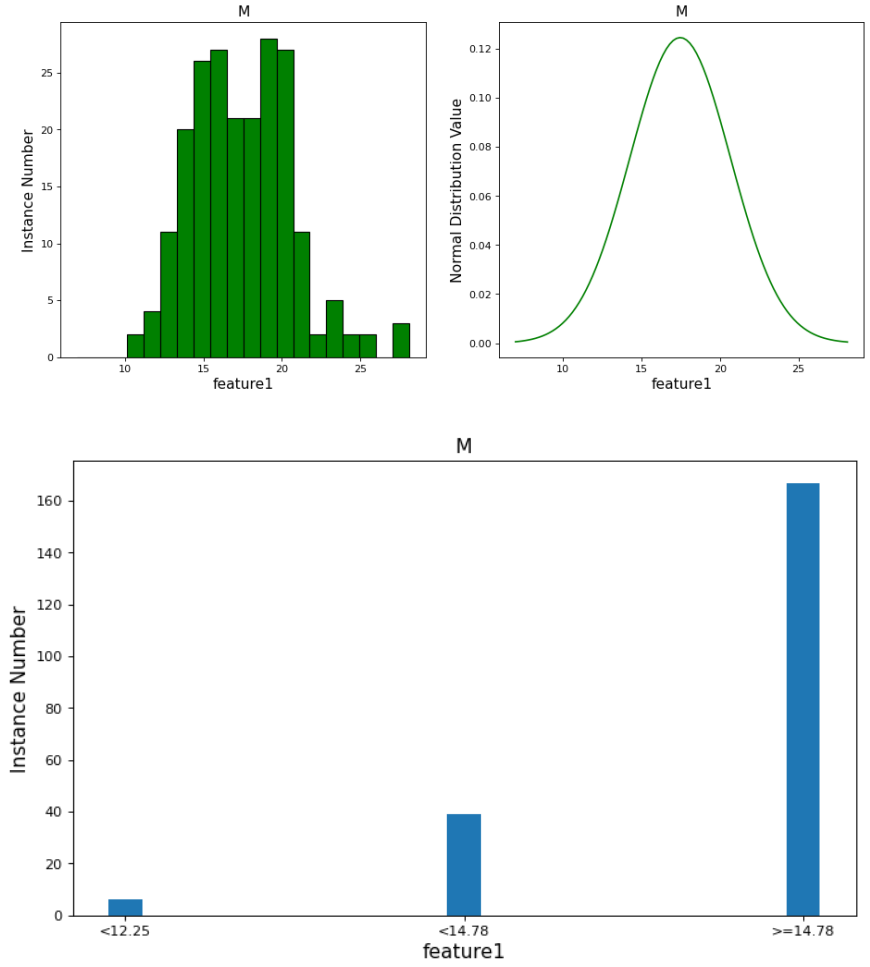
1. Baisc Implementation



We construct our program using object oriented way. The Naive Bayes model implementing the Model interface where we defined the preprocess(), train() and predict() function. We leave the evaluate() function to a evaluation class which can also draw graph and charts other than simply output statistics. The evaluation class takes a Result object(e.g. [(truth value, predict value)...]) generated from the predict() function of the model class, processes the result and tell us how the model perform.
A NB_Model contains a single ClassCounter and several AttrCounter. They are the basic calculation units to read and counter classes/attributes in preprocess stage, and to calculate the probability of classes/attributes in train stage, and finally to return the probability for a specific value in predict stage. Attribute counters consist of:

- NomCounter - for nominal attributes
- NumCounter - for numeric attributes
- Num2NomCounter - to convert numeric attributes to nominal attributes (Discretisation)
- Ord2NomCounter - to convert ordinal attributes to nominal attributes
- Ord2NumCounter - to convert ordinal attributes to numeric attributes

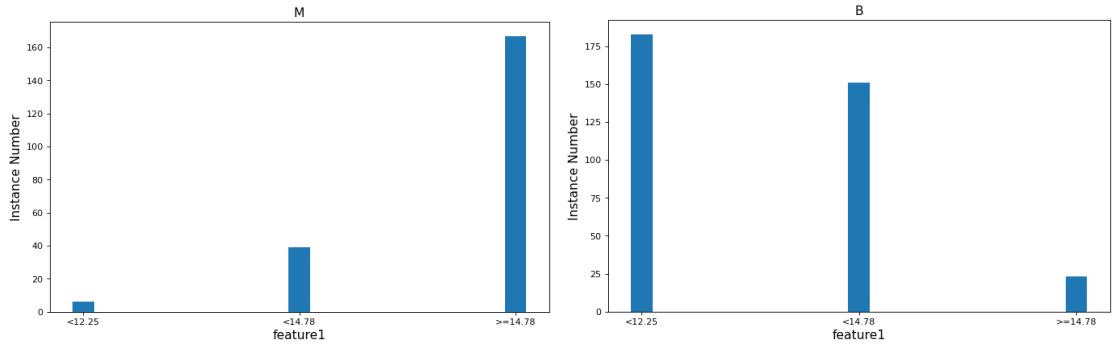
2. Discretising the Numeric Attributes



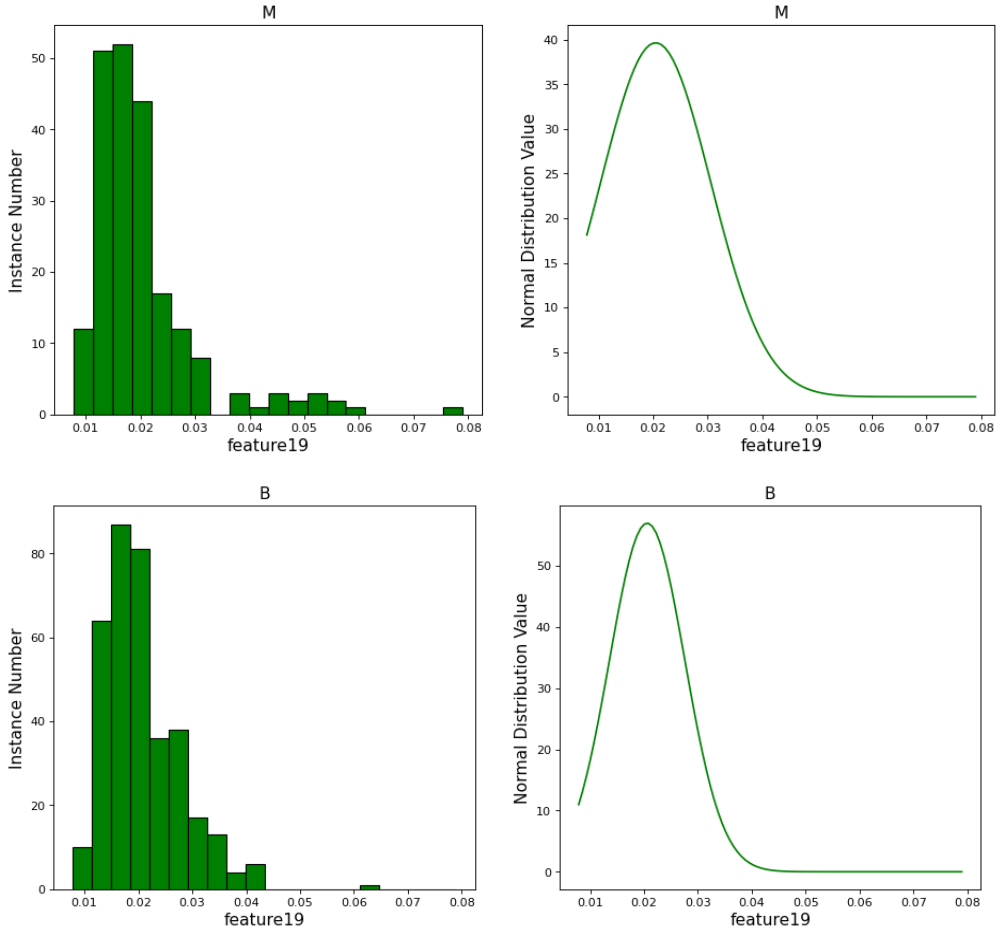
We use the equal frequency method to discretise numeric attributes. The above graph shows the feature1(the first feature)'s distribution among "M" class in the wdbc dataset. On the top left is the row data distribution and the top right is the normal distribution constructed on the data; while on the bottom it shows the result after we split the data equally into 3 folds and the data distribution over these 3 folds in class "M".

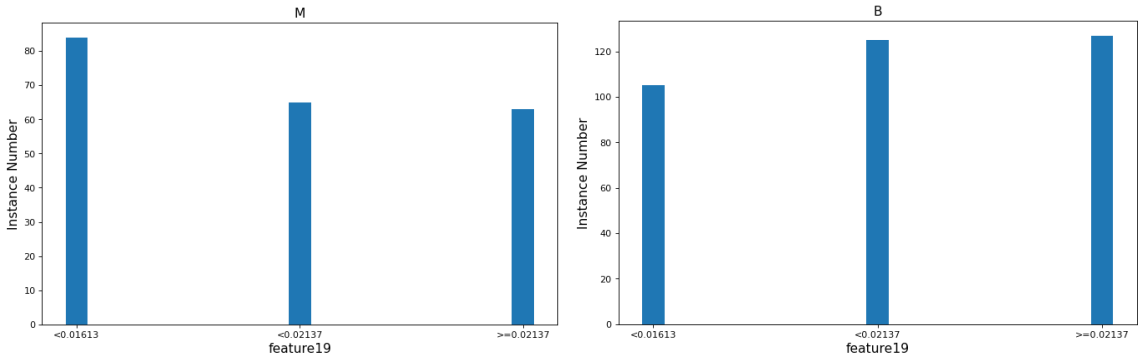
In comparison with the discretisation, the normal distribution can describe the row data distribution better. However, from the 3-fold distribution we can hardly imagine the distribution of the row data and we only know how many instances in each interval, which means the entropy is low so that the model knows little about the data. Therefore, the discretisation will actually decrease the performance of the model.

But in fact, discretisation does not decrease the performance much. We get 0.940246 accuracy in no discretisation result and 0.938489 accuray in discretisation result, and the recall/precision is also very close. Why is the result so close? Actually, the model already gets enough information from the 3 folds such that they can tell which class the instance belongs to for most attributes. For example, we can clearly tell the distribution of feature 1 among class "M" and class "B" is different from the following graph. If we find feature 1 less than 12.25 we can be very sure that the instance belongs to class "B", or greater than 14.78 then it belongs to "M".

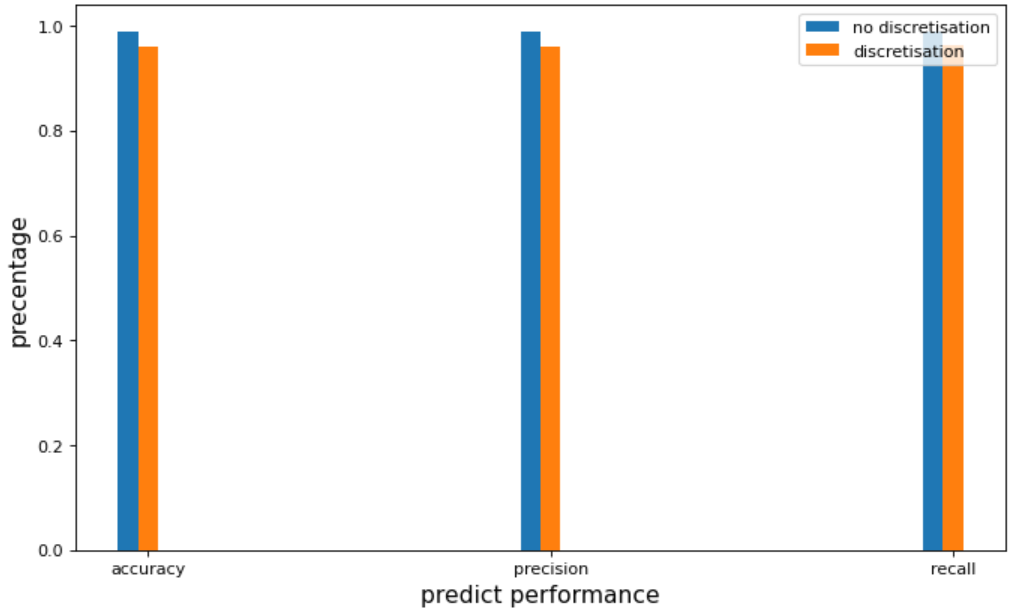


The 3 folds did show its inaccuracy sometimes. In feature 19, the distribution of the 3 folds is quite similar, in other words, less confidence to determine which class the instance belongs to for every interval. More specifically, if we found a feature 19 value at 0.05, the instance would probably belong to class ‘M’ , and The normal distribution supports this. However, 0.05 falls in the interval of >=0.02137 where we get class ‘M’ probability less than class ‘B’. So our 3 folds tell us a wrong result by just looking at feature 19.

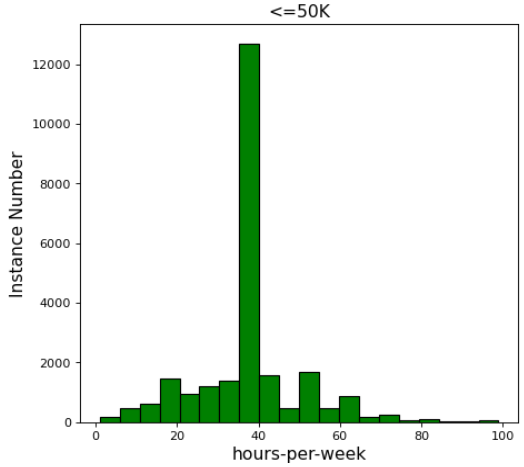




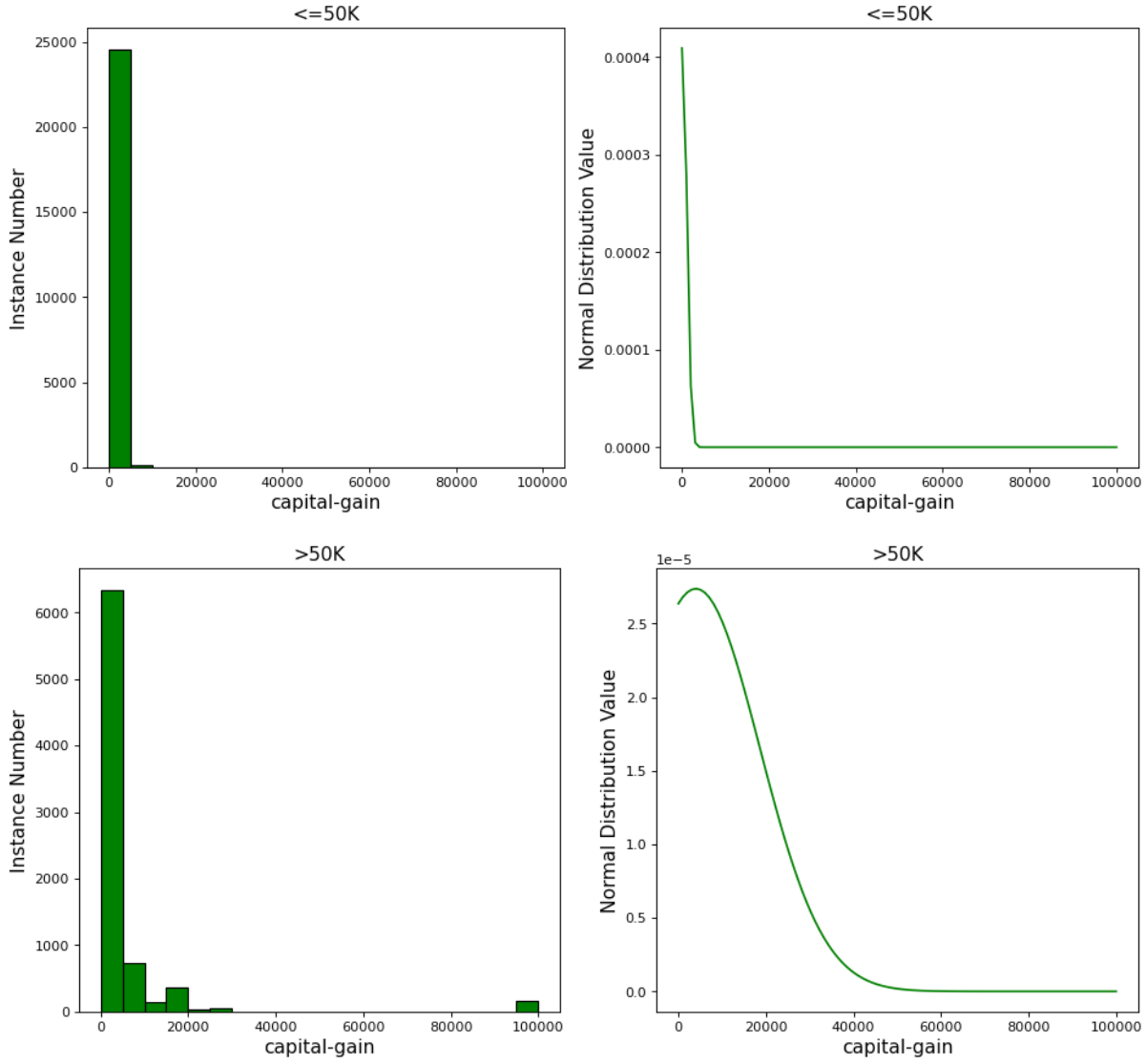
As discussed before, the discretisation method gives correct judges between two classes for most attributes, so the performance does not decrease performance much. Moreover, we also test the wine dataset and the performance differences are shown below.



The equal width method may decrease performance more because it is very sensitive to outliers. But sometimes the equal frequency method does not work. For example, the feature “hours-per-week” in the adult dataset has a dominant “40” value and when we apply an equal frequency method on it we would probably get a meaningless(40, 40) interval.



3. Implement a Baseline Model
4. Handle Ordinal Data
5. violated Gaussian Assumption



The above distribution is from the feature capital-gain of adult dataset. In both “<=50k” class and “>50k” class almost 90 percent of the instances have 0 capital-gain and a very tiny bit of them have some capital-gain. Apparently it is inappropriate to apply a Gaussian assumption on it. For most values other than 0, the assumption overestimates the probability.
There is a more dangerous effect. Actually a bunch of rich people have near 100000 capital-gain. If we test the model with such people, the normal distribution function will return 0 for that capital-gain value because it is too extreme, and multiplying by 0 will ruin our Naive Bayes method. So in my implementation, I assign a very small number to replace 0.
In conclusion, if any attribute in the dataset is not naturally distributed and has some extreme outliers, Gaussian assumption could be violated.