The dataset we chose to implement our data mining algorithm is the wine dataset. It is taken from archive.ics.uci.edu. Wine dataset is one of the most popular datasets available in the website. The dataset is suitable for classification algorithms. It contains no missing values. The attribute values are integers and real numbers. It contains 178 records, which is suitable for learning purpose, and contains 14 attributes including class.

We decided to choose classification because the wine dataset is suitable for classification purposes. Aside from that, we also think that classification is useful to predict future data by using the available samples.

We decided to implement OneR data mining algorithm. We chose OneR because it is fairly simple and straight forward. It learns one level of decision tree therefore it is faster and simpler than decision tree but less accurate. OneR is also frequently used for learning purposes. According to saedayad.com, OneR produces rules that are slightly less accurate than latest classification algorithms. OneR also produces rules that are simple for humans to learn and interpret. Another factor we chose OneR is because it is pretty easy to implement.

We produced 2 java application versions of OneR. They differ in the pre-processing stage. One program is using equal width binning. The equal width binning basically classifies values into a series of bins with equal range/interval (for e.g. 3-5, 5-7, 7-9, …). It is the simplest form of data discretization. The interval can be calculated by using the formula

**Interval = (max-min)/k**

Where:

‘max’ is the maximum value of the attribute values

‘min’ is the minimum value of the attribute values.

‘k’ is the number of bins

After the interval is calculated, we calculate the bin boundaries using:

**Min + interval, min + 2\*interval, …**

Now that the data had been categorized, OneR algorithm can be executed as usual. The figure below shows the pseudo code for the program. Step 4 is the pre-processing stage where the numeric values are being categorized. Step 5-8 is the OneR algorithm implementation.

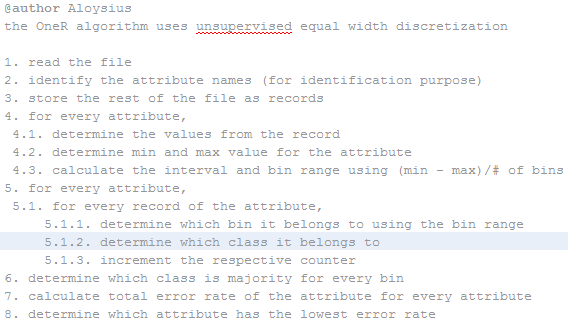


Figure Pseudo code for the OneR Implementation

The other pre-process method we implemented is the error-based discretization. The values should be sorted in ascending order first. We put break points when the majority class changes, thus reducing the number of errors. However it also causes overfitting, where the program/algorithm learns too much from training data, causing it to have low accuracy when being supplied with different test data. An improvement to this is to define minimum bucket size. In our program, we defined the minimum bucket size of 6. That means there should be 6 values included in the bucket with the exception of last bucket where there might be not enough values to satisfy the minimum bucket size. This reduces the overfitting effect but also increase the occurrences of errors.

In our program, we have a list containing values. We then sort the list in ascending order. We iterate through the list. For every value in the list, check the bucket size. If bucket size is less than the defined minimum bucket size, then add the record the bucket. Else, calculate which class is the majority class of the bucket. After that, check the next record. If the next record’s class is the same as the majority class of the bucket, add the record to the bucket and check the next value again. Repeat this until the next record’s class is not equal to the bucket’s majority class or the next record’s value is not equal to the last value in the bucket. Bucket boundaries are calculated by taking the average value between the last value in the bucket and the next value (or the first value of the next bucket).

Figure below shows the pseudo code for our program which uses error-based discretization. Step 5 shows the pre-processing stage. Step 6-10 are the implantation of the OneR algorithm.

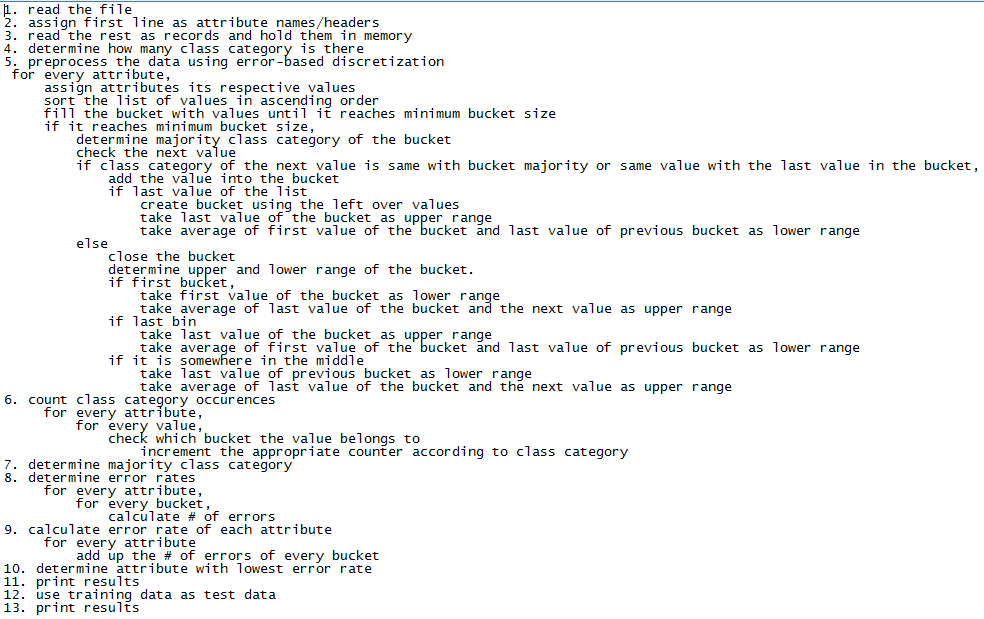


Figure Pseudo Code for OneR implementation

There are limitations regarding the dataset our program can use. The file that contains the dataset should be separated using comma. The first line of the data set should contain attribute headers. The class attribute (attribute that will be predicted) should be placed in first column. The class attribute should be categorical. The values of the attributes should be numerical because the pre-processing stage involves the categorization of continuous values.

Our program will output information such as: # of attribute, # of records, best rule, error rate of best rule, rules, and percentage of correctly classified and incorrectly classified records.

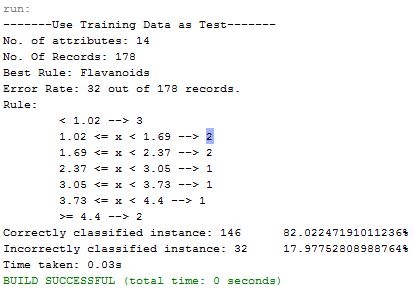


Figure output of the program (equal width binning)

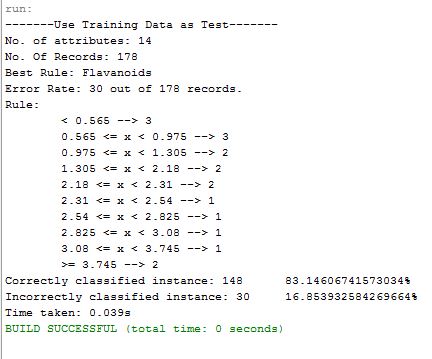


Figure output of our program (error-based discretization)

To run the file on NetBeans, you can go to Run menu item in the menu bar and click ‘run file’ (keyboard shortcut shift+f6).

(Note: we don’t include codes since it will become complicated and too technical. Therefore, we include pseudo code instead)