Assignment 2: Apriori Rule Generation

TDT4300

Jonas Myrlund

March 17, 2014

1 Decision trees

1.1 Compute the Gini index for the entire training set.

The distribution of classes in the entire training set is as follows:

	Count
Buy PC = Yes	12
Buy PC = No	8

This results in the following Gini index:

$$\text{Gini} = 1 - (12/20)^2 - (8/20)^2 = 0.48 \quad (1)$$

1.2 Compute the Gini index for the UserID attribute.

Considering the UserID attribute categorical, the values are unique throughout the entire data set, and thus every value has only one outcome. The Gini index will therefore be 0.

This, however, is useless – if a new UserID is to be classified, the decision tree will not be able to select a class.

Seeing as the number doesn't have a causal relationship to the class, the Gini index will converge to 0.5.

1.3 Compute the Gini index for the Age attribute.

Age	Buy PC = Yes	Buy PC = No	Gini
young	4	3	0.49
medium young	5	1	0.28
old	3	4	0.49

Table 1: Gini index for the Age attribute.

The results in table 1 gives a weighted average Gini index of 0.43.

1.4 Compute the Gini index for Student attribute.

Student	Buy PC = Yes	Buy PC = No	Gini
yes	8	2	0.32
no	4	6	0.48

Table 2: Gini index for the Student attribute.

The results in table 2 gives a weighted average Gini index of 0.40.

1.5 Compute the Gini index for Creditworthiness attribute.

Creditworthiness	Buy PC = Yes	Buy PC = No	Gini
high	6	4	0.48
pass	6	4	0.48

Table 3: Gini index for the Creditworthiness attribute.

The results in table 3 gives a weighted average Gini index of 0.48.

1.6 Which attribute is the best?

The Student attribute has the lowest Gini index, and can be considered the best.

2 Classification

2.2 Datasets

Iris

The Iris dataset contains data for classification of different iris plants.

It consists of 150 instances, and 5 variables – all real:

1. Sepal length in cm
2. Sepal width in cm
3. Petal length in cm
4. Petal width in cm
5. Class

The class variable takes on 3 values, evenly distributed with 50 samples each.

Diabetes

The diabetes dataset contains patient data, and classifies them as having tested either positively or negatively for diabetes.

It consists of 768 instances, and has 9 attributes – all real:

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure
4. Triceps skin fold thickness
5. 2-Hour serum insulin
6. Body mass index
7. Diabetes pedigree function
8. Age
9. Class

The class takes on either *positive* or *negative*.

Spambase

The Spambase dataset consists of 4601 documents, in the form of various metadata.

It has 58 distinct attributes, but many of them measure similar things. The 6 "attribute categories" are distributed as follows:

1. 48 continuous real $[0, 100]$ attributes of type word_freq_WORD (percentage of words in the e-mail that match WORD)
2. 6 continuous real $[0, 100]$ attributes of type char_freq_CHAR (percentage of characters in the e-mail that match CHAR)
3. 1 continuous real $[1, ...]$ attribute of type capital_run_length_average (average length of uninterrupted sequences of capital letters)
4. 1 continuous integer $[1, ...]$ attribute of type capital_run_length_longest (length of longest uninterrupted sequence of capital letters)
5. 1 continuous integer $[1, ...]$ attribute of type capital_run_length_total (sum of length of uninterrupted sequences of capital letters)
6. 1 nominal $0, 1$ class attribute of type spam

2.3 Classification

J48

An open source implementation of the proprietary C4.5 algorithm. It has the following important parameters:

binarySplits Whether to use binary splits on nominal attributes when building the tree.

confidenceFactor The smaller the value, the more pruning is done.

```
=== Spambase Confusion Matrix ===
   a   b   <-- classified as
 896  51 |   a = 0
  71 546 |   b = 1

=== Diabetes Confusion Matrix ===
   a   b   <-- classified as
 152  26 |   a = tested_negative
  36  47 |   b = tested_positive

=== IRIS Confusion Matrix ===
  a  b  c   <-- classified as
 15  0  0 |  a = Iris-setosa
  0 19  0 |  b = Iris-versicolor
  0  2 15 |  c = Iris-virginica
```

kNN

Bases classification on the class of the k nearest neighbors in a Euclidian space.

The k parameter is the big one, specifying how many neighbors to consider upon classification. Results go from 85% to 89% on the spambase dataset when upping k from 1 to 3. However, higher values than 5 yield slightly poorer performance.

```
=== Spambase Confusion Matrix ===
   a   b   <-- classified as
 874  73 |   a = 0
  91 526 |   b = 1

=== Diabetes Confusion Matrix ===
   a   b   <-- classified as
 148  30 |   a = tested_negative
  36  47 |   b = tested_positive
```

```
=== IRIS Confusion Matrix ===
  a  b  c   <-- classified as
 15  0  0 |  a = Iris-setosa
  0 19  0 |  b = Iris-versicolor
  0  2 15 |  c = Iris-virginica
```

SMO

The most important parameter in SVMs are arguably the kernel, allowing for different kinds of boundaries to be used to divide the data vectors into classes. The Puk kernel works best out of the ones I tried, achieving an accuracy of 93.7%.

```
=== Spambase Confusion Matrix ===
   a   b   <-- classified as
 912  35 |   a = 0
  63 554 |   b = 1

=== Diabetes Confusion Matrix ===
   a   b   <-- classified as
 161  17 |   a = tested_negative
  37  46 |   b = tested_positive

=== IRIS Confusion Matrix ===
  a  b  c   <-- classified as
 15  0  0 |  a = Iris-setosa
  0 19  0 |  b = Iris-versicolor
  0  2 15 |  c = Iris-virginica
```

2.4 Evaluation

Cross-validation is better than percentage split in many different cases.

While percentage split simply splits the dataset into a training set and a test set, cross-validation does this k times. In this way, the entire dataset is used as both training and testing, and the final benchmark is the average performance.

This is a better approach if the values in the dataset is not evenly/randomly distributed, for instance when it is ordered. In a percentage split on an ordered dataset, the algorithms would train for conditions not matched in the test set.

2.5 Best classifiers

For the Spambase, the SMO with the Puk kernel performs best with 93.7% correctly classified instances. J48 trails right behind with just above 92%, while kNN gets 89% with $k = 3$. On the other two datasets, the results are a bit worse, and more even.

Of the other classifiers, BayesNet – Bayesian networks – reports quite good results on most of the datasets.