Result Report on C4.5 and Naive Bayesian Classifier

Zhengzhe Yang

2018-03-05

Team mates: Zhenzhe Yang

Objective of this assignment

In this assignment, we need to implement C4.5 and Naive Bayesian Classifier to classify the *mushroom* dataset. We use categorical values only.

Implementation for C4.5

To me, the hardest part is how to split the dataset while recursion. I struggled little bit and create a data structure called *data*, which holds all the tuples in the datasets. When splitting, I will create a new data, with selected tuples only. There are two kinds of constructors for *data*, one is the default one, and we can use this to create any *data* we want, and the other takes in a file name, which will create the *data* with all the tuples in that file along with the labels. It will have a *string* called *decision*, to keep the record of splitting, and thus we can know that what leads to this splitting. This decision is used to create the nodes in the decision tree. *node* is another data structure that is used by the decision tree. We then can classify any dataset using this tree by simply going through the nodes, until we encounter a label.

The other tricky part of my implementation is that, if some values do not appear in certain attributes but appear in the test dataset, there's no way we can classify that. This is different from the example in the slides, because the values of ages can in no way appear in the attribute of, say, salaries. Thus, we need to take care of this by creating a set of all the unique values in the training set. When splitting, we will split based on this unique set, which is stored in the *data*. If there's am empty split partition, we can assign a label to it with the majority vote.

Implementation for Naive Bayesian

In my implementation, I again utilized the *data* structure. When training my Bayesian classifier, I simply called two methods. First, we need to obtain the priors, that is the probability of class 1 and class 2 in the labels. This should be very straightforward. Then, we need to obtain posteriors, which requires us to traverse each column of the dataset and count the occurrences based on the class label. We need to be careful when counting because we have to obey the Bayesian theorems. The stored trained model is a hash map, in which the key is the class label, and the value is an arraylist of hash map. In these arraylists, the hash map contains each attribute and their corresponding posteriors. When testing the accuracy, we can calculate the posteriors and for each class, we classify it to the one with larger probabilities.

Results

Please see the output file. For the mushroom dataset, the accuracy for the C4.5 is 100%. The accuracy for the Naive Bayesian Classifier is 99.86%. The reason why the latter is lower is because it is naive, and thus it doesn't take the correlations between attributes into account. It treats all the attributes independently, thus resulting in a lower accuracy.

Refelection

In this assignment, I think I used rather complicated data structures. I believe there are other ways to avoid doing this, because it can be difficult if there is bug in my code, and with such complicated data structure, it can be hard to locate the bug. In the future, I think I should spend more time simplifying my code.