EECS 440 - Programming Assignment 2 – Write-up

Group 14 – David Kerrigan, Kyle Pham, Gary Yao

For this programming assignment, we were required to implement the Naïve Bayes and Logistic Regression models in class.

Our implementation should work with both nominal and continuous attributes, and the way it works is similar to what was discussed in class. For a nominal attribute, we separate it into two branches such that the information gain is maximized. To do this, we test the different bipartitions of the data attributes and check all the cases for splitting, and choose the best that maximizes the information gain.

Our implementation will also use cross validation using 5-fold stratified cross validation. We separate the examples into 5 folds, and then use 4 folds for training and the other fold for testing. Doing this 5 times, we get 5 different performance measures. We then finally average them to get our average performance measure. To ensure repeatability, we set the random seed for the PRNG to 12345, and then produce decision trees on each fold.

We also allow the users to have some certain options for our program:

* Option 1 is the path to the data. If this is “/a/b/someproblem” then you will load

“/a/b/someproblem.names” and “/a/b/someproblem.data”.

* Option 2 is a 0/1 option. If 0, use cross validation. If 1, run the algorithm on the full sample.
* Option 3 is a nonnegative integer that sets the maximum depth of the tree (the number of tests on any path from root to leaf). If this value is zero, you should grow the full tree. If this value is positive, grow the tree to the given value. Note that if this value is too large, it will have the same effect as when the option is zero.
* Option 4 is a 0/1 option. If 0, use information gain as the split criterion. If 1, use gain ratio.

Here is our result:

# Results

## a)

First, we set the depth of the tree to 1. We observe that the cross validation accuracy is:

We have a table for the results for the datasets:

|  |  |
| --- | --- |
| Dataset | CV Accuracy |
| Voting | 55.45% |
| Volcanoes | 67.23% |
| Spam | 62.54% |

## b)

Secondly, we look at first test picked by the tree for the datasets “spam” and “voting”.

We found that the first test for “voting” is “Supply-Our-Soldiers-Act-of-2011”. This is the most predictive of the label, since the information gain is the greatest if we choose to split based on this attribute.

We found that the first test for “spam” is checking whether the geographical distance is bigger than 2.6. This test is the most predictive for the label, with the information gain being the greatest if we choose to split based on this attribute and value.

## c)

Thirdly, for “volcanoes” and “spam”, we plot the CV accuracy as the depth of the tree is increased. On the x-axis, we choose depth values to test so there are at least five evenly spaced points.

## d)

Next, we pick 3 different depth values and look at the CV accuracies change for gain and gain ratio for the different problems for these values.

We choose depth 1, 2, and 3.

## e)

Then, we compare the CV accuracies and the accuracy on the full sample for depths 1 and 2.

We have a table for the CV accuracies and the accuracy on the full sample for depths 1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset | Depth 1 | | Depth 2 | |
| CV accuracies | Accuracy on the full sample | CV accuracies | Accuracy on the full sample |
| Voting | 55.45% | 55.45% | 76.81% | 86.82% |
| Volcanoes | 67.23% | 67.23% | 82.35% | 84.36% |
| Spam | 62.54% | 62.54% | 81.22% | 85.45% |

# Other observations

## Time complexity

The time complexity of our tree seem pretty long, especially for complex datasets like “volcanoes” or “spam”. For a less complex dataset such as “voting”, it works quite fast.

By slow, we mean it takes time (50-70 minutes) even to do create tree of depth 2 for both of these datasets.

## Complexity of the code

The complexity of the code was worse than we expected, but we tried our best to optimize and use good software craftsmanship.

## Observations on the datasets

The dataset “spam”, even though more complicated than “voting”, seems to do well with a tree of depth 1.

The dataset “voting”, even though the least complex does not seem to do well with our decision tree of depth 1. However, using a different depth produces a really good result without having to spend much time for training compared to the other two datasets.

“Volcanoes” seem to be a pretty complicated dataset with a lot of features and data points. It takes our model a long time to train on the dataset.