**8.1 Briefly outline the major steps of decision tree classification.**

A decision tree can be used to partition class-labeled data. The decision tree algorithm is typically run on training data, then the generalizability is tested on new data. A decision tree consists of internal nodes (attributes tests), branches (outcome of the attribute test), and leaves or terminal nodes (holds the class label for the entire branch). The goal of decision tree classification is to create branches that are as pure as possible (samples that are homogenous on the class label). The decision tree process begins with the full training data set. If all tuples are of the same class, the process stops there. If not, an attribute selection method is chosen to determine which attribute to split on first.

The book covers three attribute selection methods: information gain, gain ratio, and the Gini index. Information gain is based on the decrease in entropy (a measure of purity/homogeneity, 0 = completely homogenous) due to sorting the training data on that node/attribute compared to not being sorted on any attribute at all. Conditional entropy (entropy after sorting on a given attribute) is subtracted from entropy of the sample. The attribute with the highest information gain (greatest difference between conditional entropy and entropy of the whole sample) becomes the first node. The process is repeated within each branch.

The information gain attribute selection method is biased toward attributes with many values. The more values there are on an attribute, the more likely each branch is to be homogenous/pure. Gain ratio is used to address this bias. The gain ratio is calculated by dividing the information gain of a given attribute by splitinfo, a measure of variation in attribute value distribution. Gain ratio improves on information gain by considering the number of branches created if the data are split on that attribute and favoring attributes that results in fewer branches (less likely to overfit). Once an attribute has been selected

The gini index performs only binary splits. It is a measure of impurity, indicating the variation within each node on the class label. The attribute with the lowest gini score is selected. Once an attribute has been selected, the process is repeated within each branch.

Partitioning stops when all tuples belong to the same class, when the data have been split on all attributes, or when there are no tuples for a given branch.

Once the tree is complete on the training data, it may still need to be “pruned.” Pruning is necessary when the decision tree has been overfit to the training data, which could mean the tree is either too large (too many branches) and/or does not perform well on new data. To prune the tree, one may compare a goodness of fit measure (e.g. the attribute selection measures described above) with and without a given branch. If the measure improves, the branch is removed. Pruning generally begins at the bottom of the tree and moves up.

**8.6 Why is naive Bayesian classification called “naive”? Briefly outline the major ideas of**

**naive Bayesian classification.**

Naïve Bayesian classification is used to estimate the probability of class membership given X (a tuple or set of attribute values). This is the posterior probability of class membership.

Naïve Bayesian classification is called “naïve” because of the assumption of class-conditional independence used to reduce the computational burden of calculating the posterior probability of the evidence (set of attribute values for tuple X) give the hypothesis is true (the class label). Class conditional independence assumes that there is no dependence between attribute values, so that the posterior probability of each attribute value given the class label may be calculated independently. Once the posterior probability for each attribute value is calculated, the product of all attribute value posterior probabilities is used in place of the posterior probability of X given the class label in the Bayes theorem.

If there are no training tuples with an attribute value within a given class, the posterior probability of that attribute value will be estimated as zero. Using the naïve Bayesian classifier, the zero probability will cancel the effect of the other posterior probabilities. The solution for this problem is the Laplacian correction. 1 is added to each attribute value count to avoid zero probabilities.

The class label for which the posterior probability of X multiplied by the prior probability of the class label is highest (maximum) is assigned to that tuple. This is the maximum posteriori hypothesis.

**8.12 The data tuples of Figure 8.25 are sorted by decreasing probability value, as returned by**

**a classifier. For each tuple, compute the values for the number of true positives .*TP*/,**

**false positives .*FP*/, true negatives .*TN*/, and false negatives .*FN*/. Compute the true**

**positive rate .*TPR*/ and false positive rate .*FPR*/. Plot the ROC curve for the data.**

Angela

ROC curve = Receiver Operating Characteristic curve. It compares classifiers.

Positive tuples belong to the main class of interest. Negative don’t belong. In ML, they’re called positive samples.

ROC shows tradeoff between True Positive Rate and False Positive Rate.

The area under the ROC curve is the accuracy of the model.

P, N = number of positive tuples. Number of negative tuples.

P ′, N ′ = number of tuples that were **labeled** as positive, (…) as negative.

Cost/benefit = risks/gains.

We can assign different weights to false positive and false negative costs.

Confusion matrix design:

|  |  |  |
| --- | --- | --- |
| Actual | Classifier Decided | |
| Classes | Buys\_computer = yes | Buys\_computer = no |
| Buys\_computer=yes |  | Falses, ideally close to 0 |
| Buys\_computer=no | Falses, ideally close to 0 |  |

The True Positive Rate and the False Positive Rate both go up if the probability of the classification (the threshold) goes down.

Tuple 1 has the highest probability score of 0.95. We take that score as our threshold. So in PT(t) and FP(t), we’ll use t=0.95.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tuple # | Class | Prob | TP | FP | TN | FN | TPR | FPR |
| 1 | P | 0.95 | 1 | 0 | 5 | 4 | 0.2 | 0 |
| 2 | N | 0.85 | 1 | 1 | 4 | 4 | 0.2 | 0.2 |
| 3 | P | 0.78 | 2 | 1 | 4 | 3 | 0.4 | 0.2 |
| 4 | P | 0.66 | 3 | 1 | 4 | 2 | 0.6 | 0.2 |
| 5 | N | 0.6 | 3 | 2 | 3 | 2 | 0.6 | 0.4 |
| 6 | P | 0.55 | 4 | 2 | 3 | 1 | 0.8 | 0.4 |
| 7 | N | 0.53 | 4 | 3 | 2 | 1 | 0.8 | 0.6 |
| 8 | N | 0.52 | 4 | 4 | 1 | 1 | 0.8 | 0.8 |
| 9 | N | 0.51 | 4 | 5 | 0 | 1 | 0.8 | 1 |
| 10 | P | 0.4 | 5 | 5 | 0 | 0 | 1 | 1 |

**8.14 Suppose that we want to *select between two prediction models*, *M*1 and *M*2. We have**

**performed 10 rounds of 10-fold cross-validation on each model, where the same data**

**partitioning in round *i* is used for both *M*1 and *M*2. The error rates obtained for*M*1 are**

**30.5, 32.2, 20.7, 20.6, 31.0, 41.0, 27.7, 26.0, 21.5, 26.0. The error rates for *M*2 are 22.4,**

**14.5, 22.4, 19.6, 20.7, 20.4, 22.1, 19.4, 16.2, 35.0. Comment on whether one model is**

**significantly better than the other considering a significance level of 1%.**

|  |  |  |
| --- | --- | --- |
|  | Error Rates | |
| **Round #** | **M1** | **M2** |
| 1 | 30.5 | 22.4 |
| 2 | 32.2 | 14.5 |
| 3 | 20.7 | 22.4 |
| 4 | 20.6 | 19.6 |
| 5 | 31 | 20.7 |
| 6 | 41 | 20.4 |
| 7 | 27.7 | 22.1 |
| 8 | 26 | 19.4 |
| 9 | 21.5 | 16.2 |
| 10 | 26 | 35 |
| **mean error rate** | **27.72** | **21.27** |

Variance = (1/k) \*

Variance = 68.1225

t = 2.47123716

Since the t value of 2.47123716 is < the value of 2.821 (and 2.821 is the z value in the t distribution table corresponding with a degree of freedom of 9 and a significance level of 1%), we accept the null hypothesis, and we can declare that in 99% of cases, any difference between M1 and M2’s means is merely because of chance, not because of statistical significance.