**HW 5**

**INFO 523**

**Matt Miller**

**Kai Blumberg**

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

A decision tree classification algorithm starts with a single node, the root which contains all the training data tuples.

At the node, if all the tuples are of the same class, then such node will become a leaf with that class label.

Otherwise, it will determine a splitting criterion by calling an attribute selection method, which could involve using a heuristic or a statistical measure such information gain or gini index to optimally separate the tuples into classes.

The node is assigned a splitting criterion, which will bifurcate incoming data by testing it at that node and passing the data tuples along to the appropriate branch. This is done such that there is a branch for each possible outcome of the splitting criterion. This partitioning could happen three different ways. For discrete splitting attributes, the algorithm makes a branch for each possible outcome of the splitting attribute. For a continuous splitting attribute, it could make two branches one for greater than or equal to and one for less than a given split point. For a discrete splitting attribute (A), for which a binary tree must be produced, then the test is of the form A ∈ SA, where SA is the splitting subset for A. If a given tuple has value aj of A and if aj ∈ SA, then the test at the node is satisfied.

The algorithm then recurses back to the starting step one where we have a node. Recursing to create a full decision tree for the tuples at each partitioning step.

The algorithm will stop when subject to one of the following conditions. In the simplest case, if all tuples at a given node belong to the same class, transform that node into a leaf, labeled with that class. If there are no more attributes left to create more partitions, then majority voting can be used to convert the given node into a leaf, labeled with the most common class among the tuples. Otherwise, if there are no tuples for a given branch, a leaf is created with the majority class from the parent node.

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

The classification is called “naive” because it makes the assumption that all the variables are conditionally independent from one another. There might be some dependency relations between them, but when a naive bayesian classification is used we treat all variables as conditionally independent.

The idea is to classify data by maximizing P(**X**|Ci)P(Ci) (where i is an index of the class) using Bayes' theorem of posterior probability. Given a set of m classes, C1,C2,...Cm, as well as a set of unknown data tuples, represented by an n-dimensional vector, X = (x1,x2,...,xn) which are the n measurements made on the tuple from n attributes, denoted A1,A2,..,An.

Using Bayes theorem, the classifier calculates the posterior probability of each class

conditioned on **X**, where **X** is assigned the class label of the class with the maximum posterior probability conditioned on **X**, in order to maximize P(**X**|Ci) = P(**X**|Ci)P(Ci)/P(**X**). Since

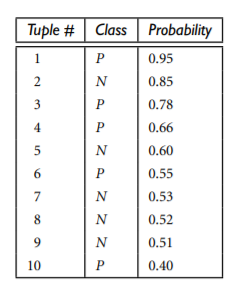
P(**X)** is constant for all classes, that means that only P(**X**|Ci) P(Ci) needs to be maximized.

If we don’t know the class prior probabilities then we can assume that the classes are equally likely (P(C1) = P(C2) …) and we can maximize P(**X**|Ci). Otherwise, we have to maximize P(**X**|Ci) P(Ci). We can estimate the the class prior probabilities with P(Ci) = si/s, where s is the total number of training tuples and si is the number of training tuples of class Ci.

To reduce computation while evaluating P(**X**|Ci), we make the “naive” assumption that classes are conditionally independent, meaning the values of the values of the attributes are also independent of each other. “Getting rid” so to speak of any dependence relationships among the attributes.

For a categorical attributes P(xk|Ci) is equal to the number of training tuples in Ci that have xk as the value for that attribute, divided by the total number of training tuples in Ci. For continuous attributes we calculate P(xk|Ci) from a Gaussian density function.

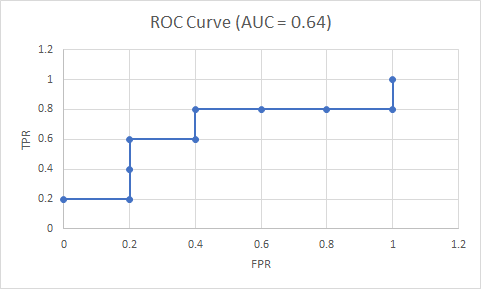
**8.12 The data tuples of Figure 8.25 are sorted by decreasing probability value, as returned by a classifier. For each tuple, compute the values for the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). Compute the true positive rate (TPR) and false positive rate (FPR). Plot the ROC curve for the data.**

****

FPR = FP/N

TPR = TP/P

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | **0.60** | **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** | **1.0** |
| **10** | **P** | **0.40** | **5** | **5** | **0** | **0** | **1.0** | **1.0** |

****

**8.14 Suppose that we want to select between two prediction models, M1 and M2. 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 M1 and M2. The error rates obtained for M1 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 M2 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%.**

Hypothesis testing

Test statistic = paired T-test (since the same data from partition i is used for M1 and M2)

H0: 𝜇\_M1 - 𝜇\_M2 = 0 (there is no difference in means between M1 and M2)

HA: 𝜇\_M1 - 𝜇\_M2 != 0 (there is a difference in means between M1 and M2)

alpha = 0.01

T = 𝜇\_diffs / s\_diffs / sqrt(n),

𝜇\_diffs = mean of the differences in error between M1 and M2 = 6.45,

s\_diffs = standard deviation of the differences in error between M1 and M2 = 8.7

N = # of samples = 10

T = 2.3444

Cutoff T-score for DF = 9 and alpha = 0.01 = 3.25. Since -3.25 < 2.3444 < 3.25, we fail to reject the null hypothesis (or accept the null hypothesis, however you want to phrase it), there is not a significant difference in error rates between models M1 and M2.