### **Data Mining I: Basic Methods and Techniques**

***Laboratory Assignment #2:***

1. Use the following learning schemes to analyze the iris data (in iris.arff):

|  |  |
| --- | --- |
| ZeroR | Weka.classifiers.ZeroR |
| OneR | - Weka.classifiers.OneR |
| Decision table | - Weka.classifiers.DecisionTable |
| C4.5 | - Weka.classifiers.j48.J48 |

* Do the decisions/model produced by the classifiers make sense to you? Why?
* How did each one of the methods perform? We will cover the evaluation techniques later in the class – for now you can choose common sense or one of the techniques that Weka presents with the model.
* Which method provided you with the most/least knowledge (incite into your data set/rules/patterns) and why?

Not all do. The OneR, Decision table, and C4.5 classifiers produce output rules that make sense in terms of variety of class type. Not coincidentally, all three produce similar classifier models and confusion matrices. Instead, the ZeroR classifier outputs a classifier model that makes no sense (“ZeroR predicts class value: Iris-setosa”) and a confusion matrix that shows exactly the same number of class members per each flower type.

If by performance we mean amount of time taken to build the results, all algorithms came very close to 0, probably because the input is only 150 instances. If, instead, by performance we mean the number of correct answers, the C4.5 algorithm yielded the highest value for accuracy at 96%. The OneR and Decision Table algorithms came close with 92% and 92.6% accuracy. However, the ZeroR algorithm produced only 1/3 correct answers.

The highest level of feedback rules were provided by the C4.5 tree, showing tree size of 9 and 5 leaves. This information seems the most detailed set of rules to me. By the same method, the OneR algorithm returned a decently informative tree with the following simple rule:

*petalwidth:*

*< 0.8 -> Iris-setosa*

*< 1.75 -> Iris-versicolor*

*>= 1.75 -> Iris-virginica*

DecisionTable resulted in three rules but I can’t quite understand what those are…. The feedback under “Classfier Model” seemed to describe how the algorithm performed:

*Number of training instances: 150*

*Number of Rules : 3*

*Non matches covered by Majority class.*

*Best first.*

*Start set: no attributes*

*Search direction: forward*

*Stale search after 5 node expansions*

*Total number of subsets evaluated: 12*

*Merit of best subset found: 96*

*Evaluation (for feature selection): CV (leave one out)*

*Feature set: 4,5*

Finally, the ZeroR algorithm returned basically wrong knowledge: *“ZeroR predicts class value: Iris-setosa”*

1. Use the J48 Decision tree-learning scheme to analyze Weather.arff (this data set come with the Weka installation) data set. Make predictions for the ‘temperature’ attribute.

* Did you get an error? The method only performs on nominal class data – make sure you apply the DiscretizeFilter (unsupervised Discretize filter – you can choose the number of bins) before applying the learning method or use Weather\_nominal data set.
* Analyze the output of the model that learned the discretized attribute ‘temperature’? What was learned? How does it differ from the model produced on the nominal version of the same attribute?

When I run J48 on the Weather.Nominal data set I get 0 correct answers out of 14, whereas when I use the same algorithm and apply the Discretize filter on the numeric data set, I get 7 correct and 7 wrong answers.

The nominal weather dataset learned the following model:

*J48 pruned tree*

*--------------*

*humidity = high: mild (7.0/3.0)*

*humidity = normal: cool (7.0/3.0)*

*Number of Leaves : 2*

*Size of the tree : 3*

And the output was:

*=== Confusion Matrix ===*

*a b c <-- classified as*

*0 4 0 | a = hot*

*4 0 2 | b = mild*

*0 4 0 | c = cool*

Whereas the numeric dataset (with filter applied) learned the following:

*=== Classifier model (full training set) ===*

*J48 pruned tree*

*------------------*

*: yes (14.0/5.0)*

*Number of Leaves : 1*

*Size of the tree : 1*

And output:

*=== Confusion Matrix ===*

*a b <-- classified as*

*7 2 | a = yes*

*5 0 | b = no*

1. Use the J48 Decision tree learning scheme to analyze the bolts data (bolts.arff without the TIME attribute):

* The dataset describes the time needed by a machine to produce and count 20 bolts. (More details can be found in the file containing the dataset.)
* Analyze the model produced. What adjustments (if you were to make any) would have the greatest effect on the time to count 20 bolts (i.e. what is the most important/selective attribute/value pair in the tree)?
* According to each classifier, how would you adjust the machine to get the shortest time to count 20 bolts?

The most important attribute is be the speed setting that controls the speed of rotation (SPEED1) because it lies at the root (top) of the output decision tree.

*=== Classifier model (full training set) ===*

*J48 pruned tree*

*------------------*

*SPEED1 <= 4*

*| NUMBER2 <= 1*

*| | RUN <= 31: '(-inf-15.522]' (12.0/1.0)*

*| | RUN > 31*

*| | | RUN <= 36: '(15.522-23.724]' (2.0)*

*| | | RUN > 36: '(23.724-31.926]' (2.0/1.0)*

*| NUMBER2 > 1: '(15.522-23.724]' (8.0/2.0)*

*SPEED1 > 4*

*| TOTAL <= 20*

*| | NUMBER2 <= 1*

*| | | SENS <= 8: '(23.724-31.926]' (2.0/1.0)*

*| | | SENS > 8: '(64.734-72.936]' (2.0)*

*| | NUMBER2 > 1*

*| | | RUN <= 26: '(-inf-15.522]' (2.0/1.0)*

*| | | RUN > 26: '(15.522-23.724]' (2.0/1.0)*

*| TOTAL > 20*

*| | RUN <= 23: '(64.734-72.936]' (5.0)*

*| | RUN > 23: '(81.138-inf)' (3.0/1.0)*

According to what the decision tree says, I would have the lowest time by having SPEED1 (speed setting that controls the speed of rotation) < =4, NUMBER2 (the number of bolts to be counted at second speed) <=1, and RUN (order in which the data were collected) <=31.

1. Produce a K-NN model (classifiers/lazy/IBk) for Weather data set. What is the output? How many instances did it classify correctly and how many incorrectly?

* Try changing the parameter K – the number of neighbors. Did that influence the model’s performance?
* Try using different weighting schemes. Did does this change influence the model’s performance?

The output is

=== Summary ===

Correctly Classified Instances 8 57.1429 %

Incorrectly Classified Instances 6 42.8571 %

I increased the K parameter and the following table shows how correct and incorrect results changed:

|  |  |  |
| --- | --- | --- |
| **K Value** | **Correctly Classified Answers** | **Incorrectly Classified Answers** |
| 2 | 2 | 12 |
| 5 | 2 | 12 |
| 10 | 0 | 14 |

The following table answers the last question

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **K Value** | **Test Option** | **Weight** | **Correctly Classified Answers** | **Incorrectly Classified Answers** |
| 1 | Training Set | 1/ Distance | 13 | 1 |
| 1 | Training Set | 1. Distance | 13 | 1 |
| 10 | Training Set | 1/ Distance | 13 | 1 |
| 10 | Training Set | 1. Distance | 12 | 2 |
| 1 | Cross Validation (10) | 1/ Distance | 2 | 12 |
| 1 | Cross Validation (10) | 1 – Distance | 2 | 12 |