**Machine Learning from Data – IDC**

**HW2 – Decision Tree**

**\*\*\* This assignment can be submitted in pairs.**

In this assignment you will implement a Decision Tree algorithm that uses Rules in order to classify. This kind of Decision Tree hold in every node its corresponding rule.

Every rule is building from a set (possibly empty, like in the root node) of condition and a returning value, for example: *(2=5) (3=4) returning value: 1*. The interpretation of this rule is: if in attribute 2 the value is 5 and in attribute 3 the value is 4 the returning value (=class) is 1. If we are looking at a path from the root of the tree to a leaf then the first (the left) condition is the root attribute split and its value and the last condition (the most right) is the attribute split and its value that leading to the leaf.

The 'leaf rules' (rules that store in the leaf nodes) are the ones that uses to classify new instance.

The classification of an instance is done by searching for the most suitable rule. The definition for the most suitable rule is follow by this steps:

1. If an instance meets all conditions for a given rule then its classification will be the rule returning value.
2. If there is no such a rule then you need to find the most suitable one, the one that meets the largest number of consecutive conditions (from the most left condition – meaning the largest path from the root).
3. If there are more than one rule with the largest number from the previous step then classify with the majority of the returning values of those rules.

In addition to building the tree you will implement 2 methods of pruning on the tree:

1. Chi square pruning:

In this method you will use the Chi square test in order to decide whether to prune a branch of the tree or not. The number you should compare to for the 'cancer data' is 15.51. This number comes from the chi squared chart in the row for 8 degrees of freedom (which is the number of attributes in the cancer data minus 1) and the column for 0.95 confidence level. If your chi squared statistic is less than the threshold you prune.

PAY ATTENTION – where you need to perform this test, what you should do if the result is to prune.

1. Rule pruning:

In this method you will check if removing a rule improve the result. In order to do that you will use a validation set. After you complete building the tree you will go over all the rules and check if removing a rule will improve the error on the validation set. You will pick the best rule to remove according to the error on the validation set and remove it from the rule set. You will stop removing rules when there is no improvement.

PAY ATTENTION – for how you loops over the rule, how you remove rules during this loop, how you decide to stop.

In order to do so you need to first install WEKA:

1. See instruction in HW1.

Prepare your Eclipse project:

1. Create a project in eclipse called HomeWork2.
2. Create a package called HomeWork2.
3. Move the DecisionTree.java and MainHW2.java that you downloaded from the Moodle into this package.
4. Add WEKA to the project:
   1. See instruction in HW1.

Your goal is to predict whether a breast cancer tumor has a recurrence based on parameters of the patient and the tumor. In order to do so you will implement the decision tree that describe above. For making your code more readable you will use several mandatory methods. Only in the first 2 methods (classifyInstance, buildClassifier) we supply an input and output signature that you must follow (those methods are override methods, and you must implement them accordingly), and in the rest you can implement the methods according to your discretion (we added input \ output descriptions for your help, but you can change them the way you want):

1. double classifyInstance: Return the classification of the instance.
   1. Input: Instance object.
   2. Output: double number, 0 or 1, represent the classified class.
2. void buildClassifier: Builds a decision tree from the training data. buildClassifier is separated from buildTree in order to allow you to do extra preprocessing before calling buildTree method or post processing after.
   1. Input: Instances object.
3. void buildTree: Builds the decision tree on given data set using either a recursive or queue algorithm.
   1. Input: Instances object (probably the training data set or subset in a recursive method).
4. calcAvgError: Calculate the average on a given instances set (could be the training, test or validation set). The average error is the total number of classification mistakes on the input instances set and divides that by the number of instances in the input set.
   1. Input: Instances object.
   2. Output: Average error (double).
5. calcInfoGain: calculates the information gain of splitting the input data according to the attribute.
   1. Input: Instance object (a subset of the training data), attribute index (int).
   2. Output: The information gain (double).
6. calcEntropy: Calculates the entropy of a random variable where all the probabilities of all of the possible values it can take are given as input.
   1. Input: A set of probabilities.
   2. Output: The entropy (double).
7. calcChiSquare: Calculates the chi square statistic of splitting the data according to this attribute as learned in class.
   1. Input: Instances object (a subset of the training data), attribute index (int).
   2. Output: The chi square score (double).

In addition to the methods describe above you are more than welcome to add more methods to the required ones if it helps you make the code more organized.

Your decision tree should have the ability to use pruning. The 2 pruning methods are describe above. You need to set the pruning method with the setPruningMode method using the enum PruningMode (No pruning is the default method).

As written at the beginning your tree uses rules in order to classify new instance. We provided you the basic architecture of tree that uses rules (see the classes in the DecisionTree.Java file.

You should think how to use the Node, Rule & BasicRule objects, and what fields \ methods to add them.

You will evaluate your tree with the caclAvgError method as describe above and the code output should be like this:

*Decision Tree with No pruning*

*The average train error of the decision tree is XXX*

*The average test error of the decision tree is XXX*

*The amount of rules generated from the tree XXX*

*Decision Tree with Chi pruning*

*The average train error of the decision tree with Chi pruning is XXX*

*The average test error of the decision tree with Chi pruning is XXX*

*The amount of rules generated from the tree XXX*

*Decision Tree with Rule pruning*

*The average train error of the decision tree with Rule pruning is XXX*

*The average test error of the decision tree with Rule pruning is XXX*

*The amount of rules generated from the tree XXX*

You should hand in a DecisionTree.java, MainHW2.java and hw2.txt files which the grader will use to test your implementation. All of these files should be placed in a hw\_2\_##id1##\_##id2##.zip folder with the id of both of the members of the group.

\*\*\* Submitting in groups on Moodle does not work. Please only submit one zip folder per pair