Juan Pablo Vielma
A01203239

Lab Report ID3

For the implementation of the ID3 decision tree, java was used as the coding language. The following description described the classes and methods used to achieve the implementation.

Classes

Attribute: contains name and possible values for an attribute object.

DataNode: contains Attribute and assigned value for a single element of a data entry.

Data: Collection of DataNode objects. Full data entry row.

DataSet: Collection of Data objects. The entire input data.

Main Methods:

-getEntropy(Attribute attr): returns the entropy of the dataset for the given Attribute.

-getGainedValue(Attribute attr): returns gained value of splitting dataset on the given attribute.

-getBestSplit(): returns Attribute with the highest gain value.

-add(Data data): inserts new data into the data set

TreeNode: Decision tree Node, contains other tree nodes and a data subset

Main Methods:

-split(): splits the treeNode data set into two or more subsets based on the highest gained value and creates a new tree node child for every new subset.

Main: Main class for implementation

Main Methods:

-parseInput(): read data and initialize elements;

-output():print decision tree

Implementation

Reading the Input:

While parsing through the input, a new Attribute object is created for each attribute in the file. The system will store all attributes in a list to keep track of them. When parsing through the data, for every value read a dataNode object will be created containing the corresponding attribute and the value. When a whole row is parsed, it will create a Data object with all dataNodes for the current row. Once the whole file is parsed, the system will add all Data objects to a single DataSet object.

Juan Pablo Vielma
A01203239

Creating the Tree:

To create the tree we we used the class TreeNode as the base of the implementation. This class contained a list of child treeNodes, an Attribute assigned to the class and a data set. To expand the tree it uses the function split() which analyzed the data set for attribute with the best gained value. Once the best value is determined, the method creates a new subset for each possible value of the attribute. This subset will be passed on to create a new child tree node for the current node. In case the set is pure and there is no way to split it, a utility value will be assigned to the tree node. Such value will be treated as a leaf node and in case it is reached, it will return the assigned value.

```
public void split(){
    if(!isPure){
        Attribute attr=dataSet.bestSplit();//get Attribute with the biggest gained value
        attribute=attr; //assign attribute to tree node
        DataSet subset;
        if(attr!=null){
            int n=attr.getN();
            for(int i=0; i<n; i++){
                subset=new DataSet();
                for(int j=0; j<dataSet.size(); j++){
                    if(dataSet.get(j). getValueForAttribute(attr).equals(attr.getValue(i))){
                        subset.add(dataSet.get(j));
                        dataSet.delete(j);//remove data ebtry from current node's subset
                    }
                }
                subset.deleteAttribute(attribute);//eliminate Attribute for all entries of new subset
                children[i]=new TreeNode(subset); //create new child node with new subset of data
            }
        }
    }
}
```

DataSet functions:

```
public float getEntropy(Attribute attr){
    float entropy=0;
    int count=0;
    int size=data.size();
    for(int i=0; i<attr.getN(); i++){
        for(int j=0; j<data.size(); i++){
            if(attr.getValue(i).equals(data.get(j).getValueForAttribute(attr)))count++;
        }
        if(size!=0)
            entropy=entropy-((count/size)*log2(count/size));
        count=0;
    }
    return entropy;
}
```

```
public float gainedValue(Attribute attr){
    DataSet[] subsets= new DataSet[10];
    int size=data.size();
    float entropy=getEntropy(goal);
    float gainedValue=entropy;
    for(int i=0; i<attr.getN(); i++){
        subsets[i]=new DataSet();
        for(int j=0; j<data.size(); i++){
            if(attr.getValue(i).equals(data.get(j).getValueForAttribute(attr)))subsets[i].add(data.get(j));
        }
        gainedValue=gainedValue-((subsets[i].size()/size)*subsets[i].getEntropy(goal));
    }
    return gainedValue;
}
public Attribute bestSplit(){
    float max, aux;
    Attribute attr=null;
    max=0;
    for(int i=0; i<attributes.size()-1; i++){
        aux=gainedValue(attributes.get(i));
        if(aux>max){
            max=aux;
            attr=attributes.get(i);
        }
    }
    return attr;
}
```

Printing the Output:

Recursion will be used to print the tree. For each tree node found, it will print it's name and the value that connects to the corresponding child node. whenever a leaf node is reached the system will print "ANSWER: " and the current tree node value.

```
public static void output(TreeNode root){
    if(root.isLeaf()){
        System.out.println("ANSWER: "+root.getValue());
    }else{
        for(int i=0; i<root.attribute.getN(); i++){//iterar sobre los posibles valores de cada atributo
            System.out.println(root.attribute.getName()+": "+root.attribute.getValue(i));
            output(root.getChild(i));
        }
    }
}
```