****

**Name Moses Tanaka**

**Surname Chipanda**

**Reg Number M192246**

**Programme Information system**

**Course Code Data mining**

**Module**

**Year and Level 4.1 (2022)**

1. **Hunt’s Algorithm**

* Hunt's algorithm grows a decision tree in a recursive fashion by partitioning the training data into successively into subsets.
* Let Dt be the set of training data that reach a node ‘**t’**. The general recursive procedure is defined as:

1. If Dt contains records that belong the same class yt, then t is a leaf node labeled as y
2. If Dt is an empty set, then t is a leaf node labeled by the default class, yd
3. If Dt contains records that belong to more than one class, use an attribute test to split the data into smaller subsets.

* It recursively applies the procedure to each subset until all the records in the subset belong to the same class.
* The Hunt's algorithm assumes that each combination of attribute sets has a unique class label during the procedure.
* If all the records associated with Dt have identical attribute values except for the class label, then it is not possible to split these records any In this case, the node is declared a leaf node with the same class label as the majority class of training records associated with this node.

Decision tree is grown in a recursive fashion

Dt training records

T mode

Y class label (y = {y1, y2, y3,….}

Algorithm

Step 1

If all records inn Dt belong to the same class, then t leaf node labelled as Yt

Step 2

If Dt contains records that belongs to more than one class an attribute test condition is selected to partition the records into smaller subjects.

Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T.id | Home owner | Marital status | Annual income | Default borrower |
| 1 | Yes | Single | 125 000 | No |
| 2 | No | Married | 100 000 | No |
| 3 | No | Single | 70 000 | No |
| 4 | Yes | Married | 120 000 | No |
| 5 | No | Divorced | 95 000 | Yes |
| 6 | No | Married | 60 000 | No |
| 7 | Yes | Divorced | 220 000 | No |
| 8 | No | Single | 85 000 | Yes |
| 9 | No | Married | 75 000 | No |
| 10 | No | Single | 90 000 | Yes |

(a) Defaulted borrower = No

(b)

Home owner

Yes No

D.B = No D.B = No

(c) Home Owner

Yes No

D.B= No Marital Status

Married Single/Divorced

D.B = No D.B=Yes

(d)

Home Owner

Yes No

D.B= No Marital Status

Married Single/Divorced

D.B = No Annual Income

>=70 000 <70 000

D.B = No D.B = Yes

**Bi. Gini impurity index**

What is Gini Impurity?

The Gini impurity measure is one of the methods used in decision tree algorithms to decide the optimal split from a root node, and subsequent splits. It is the most popular and the easiest way to split a decision tree and it works only with categorical targets as it only does binary splits.

Gini Impurity is calculated using the formula,

Gini impurity = 1 – Gini

Lower the Gini Impurity, higher is the homogeneity of the node. The Gini Impurity of a pure node(same class) is zero.

To calculate Gini impurity, let's take an example of a dataset that contains 18 students with 8 boys and 10 girls and split them based on performance as shown below.

Split according to their performance in class

A class of 18 students

10 girls and 8 boys

Above average below average

3 boys and 6 girls 5 boys and 4 girls

The calculation of Gini impurity of the above scenario is as follows

**For the above students**

Total number of students is 9

Probability of boys = 3/9

0.33

Probability of girls = 6/9

0.66

Gini impurity of the above average student = 1 – [(0.33)\*(0.33) + (0.66)\*(0.66)]

= 0.45

**For those below average**

Total number of students is 9

Probability of boys = 5/9

0.55

Probability of girls = 4/9

0.44

Gini impurity of the above average student = 1 – [(0.55)\*(0.55) + (0.44)\*(0.44)]

= 0.5

The weighted Gini impurity of the split based on performance in class (9/18)\*0.45+(9/18)\*0.5 = 0.475

In the above calculation, to find the Weighted Gini Impurity of the split (root node), we have used the probability of students in the sub nodes, which is nothing but 9/18 for both "Above average" and "Below average" nodes as both the sub nodes have equal no of students even though the count of boys and girls in each node varies depending on their performance in class.

Following are the steps to split a decision tree using Gini Impurity:

1. Similar to what we did in entropy/Information gain. For each split, individually calculate the Gini Impurity of each child node
2. Calculate the Gini Impurity of each split as the weighted average Gini Impurity of child nodes
3. Select the split with the lowest value of Gini Impurity
4. Until you achieve homogeneous nodes, repeat steps 1-3

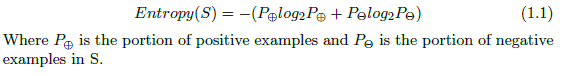
**Bii. Entropy**  
Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content.  
**Definition**: Suppose S is a set of instances, A is an attribute, Sv is the subset of S with A = v, and Values (A) is the set of all possible values of A, then  
Example:

For the set X = {a,a,a,b,b,b,b,b}

Total instances: 8

Instances of b: 5

Instances of a: 3



= -[0.375 \* (-1.415) + 0.625 \* (-0.678)]

=-(-0.53-0.424)

= 0.954

Building Decision Tree using Information Gain  
The essentials:

* Start with all training instances associated with the root node
* Use info gain to choose which attribute to label each node with
* *Note:* No root-to-leaf path should contain the same discrete attribute twice
* Recursively construct each subtree on the subset of training instances that would be classified down that path in the tree.

The border cases:

* If all positive or all negative training instances remain, label that node “yes” or “no” accordingly
* If no attributes remain, label with a majority vote of training instances left at that node
* If no instances remain, label with a majority vote of the parent’s training instances

Example:  
Decision Tree for the following data using Information gain.

Training set: 3 features and 2 classes

|  |  |  |  |
| --- | --- | --- | --- |
| X | Y | Z | C |
| 1 | 1 | 1 | I |
| 1 | 1 | 0 | I |
| 0 | 0 | 1 | II |
| 1 | 0 | 0 | II |

Here, we have 3 features and 2 output classes.  
To build a decision tree using Information gain. We will take each of the feature and calculate the information for each feature.

**Split on attribute X**

X=1

X=0

E parent = 1

GAIN = 1-( ¾) (O.9184) – (1/4) (0) = 0.3112

E child = -(1/3) log2 (1/3) – (2/3) log2 (2/3) = 0.9184

Echild2 = 0

**Split on feature Y**

**Y=1**

**Y=0**

Eparent = 1

GAIN= 1-(1/2)(0)- (1/2)(0) = 1

Echild = 0

Echild2=0

**Split on feature Z**

Z=1

Z=0

E parent = 1

GAIN = 1 – (1/2)(1) – (1/2)(1) = 1

E child = 1

E child2 = 1

From the above images we can see that the information gain is maximum when we make a split on feature Y. So, for the root node best suited feature is feature Y. Now we can see that while splitting the dataset by feature Y, the child contains pure subset of the target variable. So we don’t need to further split the dataset.

The final tree for the above dataset would be look like this:

Y = 1

Y = 0

**iii. Misclassification rate**

Goal: partition with uniform category — pure leaf

Impure node — best prediction is majority value

Minority ratio is misclassification rate

Heuristic: reduce impurity as much as possible

For each attribute, compute weighted average misclassification rate of children

Choose the minimum

0.5

0 0.5 1

c = 1 Misclassification rate is linear

c ∈ {0, 1}

x-axis: fraction of inputs with c = 1

A better impurity function

Misclassification rate is linear

Impurity measure that increases more sharply performs better, empirically