OEP

Aim:-Write a program to calculate information gain for attribute selection.

```
import java.util.*;

/**
 * this class Calculation is to calculate the entropy, varianceImpurity etc. for the Decision Tree
 * @author Jinglin Li (jxl163530, 2021323767)
 */

public class Calculation {

	/**
	 * this function calculate the total entropy, which is -p0 * lg2(p0)-p1*lg2(p2)
	 * @param parsedData
	 * @return
	 */

	public double calcEntropy(List<List<Pair>> parsedData) {
		int[] count = new int[2];
		for (List<Pair> list : parsedData) {
			count[list.get(list.size() - 1).value]++;
		}
		double p0 = (double)count[0] / (double)(count[0] + count[1]);
		double p1 = 1.0 - p0;
		return -p0 * Math.log(p0) / Math.log(2) - p1 * Math.log(p1) / Math.log(2);
	}

	/**
	 * this function calculate the Variance Impurity
	 * @param parsedFile
	 * @return
	 */

	public double calcVarianceImpurity(List<List<Pair>> parsedFile) {
		int[] count = new int[2];
		for (List<Pair> list : parsedFile) {
			count[list.get(list.size() - 1).value]++;
		}
		return (double)(count[0] * count[1]) / (double) (parsedFile.size() * parsedFile.size());
	}

	/**
	 * this function calculate the node's information gain of the all the node giving the parsedData and
subData
	 * @param entropy		the current node's Heuristic Value
	 * @param leftData		the following Data of the current node, when current node's value is 0
	 * @param rightData the following Data of the current node, which current node's value is 1
	 * @param size		the size of the total tree data
	 * @param heuristic
	 * @return
	 */

	public double calcInfoGain(double entropy, List<List<Pair>> leftData, List<List<Pair>> rightData,
			int size, String heuristic) {
		List<Double> subEntropy = new ArrayList<>();
		if (heuristic.equals("Entropy")) {
			subEntropy.add(calcEntropy(leftData));
			subEntropy.add(calcEntropy(rightData));
		}
		else if (heuristic.equals("Variance Impurity")) {
			subEntropy.add(calcVarianceImpurity(leftData));
			subEntropy.add(calcVarianceImpurity(rightData));
```

```
                }
                List<Integer> subSize = new ArrayList<>();
                subSize.add(leftData.size());
                subSize.add(rightData.size());

                double gain = entropy;
                for (int i = 0; i < subEntropy.size(); i++) {
                        gain -= ((double)subSize.get(i) / (double)size) * subEntropy.get(i);
                }
                return gain;
        }
}
```

Output:-

```
1.41        3 Discharge Date
1.3903      2 Admission date
0.51513     8 Years in area
0.50894    10 Woreda of Origin
0.4611      4 Days in Hospital
0.39632    19 Sero status
0.36946     1 Ward
0.31482    24 Discharge Weight
0.28529    12 Admission Weakness
0.16798    26 Discharge Status
0.13321    25 Discharge Spleen
0.12656     6 Age
0.10286    20 Diagnosis
0.07652     7 Type of Residency
0.06963    11  Number of months sick
0.05809    18 TB Coinfection
0.0353     13 Admission Weight
0.03409    15 Body Mass Index
0.02317     9 Region of Origin
0.02284    14 Admission Height
0.01803    23 Bleeding
0.01539    16 Admission Oedema
0.01506    17 Admission Spleen
0.00843    22 Vomiting greater than two days
0.00745     5 Sex
```