KING SAUD UNIVERSITY
COLLEGE OF COMPUTER AND
INFORMATION SCIENCES
Computer Science Department

CSC 361: Artificial Intelligence

Programming Project:

Decision tree

Reem Alsuhaim 443200884
Hissah Ibn Qurmulah 443200791

Instructor
Dr. Nouf Alshunaifi

Table of content:

Details of your implementation/tools	
Programming language	Java(TM) SE Development kit 19.0.2
Libraries used	WEKA ML 3-9-6
Locate and manage dependencies way	Classpath
Additional VM arguments	add-opens java.base/java.lang=ALL-UNNAMED--

Important parts of implementation:

Converting outcome column to nominal:

```
23
24              // Convert the class attribute to nominal (interpreted by the Weka as numeric values.)
25              // for J48 outcome needs to be nominal.
26              if (data.classAttribute().isNumeric()) {
27                  NumericToNominal convert = new NumericToNominal();
28                  convert.setAttributeIndices("last");
29                  convert.setInputFormat(data);
30                  data = Filter.useFilter(data, convert);
31              }
32
33
```

Spliting data into training(70%) and testing(30%) sets & train the tree (by J48 using C4.5 algorithem):

```
33
34              // Split data into training and testing sets 70/30
35              int trainSize = (int) Math.round(data.numInstances() * 0.7);
36              int testSize = data.numInstances() - trainSize;
37              Instances trainData = new Instances(data, 0, trainSize);
38              Instances testData = new Instances(data, trainSize, testSize);
39
40              // Train the decision tree
41              Classifier tree = new J48();
42              tree.buildClassifier(trainData);
```

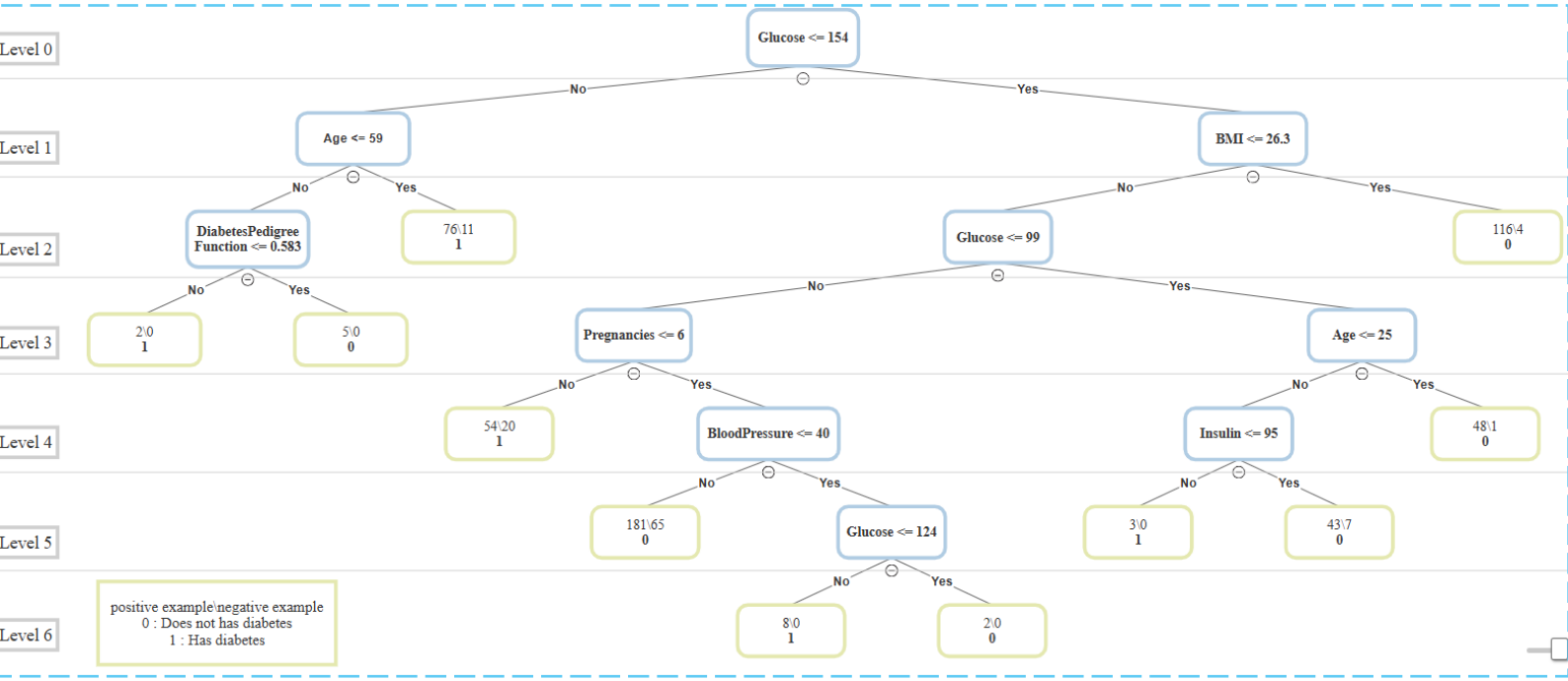
Test The tree using testing set:

```
            // Evaluate the model
            Evaluation eval = new Evaluation(trainData);
            eval.evaluateModel(tree, testData);
```

Get TP , FP , FN , TN to calculate (accuracy , precision , recall):

```
55
56              // Get TP , FP , FN , TN
57              int positiveClassIndex = 1; // "diabetes = 1"
58              double TP = eval.numTruePositives(positiveClassIndex);
59              double FP = eval.numFalsePositives(positiveClassIndex);
60              double FN = eval.numFalseNegatives(positiveClassIndex);
61              double TN = eval.numTrueNegatives(positiveClassIndex);
62
```

The resulting DT:




```
J48 pruned tree
------------------

Glucose <= 154
|   BMI <= 26.3: 0 (116.0/4.0)
|   BMI > 26.3
|   |   Glucose <= 99
|   |   |   Age <= 25: 0 (48.0/1.0)
|   |   |   Age > 25
|   |   |   |   Insulin <= 95: 0 (43.0/7.0)
|   |   |   |   Insulin > 95: 1 (3.0)
|   |   Glucose > 99
|   |   |   Pregnancies <= 6
|   |   |   |   BloodPressure <= 40
|   |   |   |   |   Glucose <= 124: 0 (2.0)
|   |   |   |   |   Glucose > 124: 1 (8.0)
|   |   |   |   BloodPressure > 40: 0 (181.0/65.0)
|   |   |   Pregnancies > 6: 1 (54.0/20.0)
Glucose > 154
|   Age <= 59: 1 (76.0/11.0)
|   Age > 59
|   |   DiabetesPedigreeFunction <= 0.583: 0 (5.0)
|   |   DiabetesPedigreeFunction > 0.583: 1 (2.0)

Number of Leaves  :     11

Size of the tree :      21
```

Experimental results:

.We use J48 classifier In Weka, J48 represents a practical implementation of the C4.5 algorithm, with features including flexible options for tree configuration and management.

The C4.5 algorithm is used to generate a decision tree that classifies data by partitioning it based on information gain or gain ratio. The algorithm starts by selecting the most appropriate attribute for partitioning, then partitions the data into branches based on the values of that attribute and repeats the process on the branches until specific classes are reached. The algorithm supports handling numerical attributes and missing values and prunes the tree to avoid overfitting

.**The results were as follow :**

```
==Results:==
---------------------------------------------------------------

Correctly Classified Instances         176               76.5217 %
Incorrectly Classified Instances        54               23.4783 %
Kappa statistic                          0.4665
Mean absolute error                      0.3353
Root mean squared error                  0.4292
Relative absolute error                 73.9186 %
Root relative squared error             90.3686 %
Total Number of Instances              230

==Confusion Matrix:==
True  Positives (TP): 48.0
False Positives (FP): 23.0
False Negatives (FN): 31.0
True  Negatives (TN): 128.0

==Accuracy:==
Precision: 68%
Recall   : 61%
F1-Score : 64%
---------------------------------------------------------------
```

- The model has correctly classified 76.52% of the instances. This indicates reasonably good performance. While About 23.48% of the predictions were incorrect.
- A Kappa statistic of 0.4665 suggests moderate agreement between the model's predictions and the true labels. While this is better than random guessing Kappa = 0, it's not very strong.
- On average, the absolute difference between predicted and actual values is 0.3353.
- A value of 0.4292 indicates moderate prediction errors.
- The model's total error is 73.92% of the error of a simple baseline model. This is high and indicates that the model isn't significantly better than the baseline.
- The model's squared error is 90.37% of the baseline model's squared error. This is close to the baseline performance, which mean limited improvement over simple predictions.

Accuracy scores (precision and recall):

True Positives (TP): Has diabetes and has been correctly classified.	48
False Positives (FP): Not diabetic and was misclassified as having diabetes.	23
False Negatives (FN): Diabetic and was misclassified as not having diabetes.	31
True Negatives (TN): Not diabetic and has been correctly classified	128
Correctly Classified Instances	176
Incorrectly Classified Instances	54
Total Number of Instances	230

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ predictions} = \frac{176}{230} = 0.765 = 76.5\%$$

Accuracy: It reflects the extent to which the system has succeeded in classifying all tests correctly. Accuracy depends on the percentage of correct cases, whether positive or negative, to the total number of all cases. For our program, the public health percentage was 76.5%, which is a satisfactory percentage.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} = \frac{48}{48 + 23} = 0.676 = 67.6\%$$

Precision: It depends on the ratio of positive cases to the positive of the situations whether they are correct or incorrect. For our program, the Precision reached 67.6%, as it depends on the cases that it correctly classifies as positive cases.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} = \frac{48}{48 + 31} = 0.608 = 60.8\%$$

Recall: It is based on the ratio of true positive cases to the total number of true positive and false negative cases. For our program, the recall rate was 60.8%. It calculates the percentage of positive instances that our program detected out of all the actual positive cases.

$$\text{F1} = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = 2\frac{0.676 \cdot 0.608}{0.676 + 0.608} = 0.640 = 64.0\%$$

F1-Score: is a metric used to evaluate the performance of a classification model, especially when dealing with imbalanced datasets. It is the harmonic mean of Precision and Recall, providing a single score that balances the trade-off between the two. For our program F1-Score was 64.0% .