

Experiment 8

Aim: Use Weka to implement the classifier-Decision Tree Algorithm

Theory:

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset.

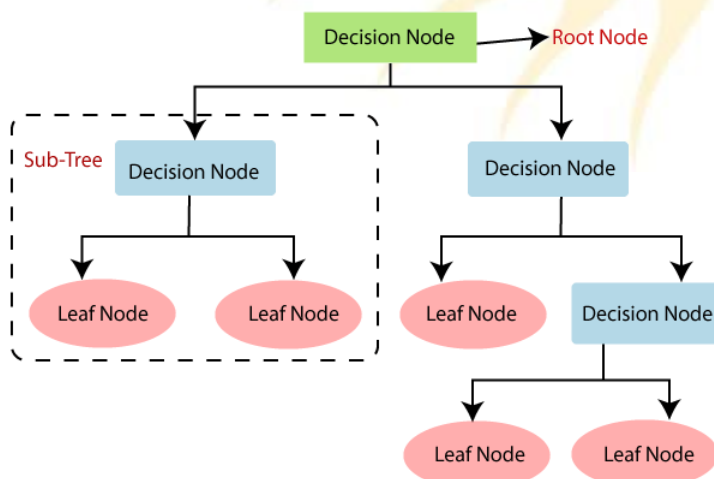
It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

Below diagram explains the general structure of a decision tree:



Advantages:

1. It is easy to grasp because it follows a constant method that somebody follows whereas creating any call-in real-life.
2. It is terribly helpful for the resolution of decision-related issues.
3. It helps to place confidence in all the attainable outcomes for a haul.
4. There is less demand for knowledge cleansing compared to alternative algorithms.

Disadvantages:

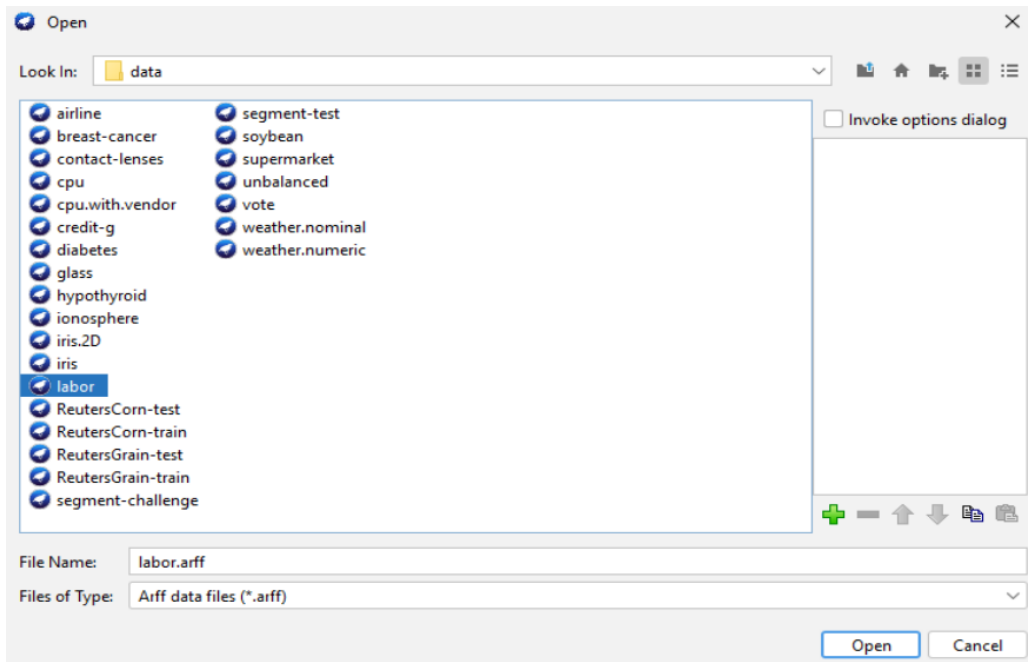
1. The decision tree contains legion layers, which makes it advanced.
2. It may have an associate overfitting issue, which might be resolved exploitation the Random Forest formula.
3. For a lot of category labels, the process quality of the choice tree could increase.

Output:

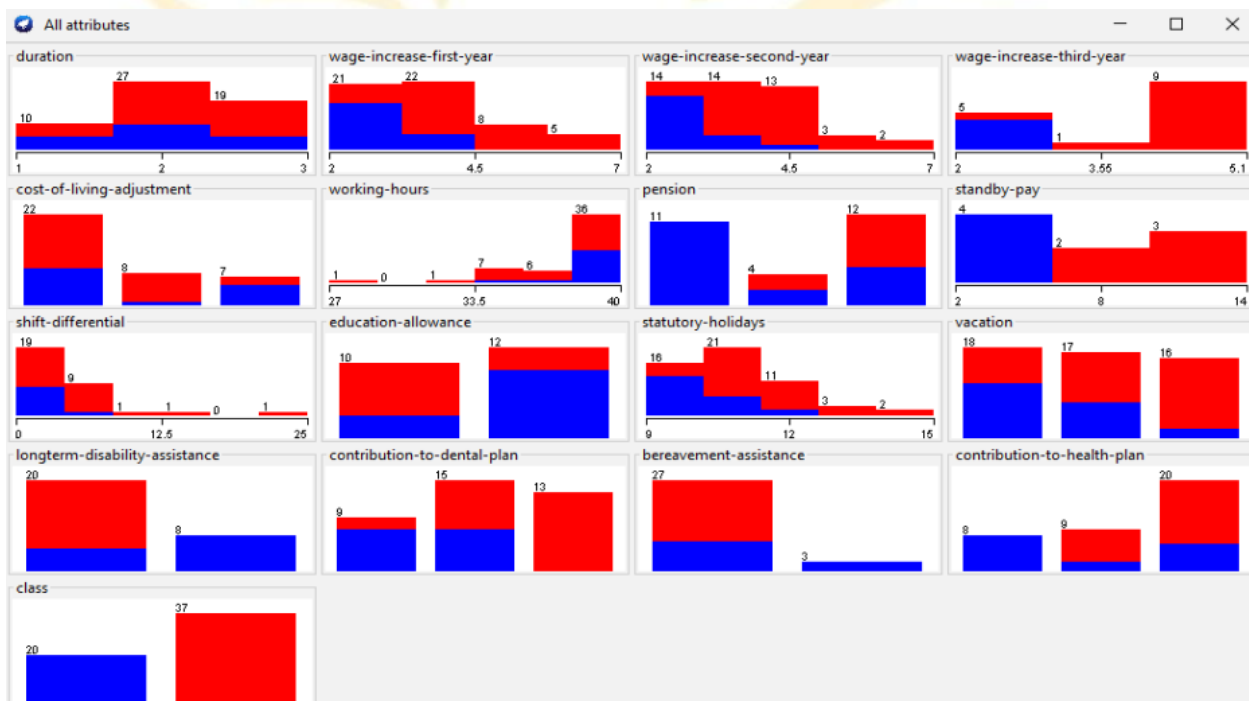
Step 1: Open Weka GUI & Select the “Explorer” option.



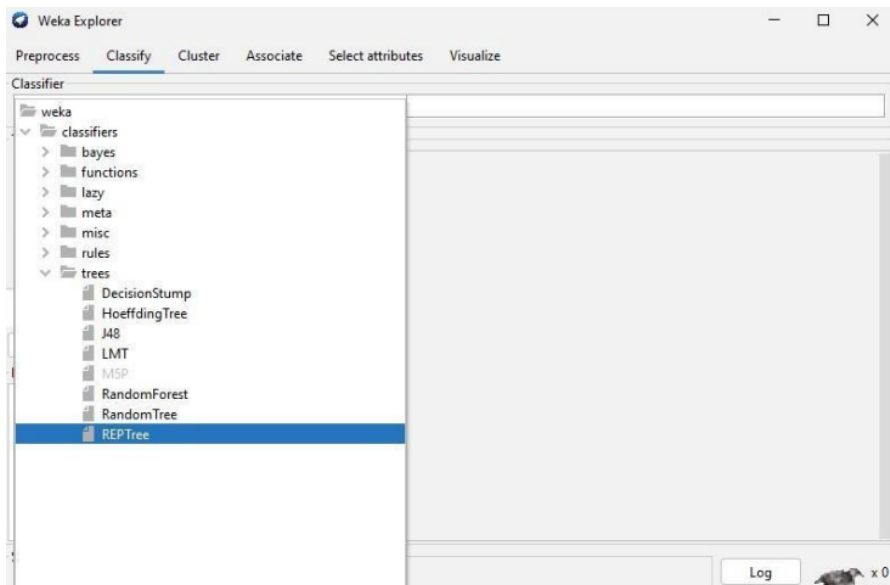
Step 2: Select “Open file” and choose your dataset.



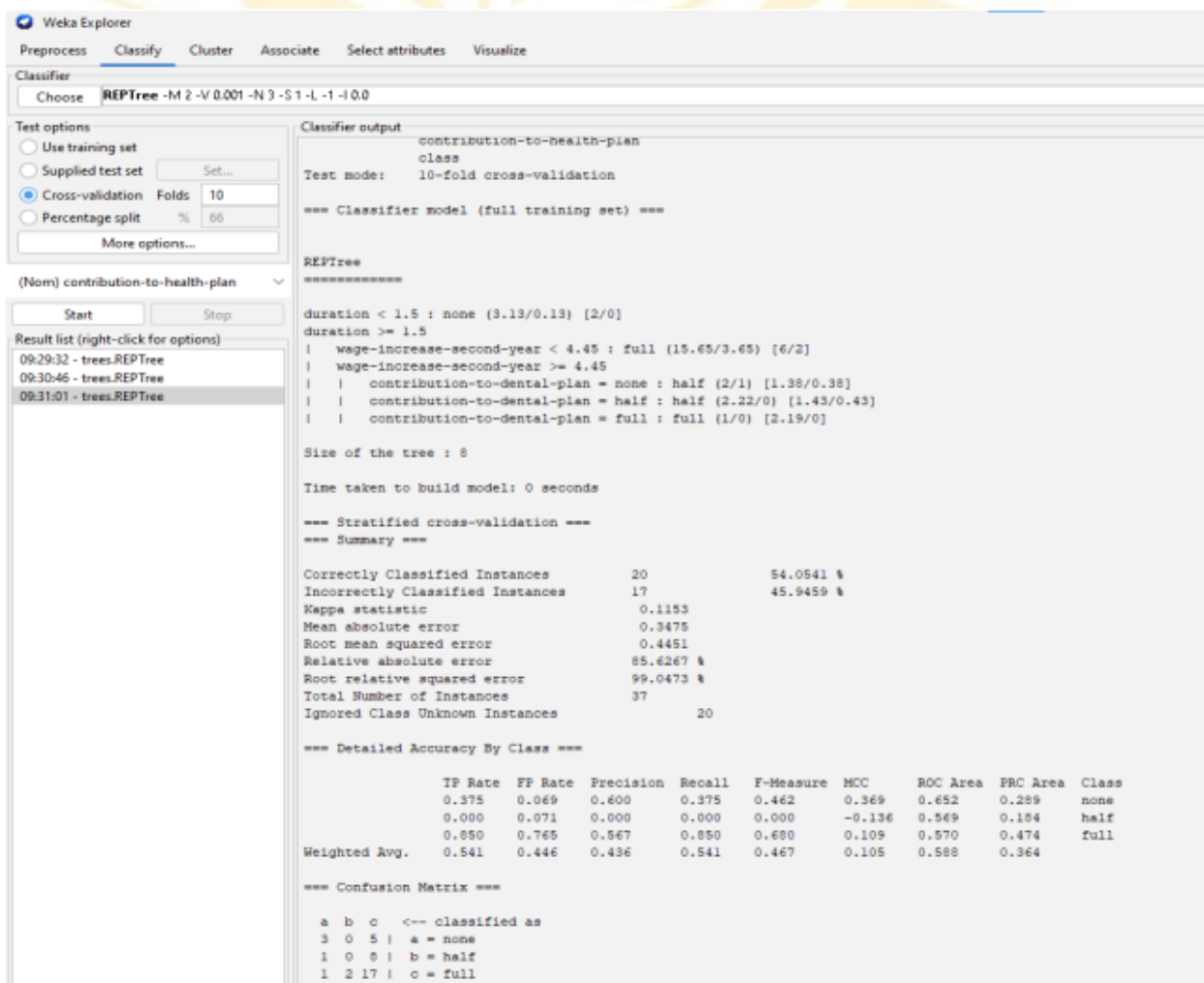
Step 3: You can even view all the plots together if you click on the “Visualize All” button



Step 4: Click on the “Classify” tab on the top and then Click the “Choose” button. From the drop-down list, select “trees” which will open all the tree algorithms select the “RepTree” decision tree.



Step 5: Press the “Start” button to start the classification



The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The 'Classifier' dropdown is set to 'REPTree - M 2 -V 0.001 -N'. The 'Test options' section has 'Cross-validation' selected with 'Folds' set to 10. The '(Nom) class' dropdown is set to 'Start'. The 'Result list' shows '15:14:44 - trees.REPTree'. The 'Status' bar at the bottom indicates 'OK'.

Overlaid on the Weka Explorer is the 'weka.gui.GenericObjectEditor' dialog box. The 'About' tab is active, displaying 'Fast decision tree learner.' with 'More' and 'Capabilities' buttons. The configuration parameters are as follows:

Parameter	Value
batchSize	100
debug	False
doNotCheckCapabilities	False
initialCount	0.0
maxDepth	-1
minNum	2.0
minVarianceProp	0.001
noPruning	False
numDecimalPlaces	2
numFolds	3

At the bottom of the dialog are buttons for 'Open...', 'Save...', 'OK', and 'Cancel'.

Weka Explorer

Preprocess **Classify** Cluster Associate Select attributes Visualize

Classifier

Choose **REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0**

Test options

☐ Use training set

☐ Supplied test set Set...

☒ Cross-validation Folds: **10**

☐ Percentage split %: **66**

More options...

(Nom) contribution-to-health-plan ▾

Start Stop

Result list (right-click for options)

- 09:29:32 - trees.REPTree
- 09:30:46 - trees.REPTree
- 09:31:01 - trees.REPTree

Classifier output

```

contribution-to-health-plan
class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

REPTree
-----
duration < 1.5 : none {3.13/0.13} [2/0]
duration >= 1.5
| wage-increase-second-year < 4.45 : full {15.65/3.65} [6/2]
| | wage-increase-second-year >= 4.45
| | | contribution-to-dental-plan = none : half {2/1} [1.38/0.38]
| | | | contribution-to-dental-plan = half : half {2.22/0} [1.43/0.43]
| | | | contribution-to-dental-plan = full : full {1/0} [2.19/0]

Time taken to build model: 0 seconds

Cross Validation Summary
Cross validation ==
instances      20          54.0541 %
instances      17          45.9459 %
              0.1153
              0.3475
              0.4451
              85.6267 %
              99.0473 %
              37
instances      20

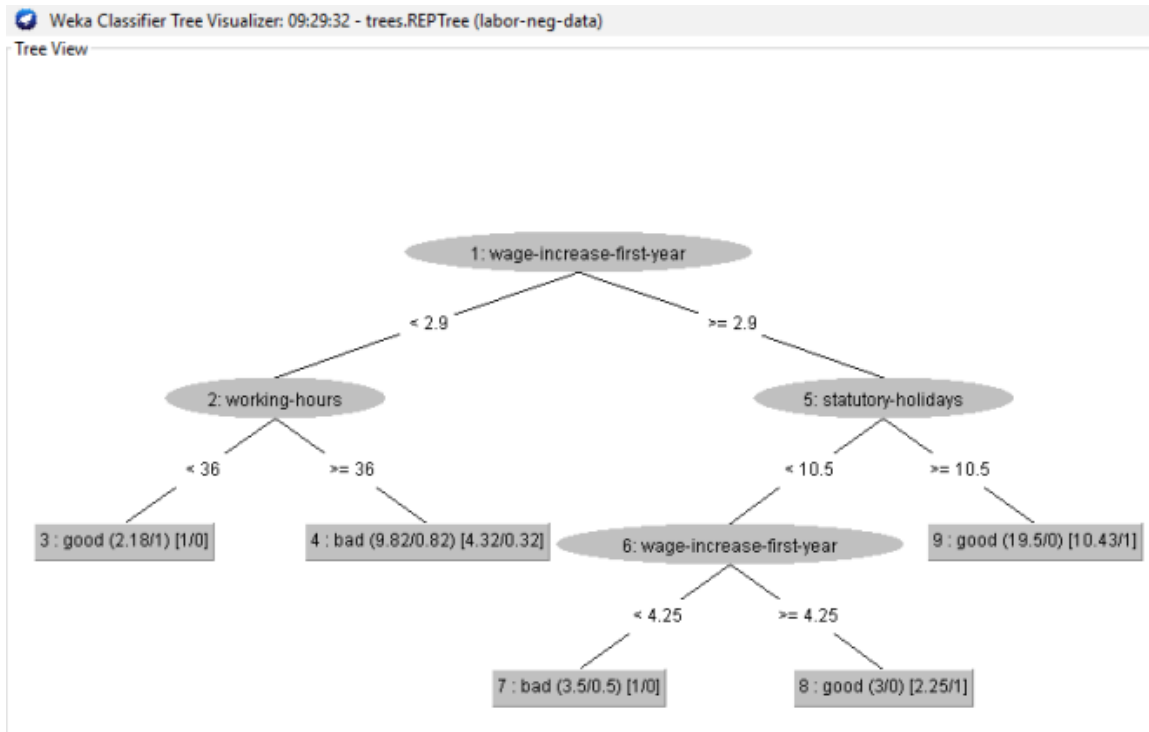
===== Detailed Accuracy By Class =====

           TP Rate   FP Rate   Precision   Recall   F-Measure   MCC   ROC Area   PRC Area   Clas
0.375   0.069   0.600   0.375   0.462   0.369   0.652   0.289   none
0.000   0.071   0.000   0.000   0.000   -0.136   0.569   0.184   half
0.850   0.765   0.567   0.850   0.680   0.109   0.570   0.474   full
Weighted Avg.   0.541   0.446   0.436   0.541   0.467   0.105   0.588   0.364

===== Confusion Matrix =====

a b c <-- classified as
3 0 5 | a = none
1 0 8 | b = half
1 2 17 | c = full
        
```

Step 8: The tree will look like:



Conclusion:

Successfully implemented K-medoid clustering algorithm using Python.