

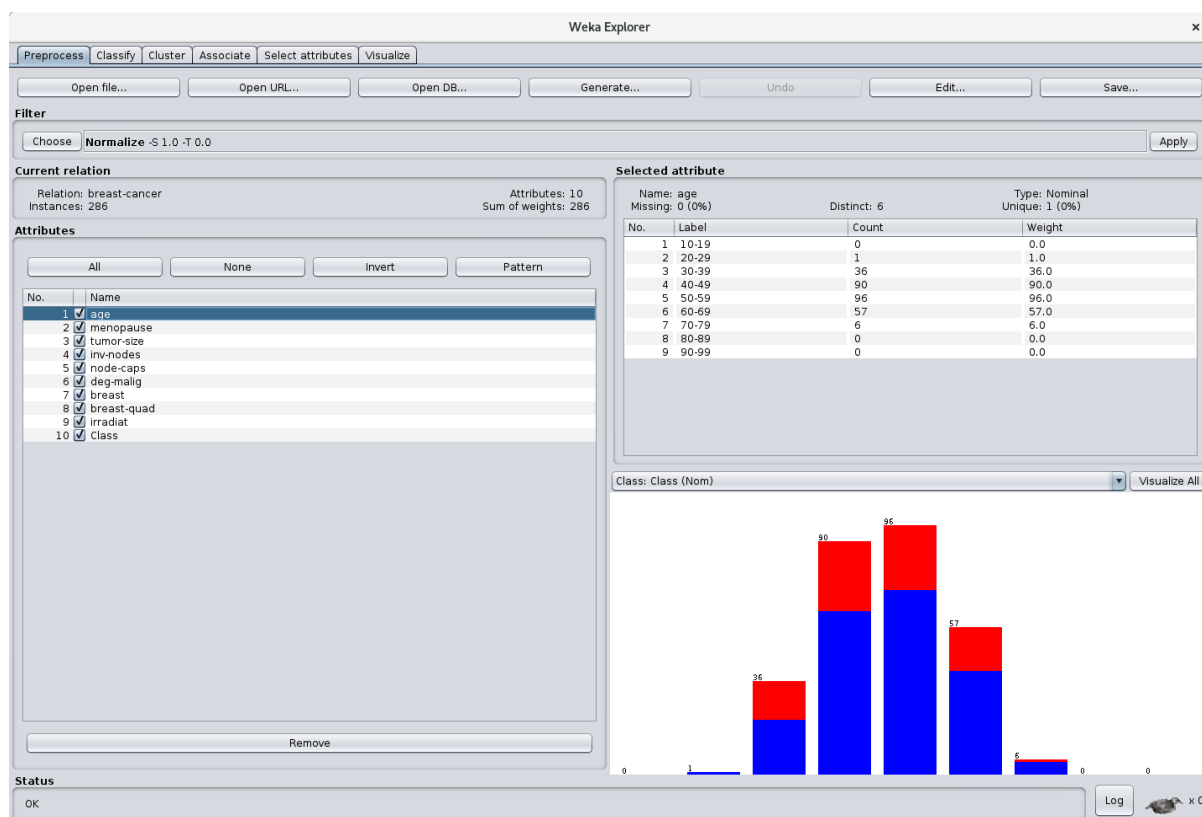
Problem:

The task is to create classifiers for the provided breast cancer dataset. We chose to do this project using the weka data mining software.¹ A J48 decision tree as well as a Naive Bayes classifier should be created.

The goal is to use open source tools to create insight into real world datasets and be able to validate the reliability of the methods.

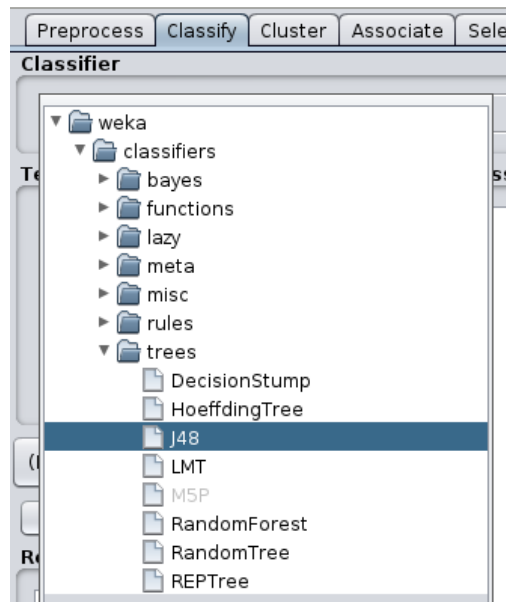
J48 - Weka GUI

When the Weka GUI is opened, the desired dataset can be opened by clicking on “Open File” and choosing the breast-cancer.arff file. The dataset can now be explored and various filters, for example normalisation, can be applied. For this exercise we chose to apply no filter on the dataset.

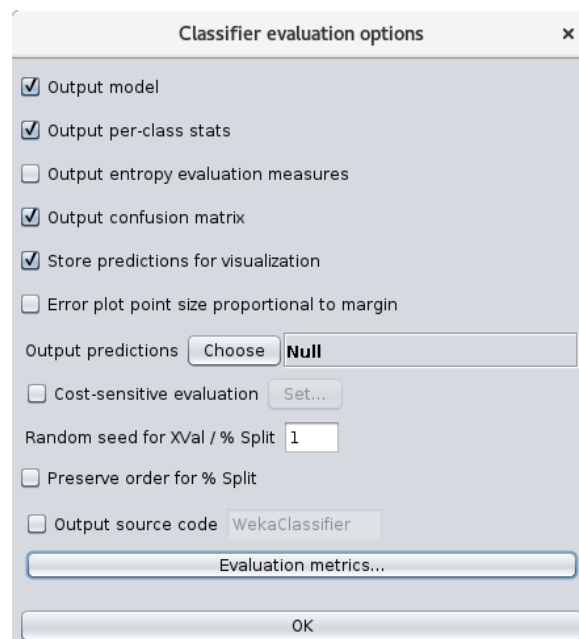


¹ <https://www.cs.waikato.ac.nz/ml/weka/>

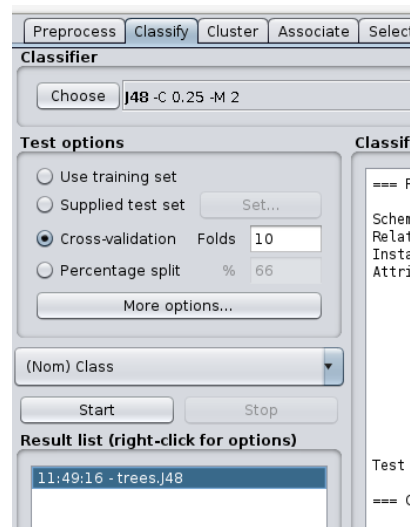
In the tab “Classify” the J48-classifier can be selected under weka > classifiers > trees > J48.



The following classifier evaluation options were chosen:



We use the k-folds cross-validation² method to test our classifier with a fold number of 10. The attribute to predict can now be selected. As is shown in the screenshot below, we select the attribute “Class”. We now click on start to generate the J48-classifier.



The following cross-validation summary is printed out.

```
=== Stratified cross-validation ===

Correctly Classified Instances      216           75.5245 %
Incorrectly Classified Instances    70           24.4755 %
Kappa statistic                    0.2826
Mean absolute error                 0.3676
Root mean squared error             0.4324
Relative absolute error             87.8635 %
Root relative squared error         94.6093 %
Total Number of Instances          286
```

As is seen above the decision tree has an accuracy of 75%.

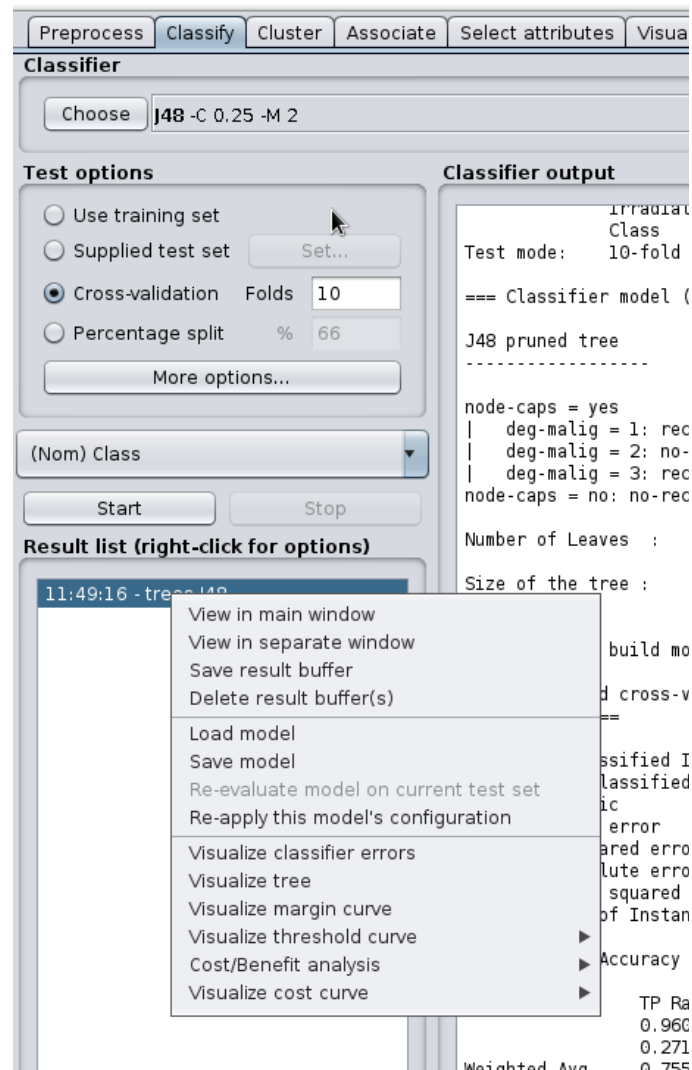
The classifier leads to the following confusion matrix.

```
=== Confusion Matrix ===
  a   b   <-- classified as
193   8 |   a = no-recurrence-events
 62  23 |   b = recurrence-events
```

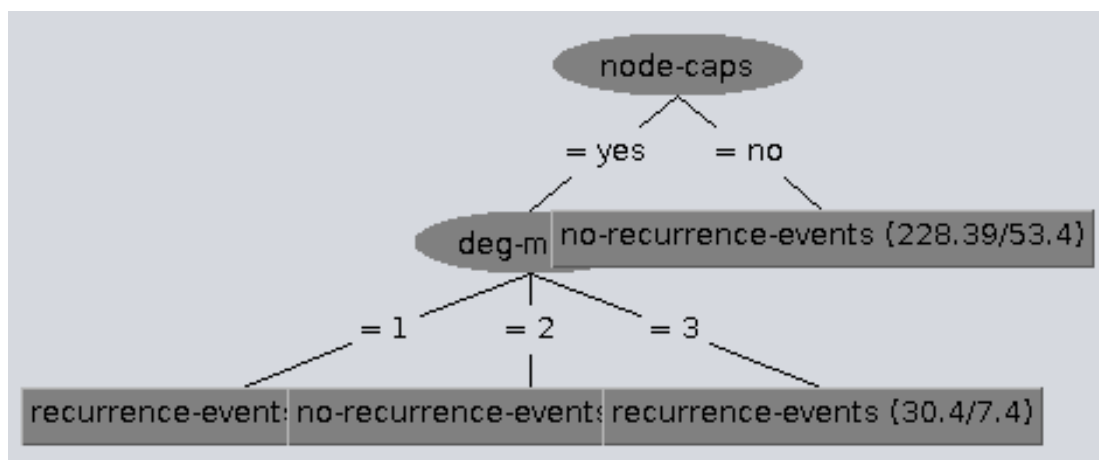
It can be seen that the majority of the “recurrence-events” class were falsely classified as class a (no-recurrence-events). In Total 70 instances were classified as the wrong class.

² [https://en.wikipedia.org/wiki/Cross-validation_\(statistics\)#k-fold_cross-validation](https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation)

The decision tree can now be visualized by right-clicking the classifier and selecting "Visualize tree".

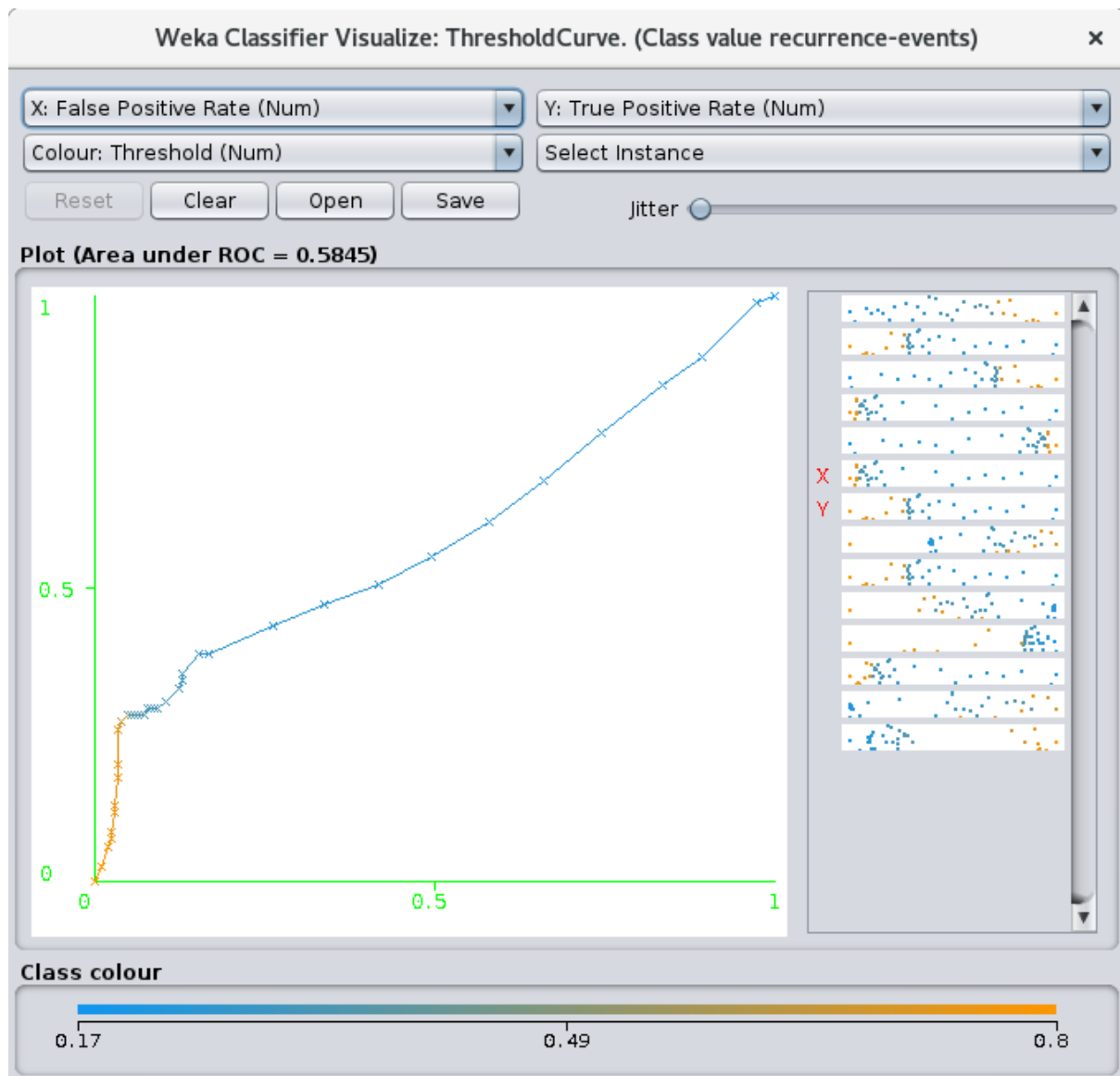


The following decision tree is now shown.



As is seen in the decision tree above, a lot attributes are purged out of the j48-tree. The final classifier is only based on the attributes “node-caps” and “deg-malig”.

A ROC curve can be created by right clicking the j48-classifier and selecting “Visualize threshold curve”. We select “recurrence-events” because we want to plot the curve for patients that are sick.



Naive Bayes - Weka Java API

The purpose of this part is to test the Naive Bayes classifier using the Java API, so we have written the following commented code

```
import weka.classifiers.Evaluation;
import weka.classifiers.bayes.NaiveBayes;
import weka.core.Instances;
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.Random;

public class Lab2 {
    public static void main(String[] args)
    {
        BufferedReader reader;
        Instances data = null;
        try {
            //Reading dataset
            reader = new BufferedReader(new
FileReader("dataset/breast-cancer.arff"));
            //Create seeded number generator
            Random rand = new Random(System.currentTimeMillis());
            //Create Instances from read data: Instances is a
class for handling an ordered set of weighted instances.
            data = new Instances(reader);
            //Randomize the position of instances.
            data.randomize(rand);
            reader.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println(data);
        //Setting the last attribute as class attribute
        data.setClassIndex(data.numAttributes() - 1);
        //Creating instances for train taking the half first part
        Instances train = data.trainCV(2, 0);
        //Creating instances for test taking the half last part
        Instances test = data.testCV(2, 1);
        try {
            //Creating Naive Bayes model
            NaiveBayes model=new NaiveBayes();
            //Building the classifier using the training Instances
            model.buildClassifier(train);
```

```
//Creating the evaluation with test instances
Evaluation eval = new Evaluation(test);
//Doing the evaluation specifying which model and test
instances use

eval.evaluateModel(model,test);
//Printing the confusion matrix
System.out.println(eval.toMatrixString());
//Printing the percentage of model
System.out.println(eval.toSummaryString());
} catch (Exception e) {
    e.printStackTrace();
}
// generate ROC curve
ThresholdCurve tc = new ThresholdCurve();
int classIndex = 0;
Instances result = tc.getCurve(eval.predictions(),
classIndex);

// plot ROC curve
ThresholdVisualizePanel vmc = new ThresholdVisualizePanel();
vmc.setROCString("(Area under ROC = " +
Utils.doubleToString(tc.getROCArea(result), 4) + ")");
vmc.setName(result.relationName());
PlotData2D tempd = new PlotData2D(result);
tempd.setPlotName(result.relationName());
tempd.addInstanceNumberAttribute();
// specify which points are connected
boolean[] cp = new boolean[result.numInstances()];
for (int n = 1; n < cp.length; n++)
    cp[n] = true;
try {
    tempd.setConnectPoints(cp);
    // add plot
    vmc.addPlot(tempd);
} catch (Exception e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}

// display ROC curve
String plotName = vmc.getName();
final javax.swing.JFrame jf =
    new javax.swing.JFrame("Weka Classifier Visualize:
"+plotName);
```

```

jf.setSize(500,400);
jf.getContentPane().setLayout(new BorderLayout());
jf.getContentPane().add(vmc, BorderLayout.CENTER);
jf.addWindowListener(new java.awt.event.WindowAdapter() {
    public void windowClosing(java.awt.event.WindowEvent e) {
        jf.dispose();
    }
});
jf.setVisible(true);
}
}

```

3 4 5

The result of the Naive Bayes classifier are the confusion matrix and the accuracy percentage, as we can see the accuracy level is almost the same as the J48 classifier. And as for the previous classifier the majority of b class is misclassified in a class.

=== Confusion Matrix ===

```

a b  <-- classified as
85 12 | a = no-recurrence-events
23 23 | b = recurrence-events

```

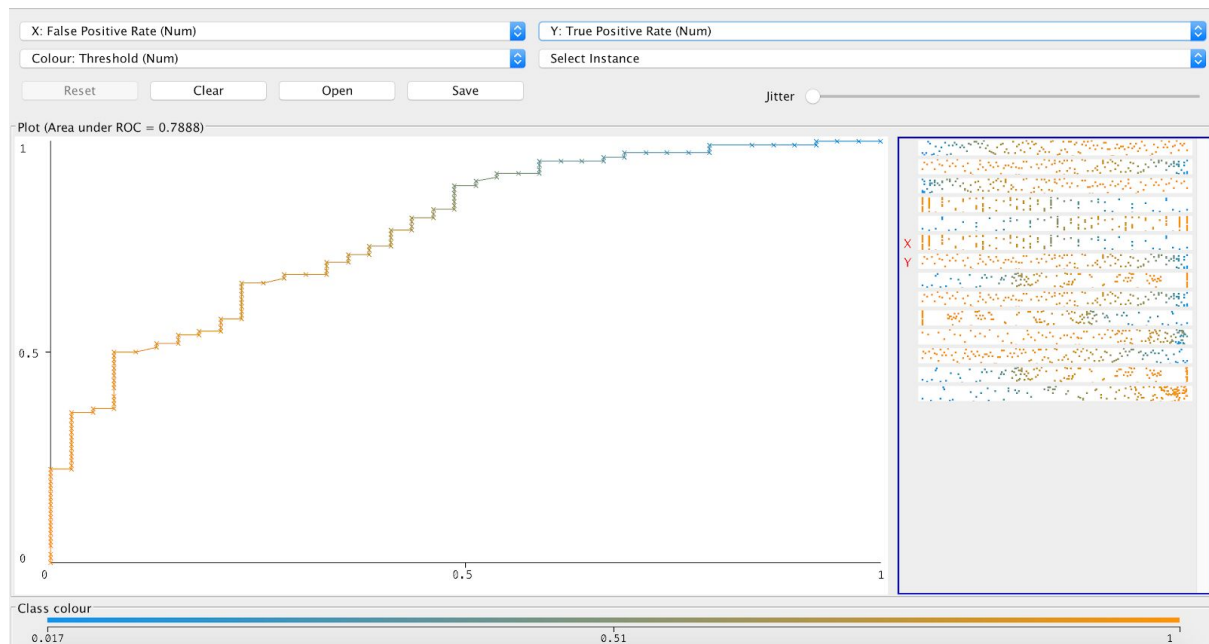
Correctly Classified Instances	108	75.5245 %
Incorrectly Classified Instances	35	24.4755 %
Kappa statistic	0.4015	
Mean absolute error	0.2951	
Root mean squared error	0.4183	
Relative absolute error	67.4922 %	
Root relative squared error	89.5506 %	
Total Number of Instances	143	

³ <http://weka.sourceforge.net/doc.stable/weka/classifiers/Evaluation.html>

⁴ <http://weka.sourceforge.net/doc.dev/weka/classifiers/bayes/NaiveBayes.html>

⁵ <https://weka.wikispaces.com/Generating+ROC+curve>

Comparing the following ROC curve with the one from j48 classifier we can see that the Naive Bayes as a better behaviour having less false negative.



Conclusion

The Weka GUI allows for a very fast and seamless process in exploring the data and applying various filters. A classifier can then easily be created, saved and modified. All this is possible without having to write code which makes it easier, especially for beginners, to do various data mining tasks. The Weka Java API can be used to obtain more complex analysis and results. As we have seen in this dataset, applying a J48 tree classifier or a Naive Bayes classifier gives us the same accuracy result of 75.5245%. From the confusion matrix we can see that in the J48 tree classifier, the false negative for recurrence-event are higher than Naive Bayes classifier (62 instead of 23), so for medical purposes should be better to use Naive Bayes classifier so that we have less probability to have patients with cancer classified as they don't have it. $23/(85+23)=21,3\%$ for Naive Bayes instead of $62/(193+62)=24,3\%$ for J48 tree.