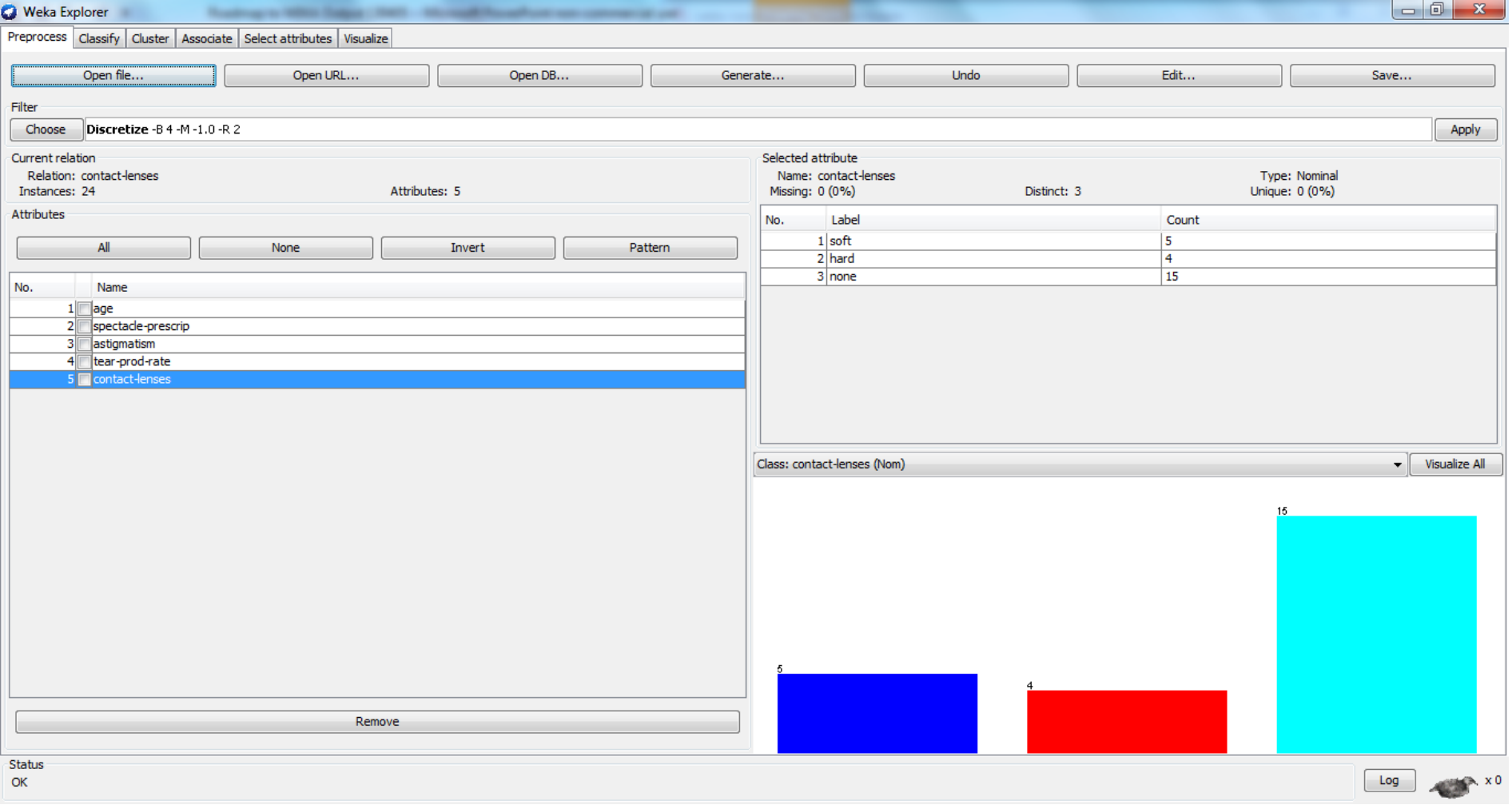
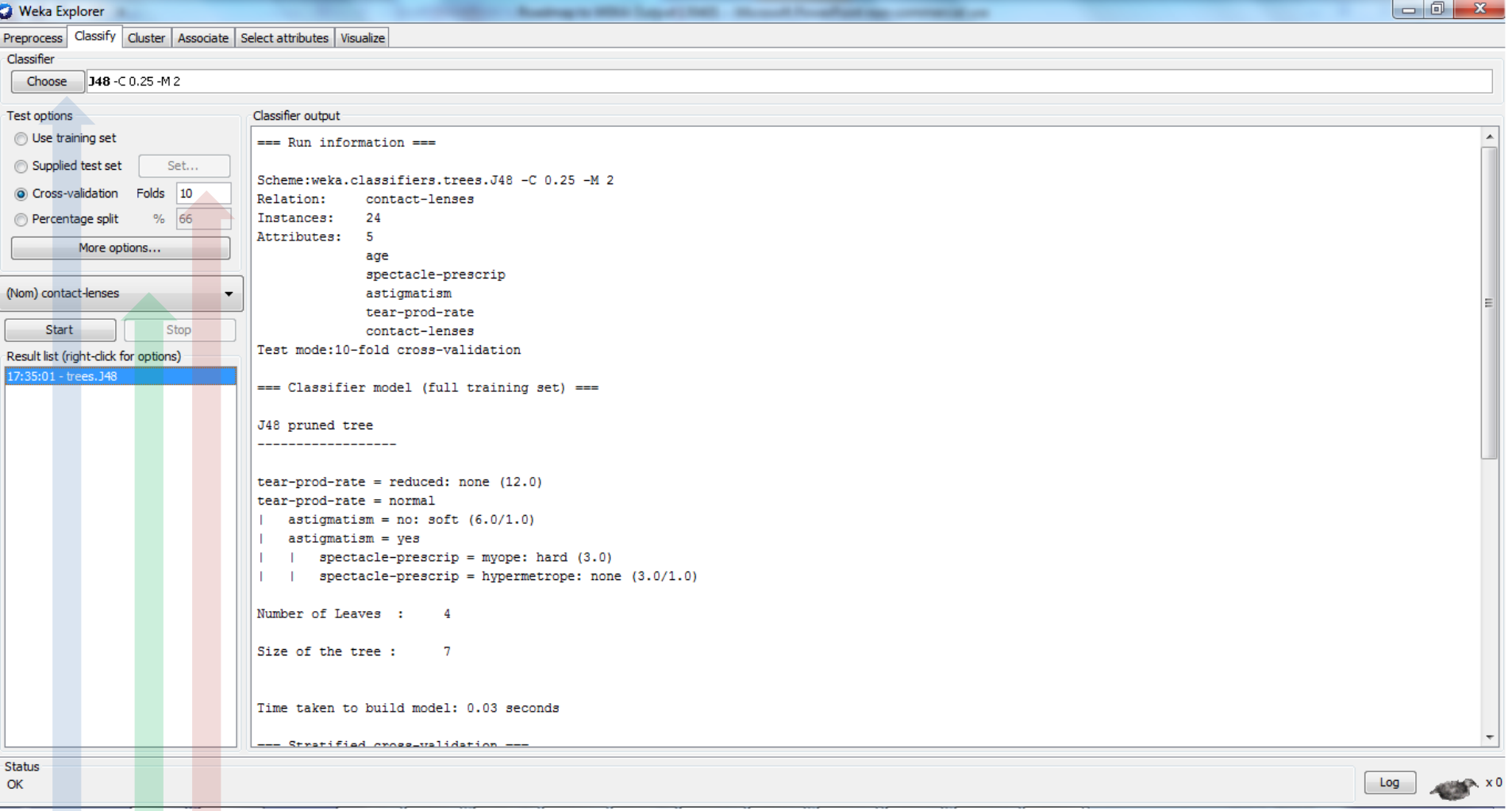
Roadmap to WEKA Classifier Output: J48 Decision Tree

The goal of this guide is to provide a brief introduction and tour of the elements in the WEKA classifier output screen

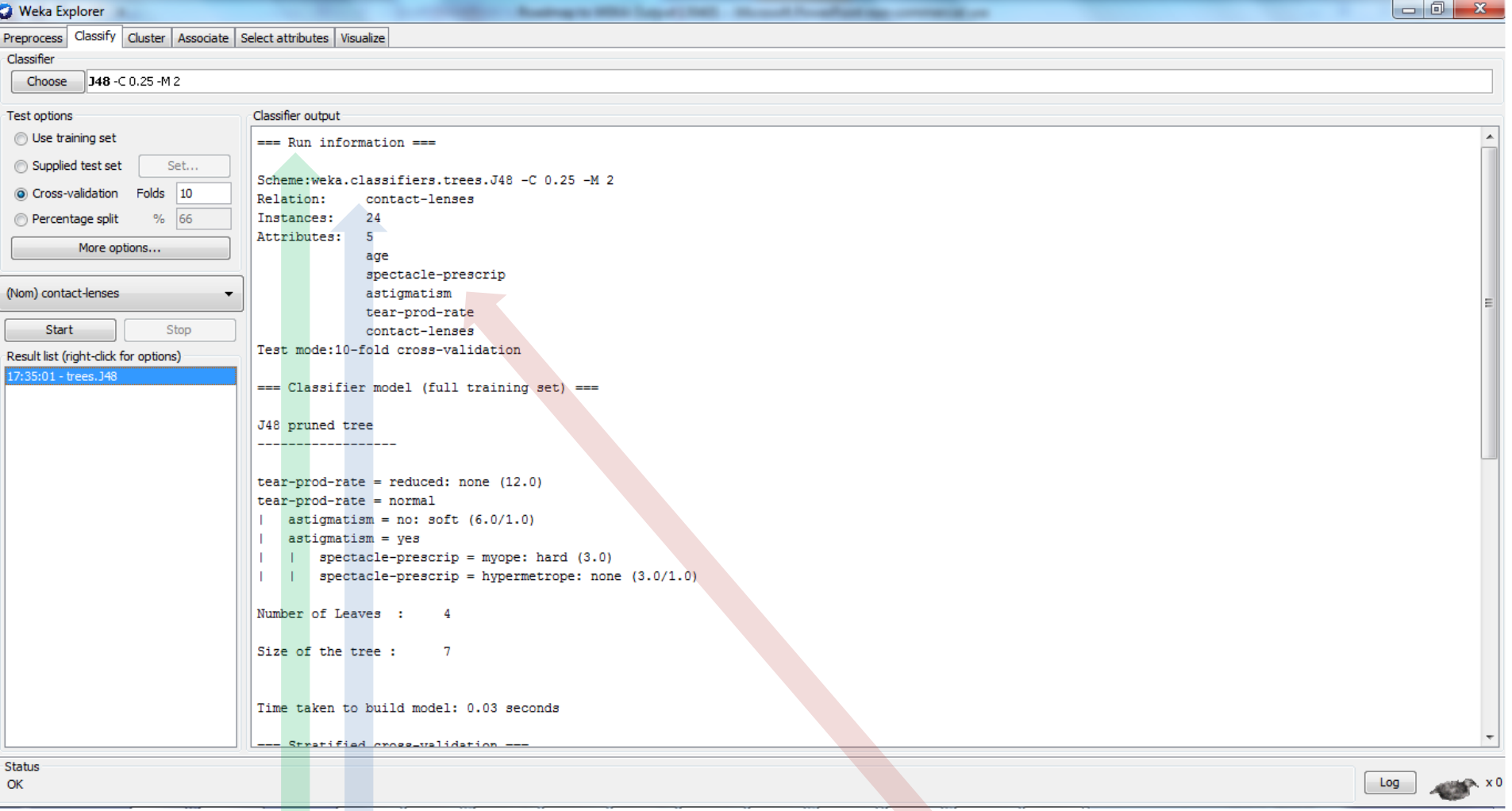


The example used in this guide is the contact lens problem explained in Chapter 1 of ***Data Mining—Practical Machine Learning Tools and Techniques.***
In this idealized problem we want to create a model to predict what type of contact lens should be prescribed given four data points about a patient (age, tear production, astigmatism, and spectacle prescription).

The data file can be found in the data folder within the WEKA program file.



In this example, we trained a ***J48 decision tree classifier*** on the data;
using the nominal attribute contact lens as the ***classification target,***
We used ***10-fold cross-validation*** to validate the criterion the decision tree learns from the data.
Refer to the course resources for more information on decision trees and cross-validation.



The Run Information Section appears at the top of the window and restates information about the run of this classifier, including:

•The classifier used and the settings applied to the data, including filters. In this case, the J48 decision tree is run with the default settings, and no filters are applied to the data.

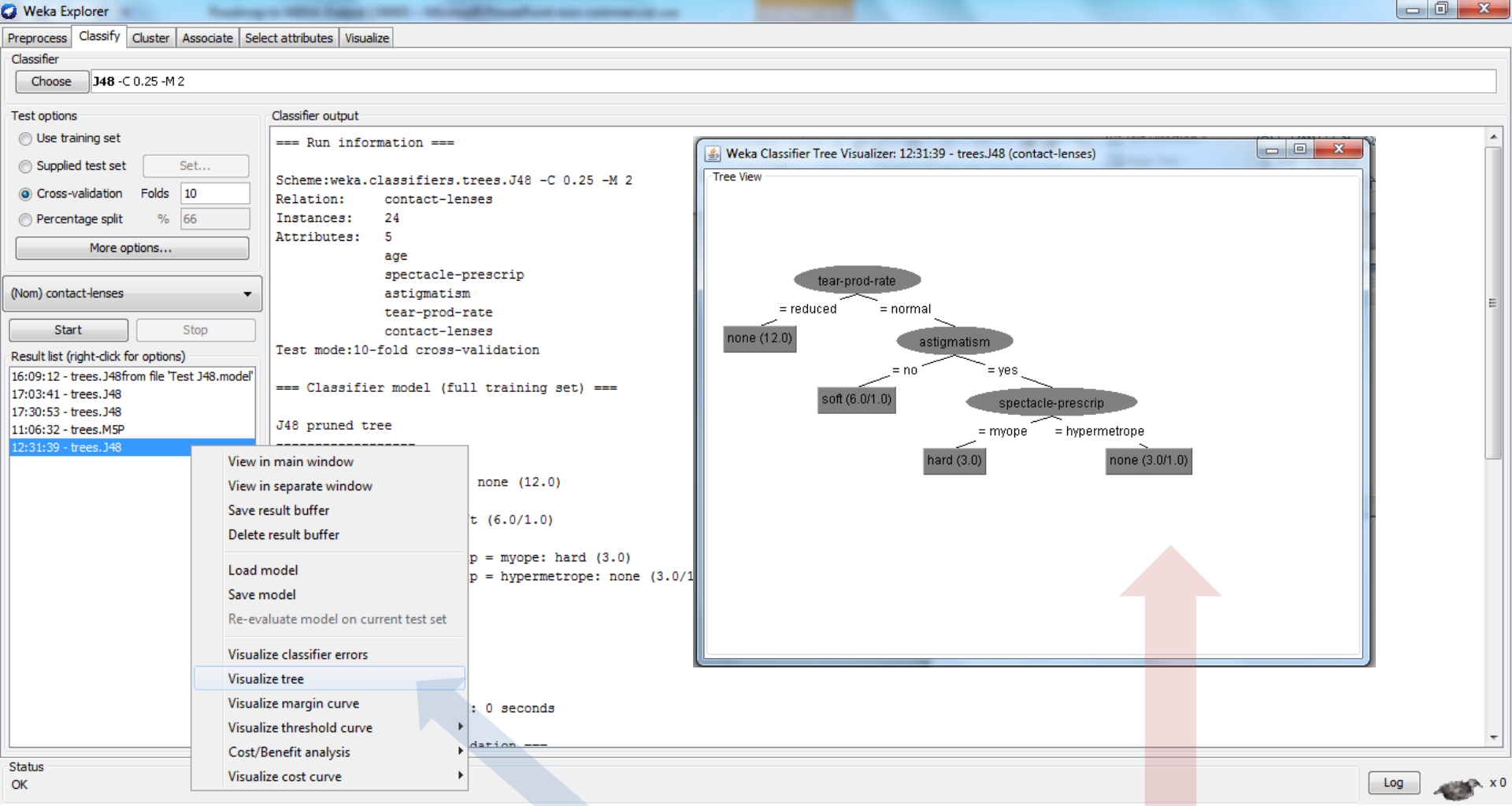
•The name of the relation (contact-lenses), which is the name of the data file used; the number of instances in the file (24), the number of attributes in the file (5), the names of each of the attributes and how the classifier was evaluated (10-fold cross-validation)

Classifier output

```
=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     contact-lenses
Instances:    24
Attributes:   5
              age
              spectacle-prescrip
              astigmatism
              tear-prod-rate
              contact-lenses
Test mode:10-fold cross-validation

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

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

tear-prod-rate = reduced: none (12.0)
tear-prod-rate = normal
|   astigmatism = no: soft (6.0/1.0)
|   astigmatism = yes
|   |   spectacle-prescrip = myope: hard (3.0)
|   |   spectacle-prescrip = hypermetrope: none (3.0/1.0)

Number of Leaves  :     4

Size of the tree :      7


Time taken to build model: 0 seconds
```

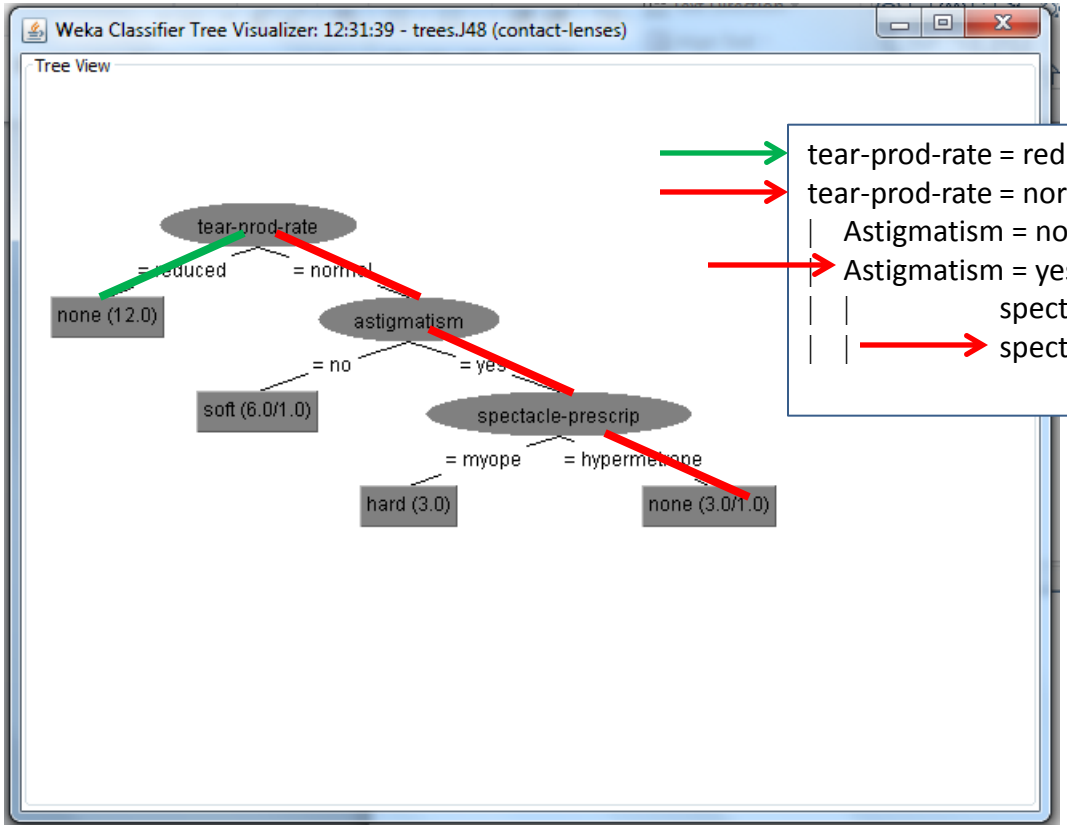
Under the "classifier model" heading, you will find the model that the classifier created.
In this case, a decision tree is constructed using the J48 classifier. Note that the tree has been pruned. The rows indicate how the classifier uses the attributes to make a decision (e.g., tear production rate is equal to zero). The first split is on tear production, the second split is on astigmatism, and the third split is on spectacle prescription.

In the tree structure, a colon introduces the class label that has been assigned to a particular leaf.
The label following the colon after a leaf node indicates which class an instance will be assigned to should that node be reached (in this case the classes are : none; : hard, : soft). The numbers in brackets after the leaf nodes indicate the number of instances assigned to that node, followed by how many of those instances are incorrectly classified as a result. In this case, we are trying to predict the kind of contact lens to prescribe, given four information points that we know about a patient (age, spectacle prescription, astigmatism, and tear production rate). In looking at the rules, we can also see that three of the four attributes have some correlation to what type of contact to prescribe, however one of the four attributes (age) does not.



To understand how to read the decision rules used to assign a class, it often helps to look at a graphical representation of the tree.

By right-clicking on the classifier and clicking **Visualize tree**, you will get a **graphical representation of the decision tree** model the classifier developed.



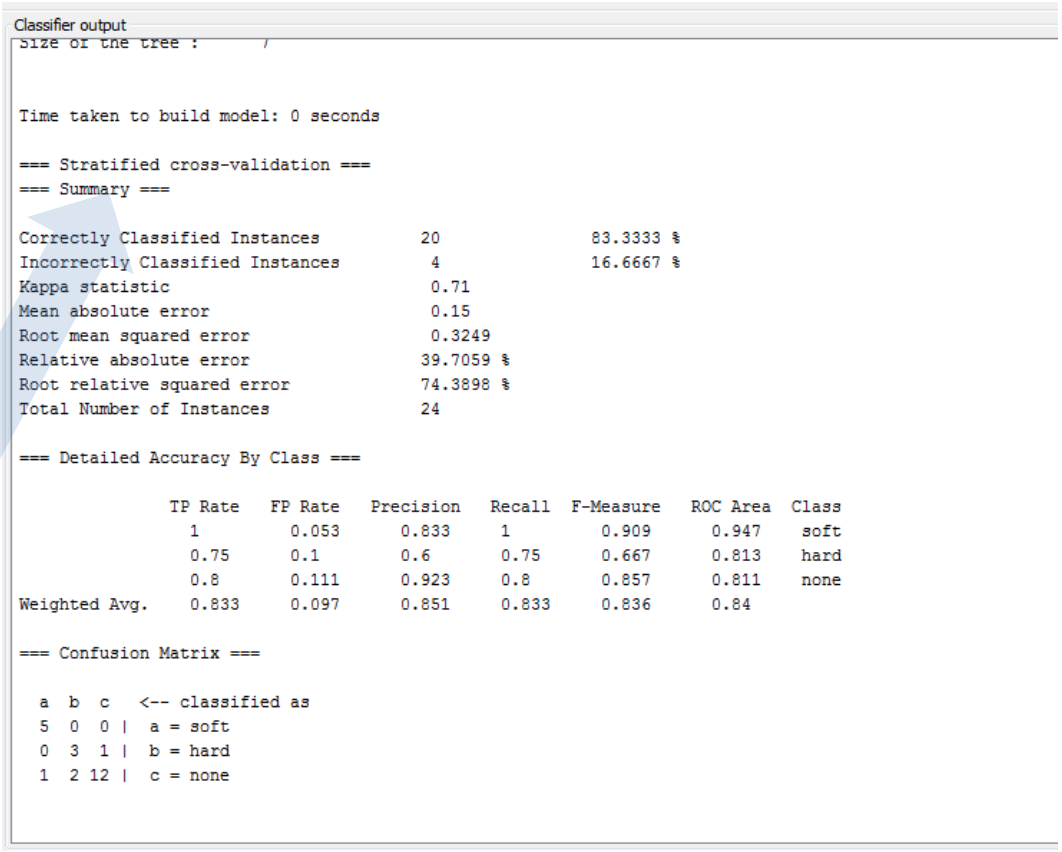

```
tear-prod-rate = reduced: none (12.0)
tear-prod-rate = normal
|   Astigmatism = no: soft (6.0/1.0)
|   Astigmatism = yes
|   |               spectacle-prescrip = myope: hard (3.0)
|   |               spectacle-prescrip = hypermetrope: none (3.0/1.0)
```

Tracing the decisions through the tree can help you understand the rules the model found to classify the outcome.

For example, if we want to understand what set of rules and attributes predict that no contact lenses are prescribed (two leaves in the tree actually assign the class "none"), we can trace up through the tree to determine what rules indicate these classifications. In doing this we find that the rules for this outcome are:

1.tear-prod rate = reduced

2.Tear-prod-rate normal; astigmatism yes and spectacle prescription of hypermetrope



The "Summary" section of the output gives statistics of the tree's predictive performance.

The line immediately above "Summary" indicates what method was used to test the performance; in this case it's the stratified cross-validation method. Stratified means that each class is represented equally in each fold of cross validation.

Classifier output

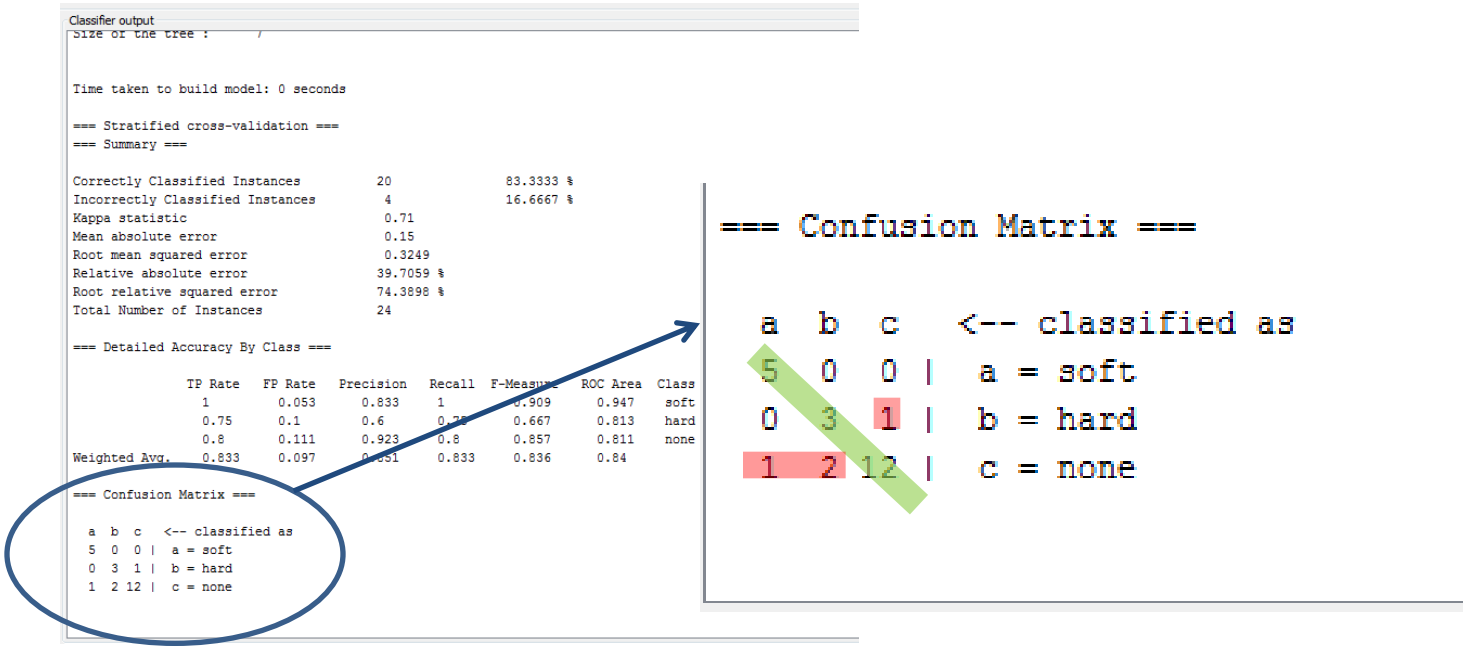
```
Size of the tree :      7


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          20               83.3333 %
Incorrectly Classified Instances         4               16.6667 %
Kappa statistic                          0.71
Mean absolute error                      0.15
Root mean squared error                  0.3249
Relative absolute error                 39.7059 %
Root relative squared error             74.3898 %
Total Number of Instances               24

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0.053      0.833     1         0.909      0.947    soft
                 0.75      0.1        0.6       0.75      0.667      0.813    hard
                 0.8       0.111      0.923     0.8       0.857      0.811    none
Weighted Avg.    0.833     0.097      0.851     0.833     0.836      0.84

=== Confusion Matrix ===

  a  b  c   <-- classified as
  5  0  0 |  a = soft
  0  3  1 |  b = hard
  1  2 12 |  c = none
```

When evaluating the accuracy of a classifier on nominal values, one place to start is **the Correctly Classified Instances and the Incorrectly Classified Instances**. With the exception of the Kappa statistic, the remaining statistics compute various error measures based on the class probabilities assigned by the tree.

•In this case, the algorithm accurately predicted the classification of contact lens about 83% of the time (20 out of 24 instances) and misclassified it about 16% of the time (4 out of 24).

In many business, scientific, and engineering contexts greater confidence is places in results that are above the psychologically significant 90% threshold. Be aware that this may not be an acceptable or optimal threshold for confidence in all cases - it is important to find the best level of performance you are able to achieve with various classifiers, then compare those results to the business, science, or engineering requirements of the problem.

The best performance of your classifier and the required operating levels of the problem are two separate values. It is important not to treat your best performance as acceptable if it does not meet the requirements of the problem.



One of the most useful tools for evaluating performance of nominal classifiers is the **Confusion Matrix**.

At a glance, this can tell you how many misclassifications the algorithm made, and what those misclassifications were. In this example, the confusion matrix can be read as:

Correct Classifications

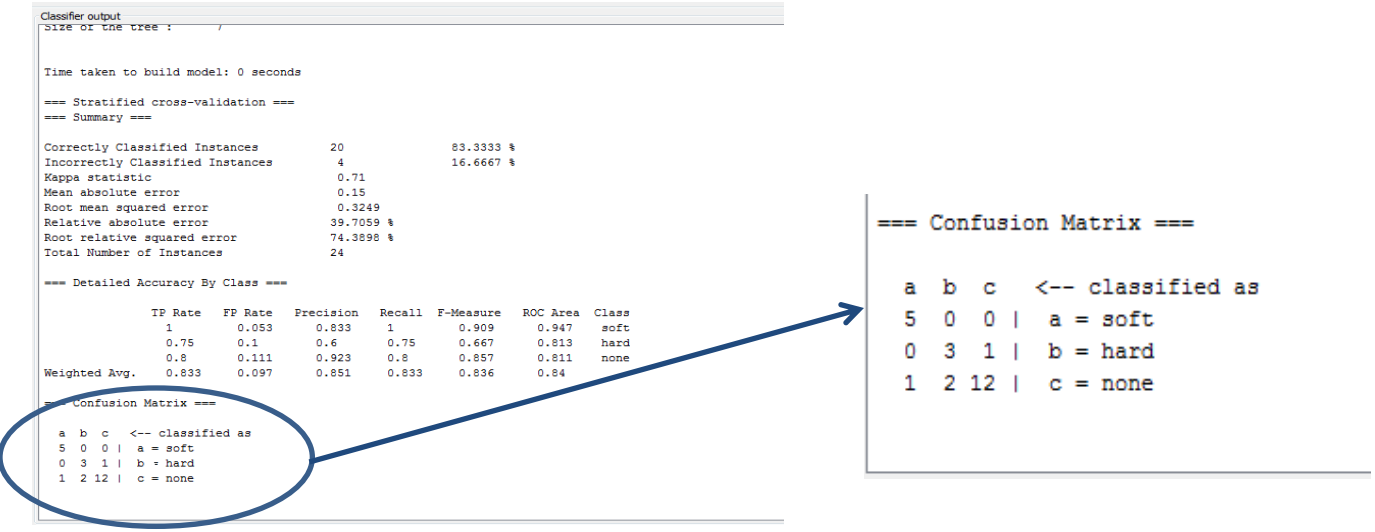
5 (“a”-soft classified as “a”-soft)+ 3 (“b”-hard classified as “b”-hard) + 12 (“c”-none classified as “c”-none)=20

Incorrect classifications

1 (“a”-soft classified as “c”-none)+ 2 (“b”-hard classified as “c”-none) + 1 (“c”-none classified as “b”-soft) = 4

Overall, this algorithm did a fair job at classifying this attribute. It did just a slightly worse job at classifying the cases where the nominal value is b (hard contacts)--misclassifying one out of four instances--than it did classifying the cases where the nominal value is c (no contacts), misclassifying three out of 15 instances.

Rule of Thumb: Good results correspond to large numbers down the main diagonal, and smaller, ideally 0, numbers off diagonal elements.



True Positives, True Negatives, False Positives, and False Negatives

For some problems, the cost of making a wrong prediction is significant; and sometimes it isn't. For example, the cost of misidentifying brake problems in a car that turns out to be free of faults is much less than overlooking a car with brakes that are about to fail. If wrong decisions have different costs, understanding the different types of wrong predictions is helpful. The confusion matrix can help us understand the types of incorrect predictions the classifier made.

Correct classifications come in two types: **true positives and true negatives.**

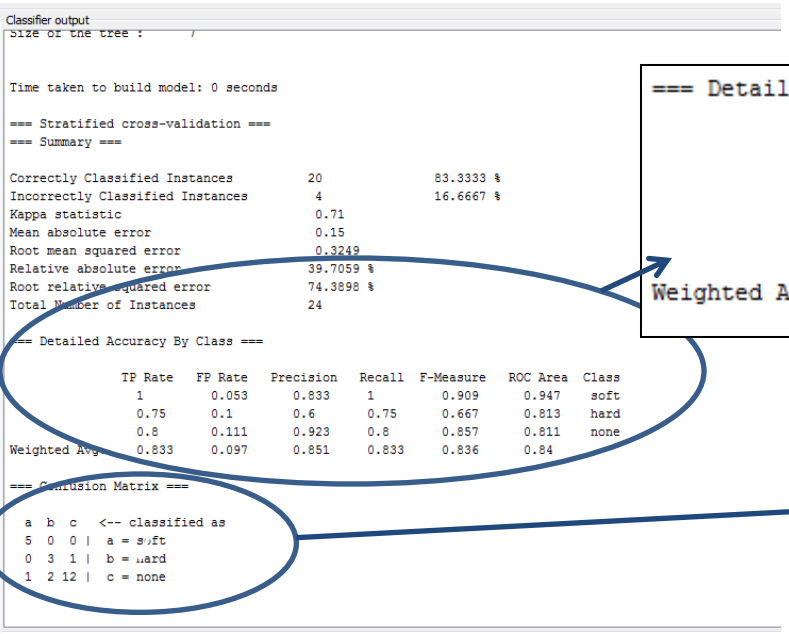
•**True positives** are the instances that are correctly predicted as the actual class. For example, looking at the confusion matrix, the number of true positives for the class **"hard "** is 3 – 3 instances are predicted as b, when the actual class is b.

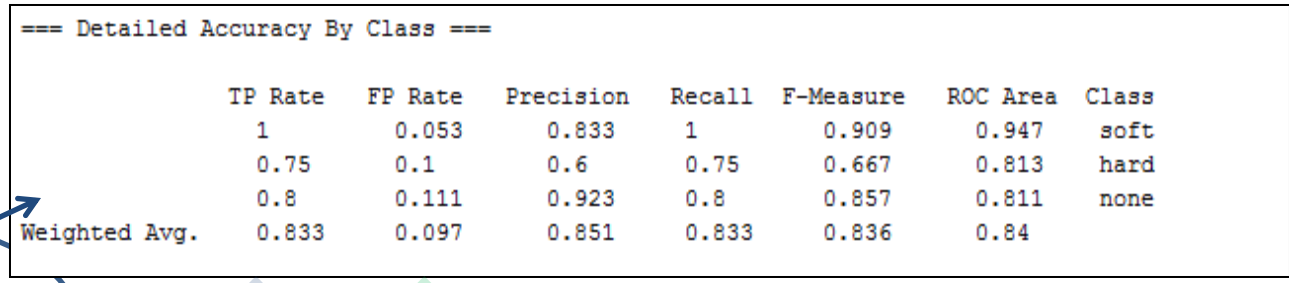
•**True negatives** are instances that are correctly predicted as not of the class. For the class **"hard,"** the number of true negatives is 18.

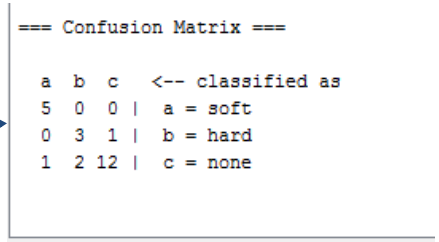
Likewise, incorrect predictions come in two types: **false positive and false negatives.**

•**False positives** are the instances that are incorrectly predicted as positive or belonging to that class, when they actually belong to another class. The number of false positives for class "**hard** " is 2. Number of instances in which a is classified *incorrectly* as b (0) + Number instances in which c is *incorrectly* classified as b (2) = 2.

•**False negatives** are instances incorrectly predicted as negative or not belonging to the class when they actually do. The number of false negatives for class **"hard"** is 1. Number of instances in which b classified as a (0) + Number of instances of b classified as c (1)= 1.







The TP (True Positive) rate and FP (False Positive) rate are statistics that can give further insight into the nature of errors by class.

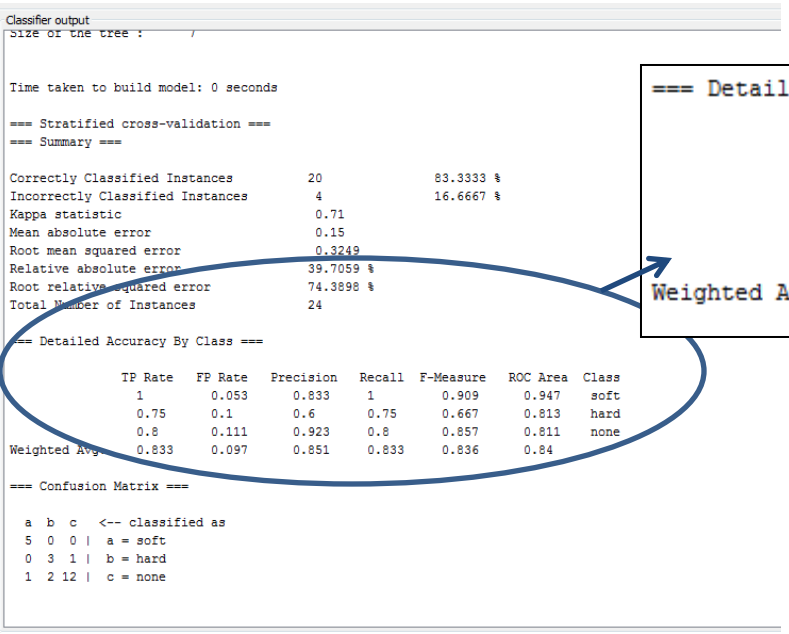
TP Rate (True Positive Rate)

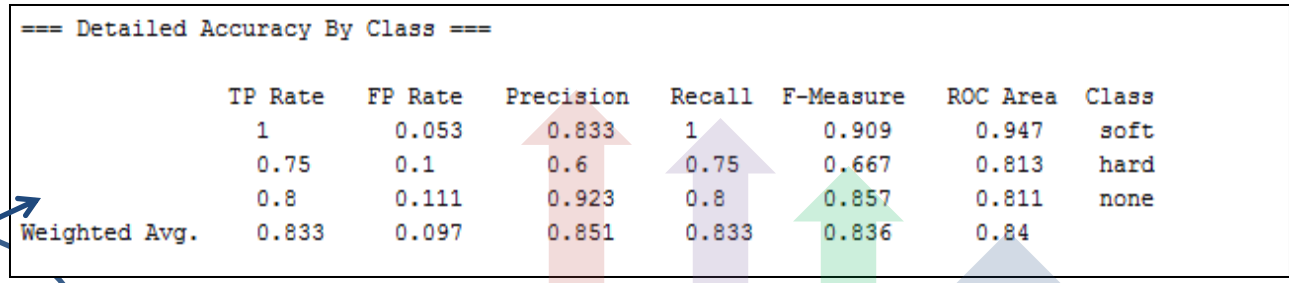
The TP Rate measures the proportion of positives that were correctly identified by the model as such.
It is calculated by dividing the number of true positives by the total number of possible positives : number of true positives/(true positives + false negatives).
So, in this case , the TP rate is calculated:
soft : 5/(5+0) = 1
hard: 3/(3+1) = .75
none : 12/(12+3)) = .8
Rule of Thumb: The TP rate is a measure of success, so higher numbers are better.

FP Rate (False Positive Rate)

The FP Rate measures the proportion of negatives that are incorrectly identified by the model as positives. The FP rate is calculated by dividing the number of false positives by the total number of possible negatives : number of false positives/(true negatives + false positives):
soft : 1/(18+1)= .053
hard: 2/(18+2) = .1
none : 1/(8+1)= .111
Rule of Thumb: The FP is also a measure of error, so lower numbers are better.

To learn more about these statistics see section 5.7 in ***Data Mining—Practical Machine Learning Tools and Techniques.***

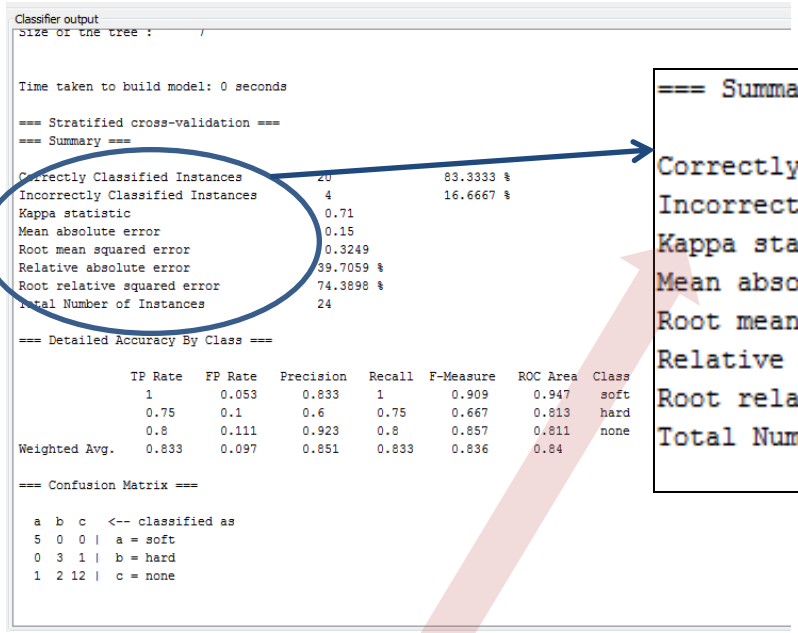


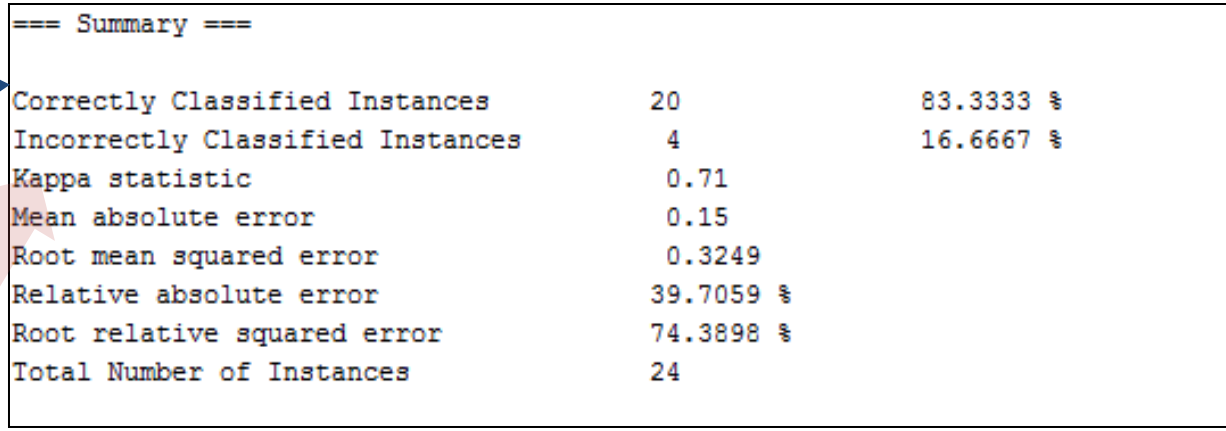


The additional measures in the "Detailed Accuracy by Class" section are used to evaluate the trade-off between false negatives and false positives.

- **Precision** (also called positive predictive value) is calculated: Number of True Positives/(Number of True Positives + the Number of False Positives).
- **Recall** It is calculated number of True Positives/(Number of True Positives + Number of False Negatives)
- **F-Measure** is a combined measure for precision and recall. It is calculated as 2*Precision*Recall/(Precision + Recall)
- **ROC Area** is a measure of false positive/false negative performance - particularly how well it can differentiate false positives from false negatives.

To learn more about these statistics, see **Section 5.7** in ***Data Mining—Practical Machine Learning Tools and Techniques.***





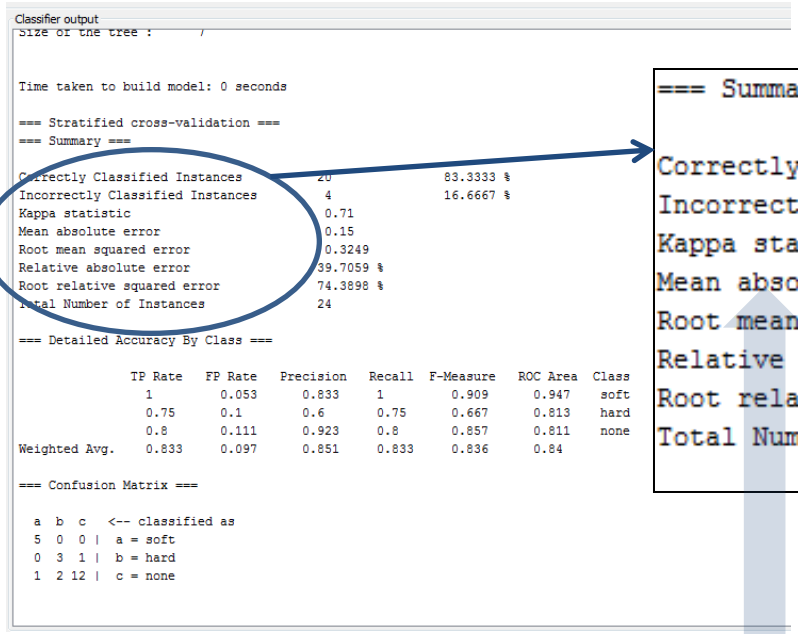
Kappa Statistic

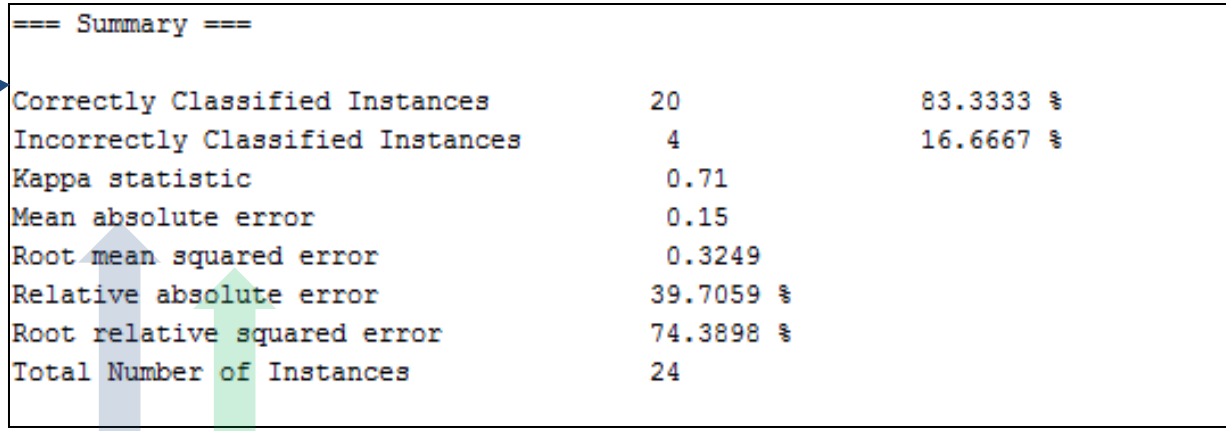
The Kappa Statistic takes into account how the classifier's prediction compares to how many correct predictions you would expect by random chance. The Kappa Statistic is calculated by deducting the number of instances a random predictor would classify correctly from the number of instances the model classified correctly and expresses the result as a proportion of the total for a perfect predictor.

The calculation is:
(# correctly predicted by the model) - # correctly predicted by random predictor)/
(total instances – # of instances predicted by random predictor.)
So in this case, since the Kappa is .71, we know that the correctly classified instances expected with a random predictor is 10. The calculation is: (20-10)/(24-10) = .71

Rule of Thumb: The maximum value of a Kappa Statistic is 1; depending on the complexity of the prediction task, a Kappa Statistic of 0.7-0.8 is generally considered good and a value of 0.9 is excellent.

*See section 5.7 in **Data Mining: Practical Machine Learning** to learn more about the Kappa statistic.*



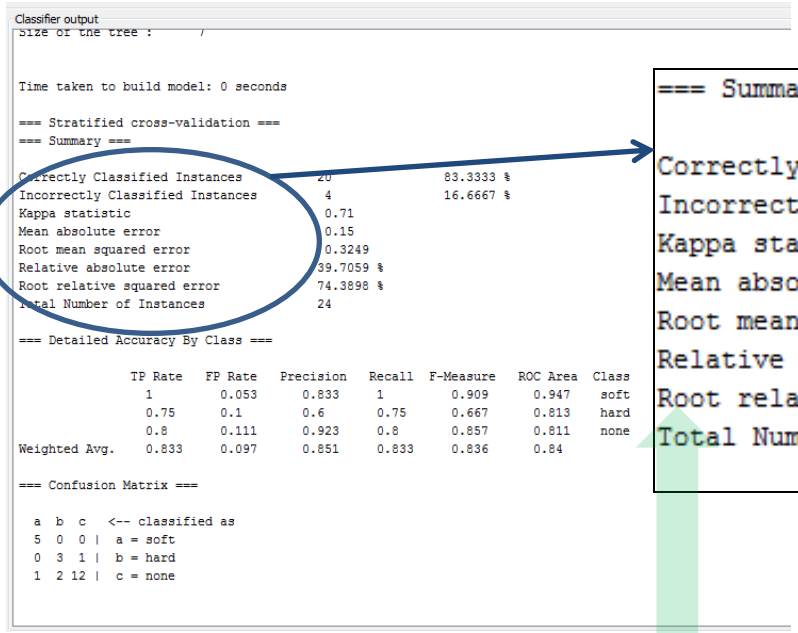


Mean Absolute Error (MAE)

The mean absolute error measures the average magnitude of errors in the set of predictions, without considering their direction. As the name suggests, MAE is the average over the all the verified predictions of the absolute values of the differences between the predicted and verified observations. All the individual differences are weighted equally in the average. MAE is most useful when we are predicting numeric attributes (such as in regression tasks) rather than nominal ones because it enables us to assess the degree of error.
Rule of Thumb: Since this statistic measures the magnitude of error, lower values are better than a higher values. The threshold of acceptable MAE is a function of the classification task and data set being analyzed.

Root Mean-Squared Error (RMSE)

RMSE is an alternative to MAE. It also measures the average magnitude of errors in a set of predictions. Expressing this formula in words, the difference between forecast and corresponding observed values are each squared and then averaged over the sample. Finally, the square root of the average is taken. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the RMSE is most useful when large errors are particularly undesirable. Conversely, mean-squared error tends to exaggerate the effect of outliers.
Rule of Thumb: Since this statistic measures the magnitude of error, lower values are better than higher values.



=== Summary ===

Correctly Classified Instances	20	83.3333 %
Incorrectly Classified Instances	4	16.6667 %
Kappa statistic	0.71	
Mean absolute error	0.15	
Root mean squared error	0.3249	
Relative absolute error	39.7059 %	
Root relative squared error	74.3898 %	
Total Number of Instances	24	

Root Relative Squared Error

It is used primarily to assess the level of error in predicting numeric values. The error is made relative to what it would have been if a simple predictor would have been used. The simple predictor is just the average of the actual values for the attribute being predicted. It is calculated by taking the total of the squared error (discussed on the previous slide) and dividing it by the total squared error of the simple (default) predictor.

Rule of Thumb: Since this statistic measures the magnitude of error, lower values are better than a higher values.

Relative Absolute Error

This is statistic is similar to the Root Relative Squared Error. The absolute error is made relative to the simple predictor as with the Root Relative Squared Error. The Relative Absolute Error is calculated by taking the Mean Absolute Error (discussed in the previous slide) and dividing it by the mean absolute error of the default predictor (the average of the actual values being predicted.)

Rule of Thumb*:* Since this statistic measures the magnitude of error, lower values are better than a higher values.

It is useful to try to gain some insight as to what acceptable ranges of error are for the problem being addressed - particularly in the engineering domain. If that is not known, you simply want to achieve the lowest error possible and record that information for comparison with future efforts to improve it.