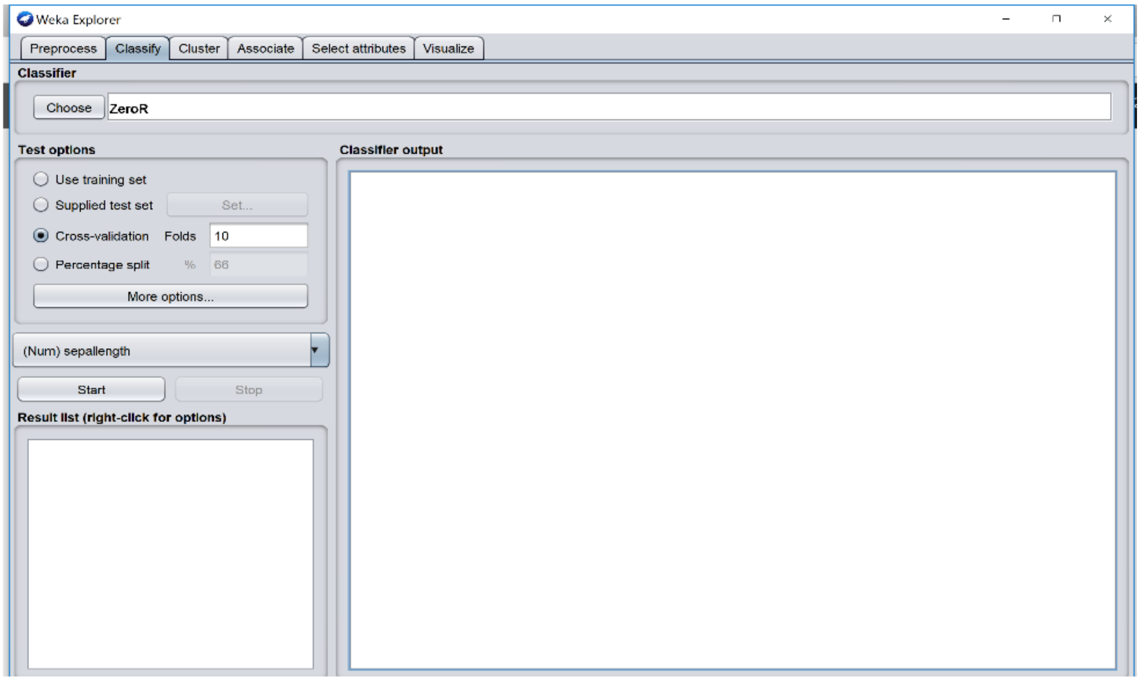
Solution to Laboratory Week 3

1. The data files for this lab can be found at /KDrive/SEH/SCSIT/Students/Courses/COSC2111/DataMining/data

2. Load the file arff/UCI/iris.arff.

(a) Go to the classifiers screen and select sepallength as the class attribute.



Classifier: ZeroR

Test options

○ Use training set

○ Supplied test set Set...

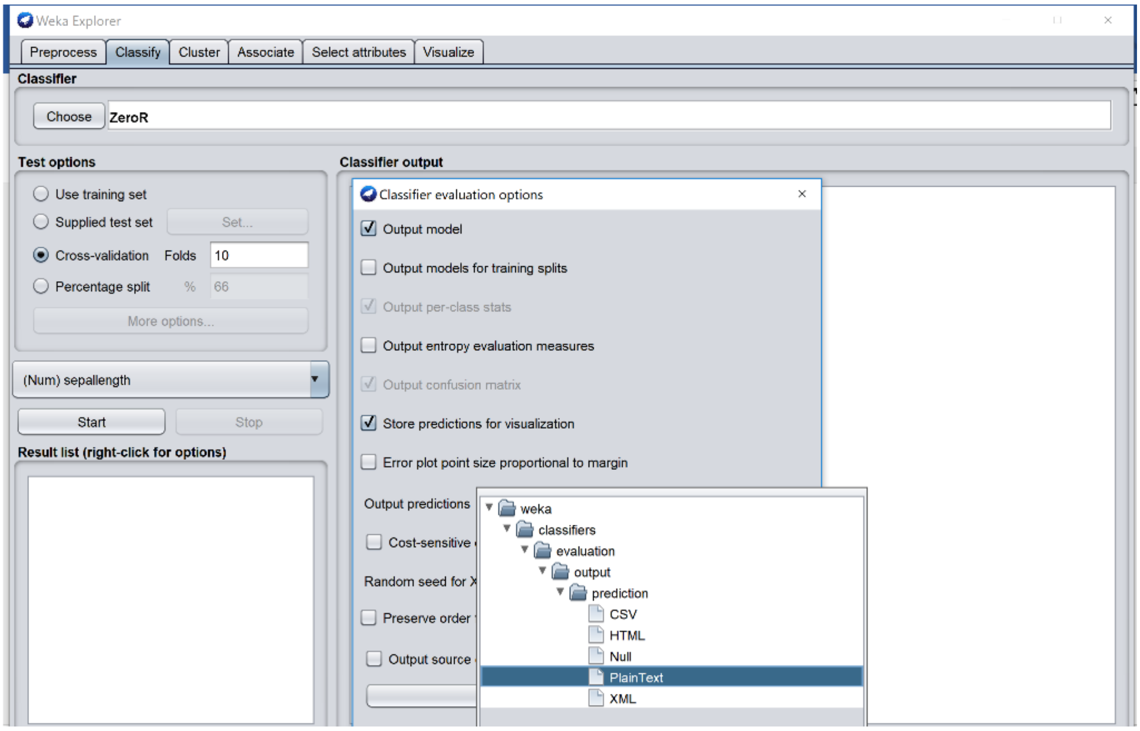
● Cross-validation Folds 10

○ Percentage split % 66

(Num) sepallength

Figure 1. Screenshot of Weka on select sepallength as the class attribute.

(b) Select ‘More Options’ and ‘Output Predictions’.



Classifier: ZeroR

Test options

○ Use training set

○ Supplied test set Set...

● Cross-validation Folds 10

○ Percentage split % 66

(Num) sepallength

Classifier evaluation options

☑ Output model

□ Output models for training splits

☑ Output per-class stats

□ Output entropy evaluation measures

☑ Output confusion matrix

☑ Store predictions for visualization

□ Error plot point size proportional to margin

Output predictions

- weka
  - classifiers
    - evaluation
      - output
        - prediction
          - CSV
          - HTML
          - Null
          - PlainText
          - XML

□ Cost-sensitive

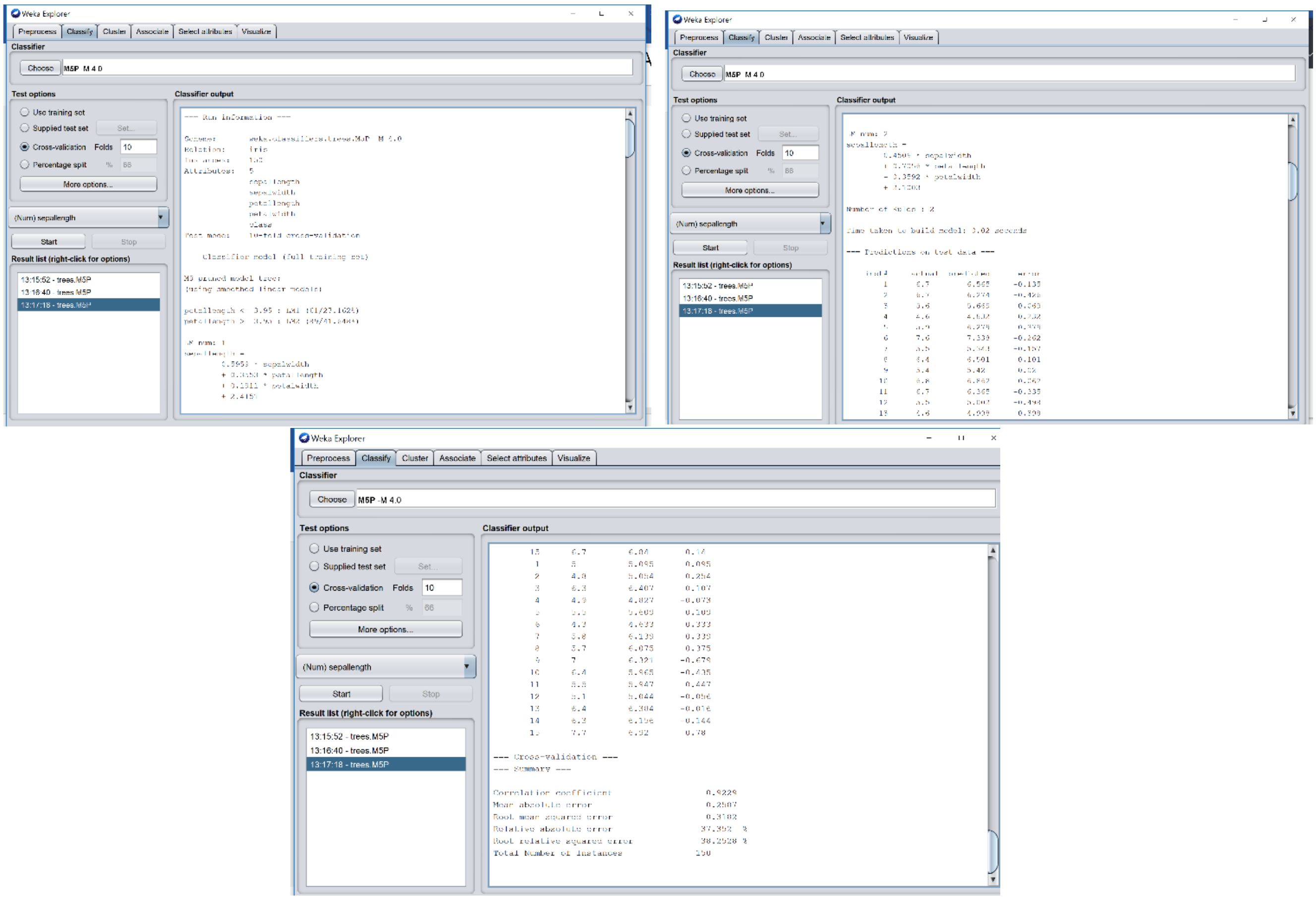
Random seed for X

□ Preserve order

□ Output source

Figure 2. Screenshot of Weka on how to print ‘output Predictions’.

(c) Run the M5P classifier with default parameters.



M5P -M 4.0

○ Use training set
○ Supplied test set   Set...
● Cross-validation   Folds 10
○ Percentage split   % 66
More options...

(Num) sepallength

```
13:15:52 - trees.M5P
13:16:40 - trees.M5P
13:17:18 - trees.M5P
```

```
=== Run information ===

Scheme:       weka.classifiers.trees.M5P -M 4.0
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    10-fold cross-validation

    Classifier model (full training set)

M5 pruned model tree:
(using smoothed linear models)

petallength <= 3.95 : LM1 (61/27.162%)
petallength >  3.95 : LM2 (89/41.848%)

LM num: 1
sepallength =
        0.5953 * sepalwidth
        + 0.3553 * petallength
        + 0.1311 * petalwidth
        + 2.4151
```

```
LM num: 2
sepallength =
        0.4505 * sepalwidth
        + 0.7256 * petallength
        - 0.3592 * petalwidth
        + 2.1003

Number of Rules : 2

Time taken to build model: 0.02 seconds

=== Predictions on test data ===

    inst#     actual  predicted      error
        1      6.7        6.565     -0.135
        2      6.7        6.274     -0.426
        3      5.6        5.665      0.065
        4      4.6        4.832      0.232
        5      5.9        6.278      0.378
        6      7.6        7.338     -0.262
        7      5.5        5.343     -0.157
        8      6.4        6.501      0.101
        9      5.4        5.42       0.02
       10      6.8        6.862      0.062
       11      6.7        6.365     -0.335
       12      5.5        5.002     -0.498
       13      4.6        4.908      0.308
```

```
       15      6.7        6.84       0.14
        1      5          5.095      0.095
        2      4.8        5.054      0.254
        3      6.3        6.407      0.107
        4      4.9        4.827     -0.073
        5      5.5        5.603      0.103
        6      4.3        4.633      0.333
        7      5.8        6.139      0.339
        8      5.7        6.075      0.375
        9      7          6.321     -0.679
       10      6.4        5.965     -0.435
       11      5.5        5.947      0.447
       12      5.1        5.044     -0.056
       13      6.4        6.384     -0.016
       14      6.3        6.156     -0.144
       15      7.7        6.92       0.78

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.9229
Mean absolute error                      0.2587
Root mean squared error                  0.3182
Relative absolute error                 37.352  %
Root relative squared error             38.2528 %
Total Number of Instances              150
```

Figure 3. Screenshots of a run of M5P classifier with default parameters.

(d) Examine the output. What do you think of the accuracy of the predictions?

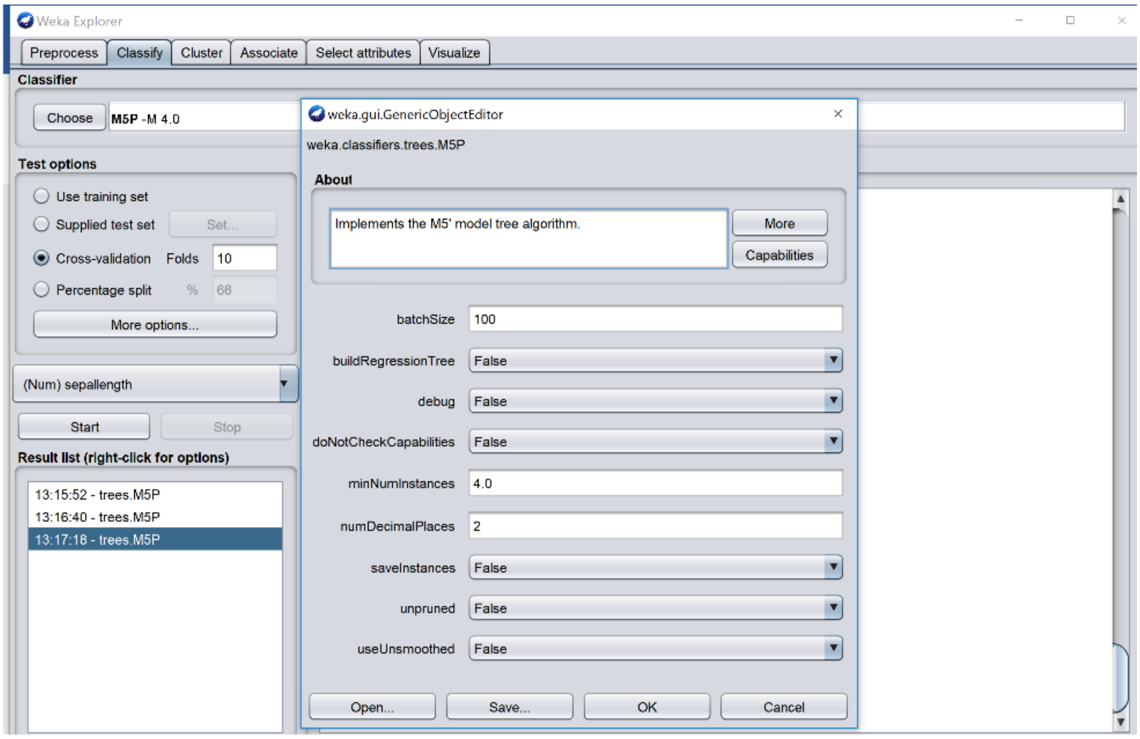
The error rate of M5-P classifier(classifier/ numeric predictor for numeric data set) has been displayed in weka as :

Correlation coefficient	0.9229
Mean absolute error	0.2587
Root mean squared error	0.3182
Relative absolute error	37.352%
Root relative squared error	38.2528%
Total Number of Instances	150

A correlation coefficient above 9 indicates a very strong correlation. In this case the 0.92 correlation between predicted and actual indicates good agreement which, indicates high accuracy. Also the values of petal length range from 4.3 to 7.9 with an average of about 6. An average error of +/- 0.25 when the target is 6 looks pretty good to me. The relative error looks surprisingly high, but we'll see late that Weka doesn't calculate it the way you might expect.

(e) Experiment with different values for the parameters. What is the effect on accuracy?

By left-click on the name of classifier as shown in figure you would get access to a menu of all parameters of the classifier where these parameters can be edited to see their effects on outputs of the classifier.



M5P -M 4.0

○ Use training set
○ Supplied test set
● Cross-validation Folds 10
○ Percentage split % 66

(Num) sepallength

13:15:52 - trees.M5P
13:16:40 - trees.M5P
13:17:18 - trees.M5P

weka.gui.GenericObjectEditor

weka.classifiers.trees.M5P

About

Implements the M5' model tree algorithm.

batchSize 100
buildRegressionTree False
debug False
doNotCheckCapabilities False
minNumInstances 4.0
numDecimalPlaces 2
saveInstances False
unpruned False
useUnsmoothed False

Figure 4. Screenshot of Weka on how to change parameters of M5-P.

Table1. Effect of varying batch-size on running time and error rate of M5-P classifier.

Batchsize	Time taken to build model (second)	Error rate (relative absolute error)
1000	0.01	37.352
500	0	37.352
200	0.01	37.352
100	0.02	37.352
50	0	37.352
30	0.01	37.352
20	0	37.352
10	0	37.352
5	0	37.352
2	0.01	37.352
1	0	37.352

As it can be seen from table 1, changing batchzise does not influence error rate of the classifier while slightly changes time taken on building model.

Table 2. Effect of varying the “M” parameter on running time and error rate of M5-P classifier.

“M” parameter	Time taken to build model (seconds)	Error rate (relative absolute error)
1	0.01	37.352
2	0.01	37.352
3	0.01	37.352
4	0.01	37.352
5	0.01	37.352
6	>0.01	37.352
7	>0.01	37.352
8	>0.01	37.352
10	>0.01	37.352
15	>0.01	37.6953
20	>0.01	37.8761
30	>0.01	38.504

Table 2 displays results from changing the “M” parameter (ie. Altering the minimum number of instances in the tree’s leaves). The effect on model build time is minor yet the larger the “M” parameter the faster the model can be built. The relative absolute error remains stable at around 37.352 until the “M” parameter increases beyond 10, whereby we see the error rate increase.

(f) Experiment with ZeroR and IBK and their various parameters.

Table 3. Effect of varying batchsize and K on accuracy and running time of zeroR and IBK.

Classifier Model	**ZeroR**		**IBK**							
			K=1		**K=5**		**K=50**		**K=100**	
Batch size	Time taken (second)	Error rate (%relative absolute)	Time taken (second)	Error rate (%relative absolute)	Time taken (second)	Error rate (%relative absolute)	Time taken (second)	Error rate (%relative absolute)	Time taken (second)	Error rate (%relative absolute)
1	0	100	0	45.5808	0	42.6264	0	57.6746	0	74.0479
5	0	100	0	45.5808	0	42.6264	0	57.6746	0	74.0479
10	0	100	0	45.5808	0	42.6264	0	57.6746	0	74.0479
20	0	100	0	45.5808	0	42.6264	0	57.6746	0	74.0479
50	0	100	0	45.5808	0	42.6264	0	57.6746	0	74.0479
100	0	100	0	45.5808	0	42.6264	0	57.6746	0	74.0479
200	0	100	0	45.5808	0	42.6264	0	57.6746	0	74.0479
500	0	100	0	45.5808	0	42.6264	0	57.6746	0	74.0479

As it can be seen from table 2, changing batchsize does not have any affect on accuracy and running time of IBK and zeroR classifiers. However, accuracy of IBK is still significantly higher than zeroR. Increasing value of K in IBK first leads to improvement in accuracy. However if K is too big the accuracy suffers.

(g) Build a table of classifier, parameter values and error. What combination gives the most accurate predictions?

Looking at table 1 and comparing these limited number of runs based on varying batchsize and K parameters, IBK with k=5 gives the most accurate prediction. However we don’t know yet whether there is a better value for k between 1 and 50.

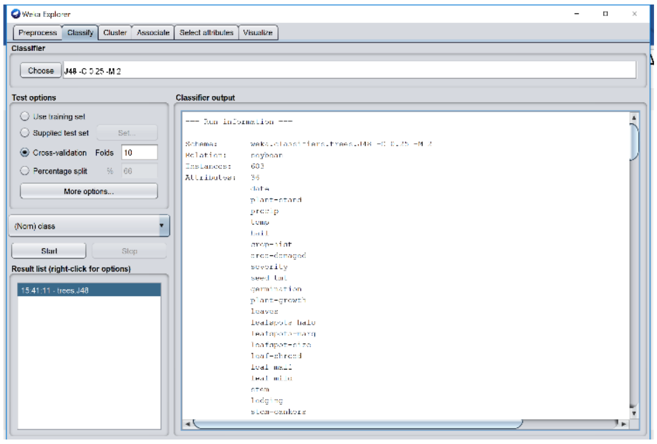
(h) Can you explain the differences in errors?

The more complex model is more accurate.

3. Repeat the previous exercise with cpu.with.vendor.arff

4. Load the file soybean.arff

(a) Run the J48 classifier with default parameters.



J48 -C 0.25 -M 2

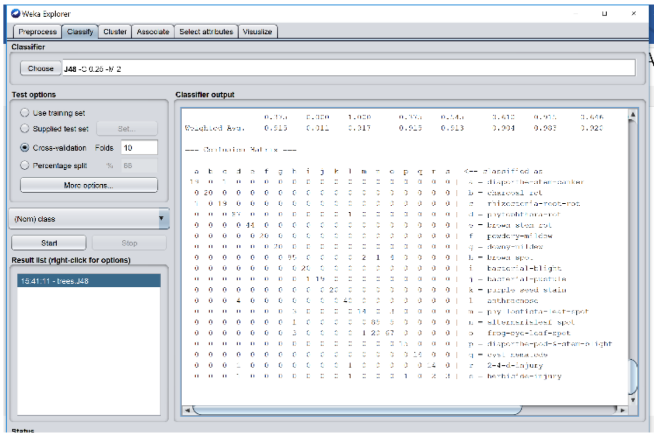
○ Use training set
○ Supplied test set
● Cross-validation Folds 10
○ Percentage split % 66

(Nom) class

15:41:11 - trees.J48

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     soybean
Instances:    683
Attributes:   36
              date
              plant-stand
              precip
              temp
              hail
              crop-hist
              area-damaged
              severity
              seed-tmt
              germination
              plant-growth
              leaves
              leafspots-halo
              leafspots-marg
              leafspot-size
              leaf-shread
              leaf-malf
              leaf-mild
              stem
              lodging
              stem-cankers
```



J48 -C 0.25 -M 2

○ Use training set
○ Supplied test set
● Cross-validation Folds 10
○ Percentage split % 66

(Nom) class

15:41:11 - trees.J48

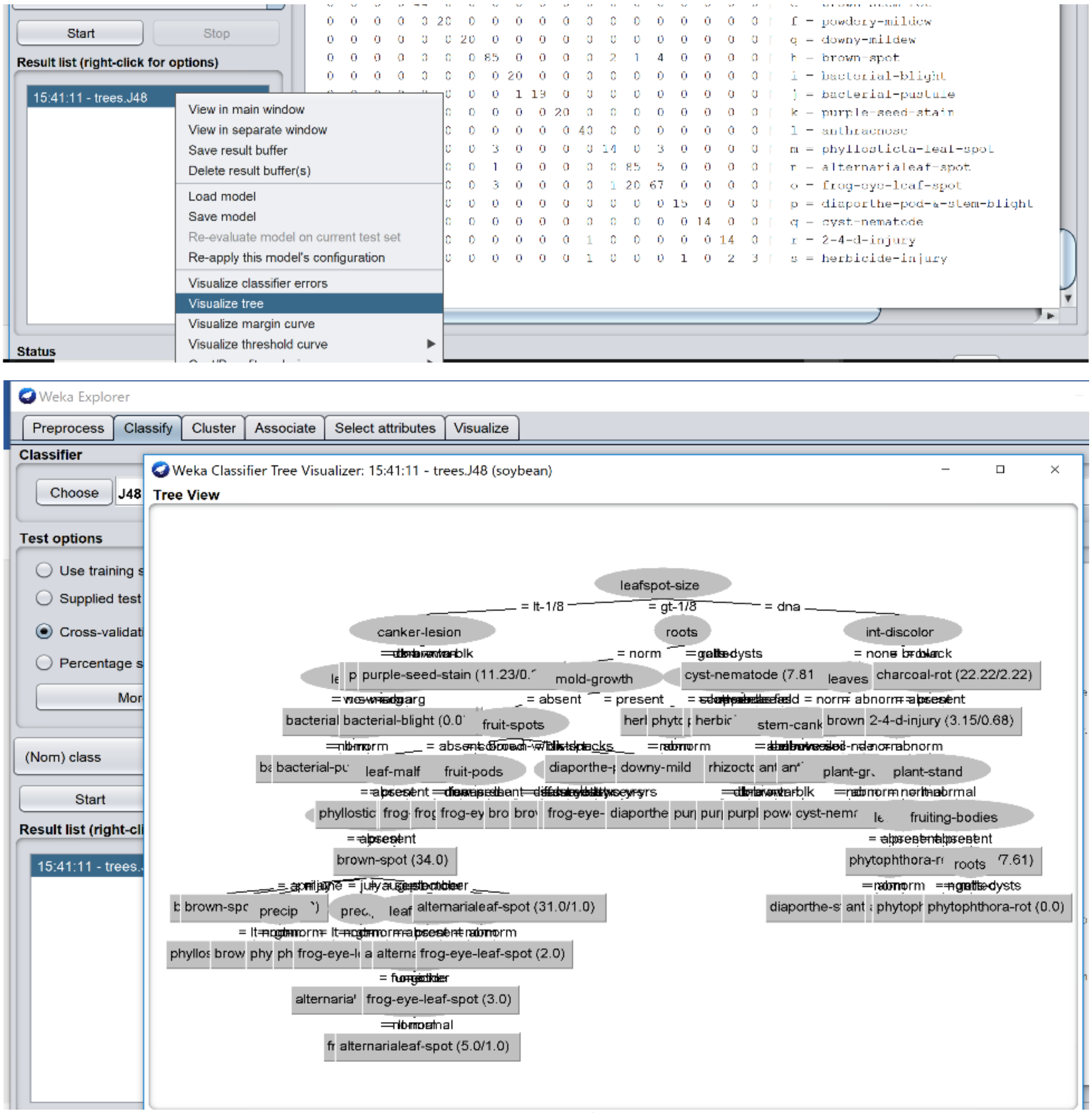
```
                 0.375    0.000    1.000      0.375    0.545      0.612    0.911     0.646
Weighted Avg.    0.915    0.011    0.917      0.915    0.913      0.904    0.983     0.920

=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s   <-- classified as
 19  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  a = diaporthe-stem-canker
  0 20  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  b = charcoal-rot
  1  0 19  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  c = rhizoctonia-root-rot
  0  0  0 87  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0 |  d = phytophthora-rot
  0  0  0  0 44  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  e = brown-stem-rot
  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0  0  0 |  f = powdery-mildew
  0  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0  0 |  g = downy-mildew
  0  0  0  0  0  0  0 85  0  0  0  0  2  1  4  0  0  0  0 |  h = brown-spot
  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0 |  i = bacterial-blight
  0  0  0  0  0  0  0  0  1 19  0  0  0  0  0  0  0  0  0 |  j = bacterial-pustule
  0  0  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0  0 |  k = purple-seed-stain
  0  0  0  4  0  0  0  0  0  0  0 40  0  0  0  0  0  0  0 |  l = anthracnose
  0  0  0  0  0  0  0  5  0  0  0  0 14  0  1  0  0  0  0 |  m = phyllosticta-leaf-spot
  0  0  0  0  0  0  0  1  0  0  0  0  0 85  5  0  0  0  0 |  n = alternarialeaf-spot
  0  0  0  0  0  0  0  3  0  0  0  0  1 20 67  0  0  0  0 |  o = frog-eye-leaf-spot
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 15  0  0  0 |  p = diaporthe-pod-&-stem-blight
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 14  0  0 |  q = cyst-nematode
  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0 14  0 |  r = 2-4-d-injury
  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  1  0  2  3 |  s = herbicide-injury
```

Figure5. Screenshots of running J48 classifier on soybean data set.

(b) Make sure you can visualise the tree.



```
 0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0  0  0 |  f = powdery-mildew
 0  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0  0 |  g = downy-mildew
 0  0  0  0  0  0  0 85  0  0  0  0  2  1  4  0  0  0  0 |  h = brown-spot
 0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0 |  i = bacterial-blight
                   0  0  0  1 19  0  0  0  0  0  0  0  0  0 |  j = bacterial-pustule
                   0  0  0  0  0 20  0  0  0  0  0  0  0  0 |  k = purple-seed-stain
                   0  0  0  0  0  0 40  0  0  0  0  0  0  0 |  l = anthracnose
                   0  0  3  0  0  0  0 14  0  3  0  0  0  0 |  m = phyllosticta-leaf-spot
                   0  0  1  0  0  0  0  0 85  5  0  0  0  0 |  n = alternarialeaf-spot
                   0  0  3  0  0  0  0  1 20 67  0  0  0  0 |  o = frog-eye-leaf-spot
                   0  0  0  0  0  0  0  0  0  0 15  0  0  0 |  p = diaporthe-pod-&-stem-blight
                   0  0  0  0  0  0  0  0  0  0  0 14  0  0 |  q = cyst-nematode
                   0  0  0  0  0  0  1  0  0  0  0  0 14  0 |  r = 2-4-d-injury
                   0  0  0  0  0  0  1  0  0  0  1  0  2  3 |  s = herbicide-injury
```

Figure 6. Screenshots of visualising J48 tree.

(c) Experiment with different values of the C and M parameters.

Table 4. Effects of varying M and C parameters on accuracy and running time of J48.

C	M	Time Taken	Accuracy
0.25	2	0.03	91.5081
	3	0	89.3119
	5	0	89.6047
	10	0	84.4802
	20	0	78.4773
	50	0	54.3192
1	2	0.63	91.5081
	3	0.56	89.6047
	5	0.44	88.1406 %
	10	0.28	84.7731
	20	0.19	78.4773
	50	0.05	54.3192

10	2	0.72	91.5081
	3	0.6	89.6047
	10	0.43	88.1406
	20	0.13	78.4773
	50	0.02	54.3192

As shown in table 3, accuracy of J48 is decreasing by increasing M, while increasing confidence C does not have any affect on accuracy of classifier. Therefore, the best combination of c and M is when M = 2 and c = 0.25.

(d) Using percentage split build a table of training and test errors

#	**Percentage Split**	**Test Accuracy**	**Train Accuracy**
1	**%66**	90.5172	96.3397
2	**%40**	85.3659	96.3397
3	**%80**	90.5109	96.3397

(e) Is there any overfitting?

In all runs, accuracy of training is greater than test error therefore there is overfitting.

(f) What would you say is the best combination of parameter values

The best combination of c and M is when M = 2 and c = 0.25.

5. Repeat the previous exercise with the glass.arff

6. Experiment with other classifiers and data files.