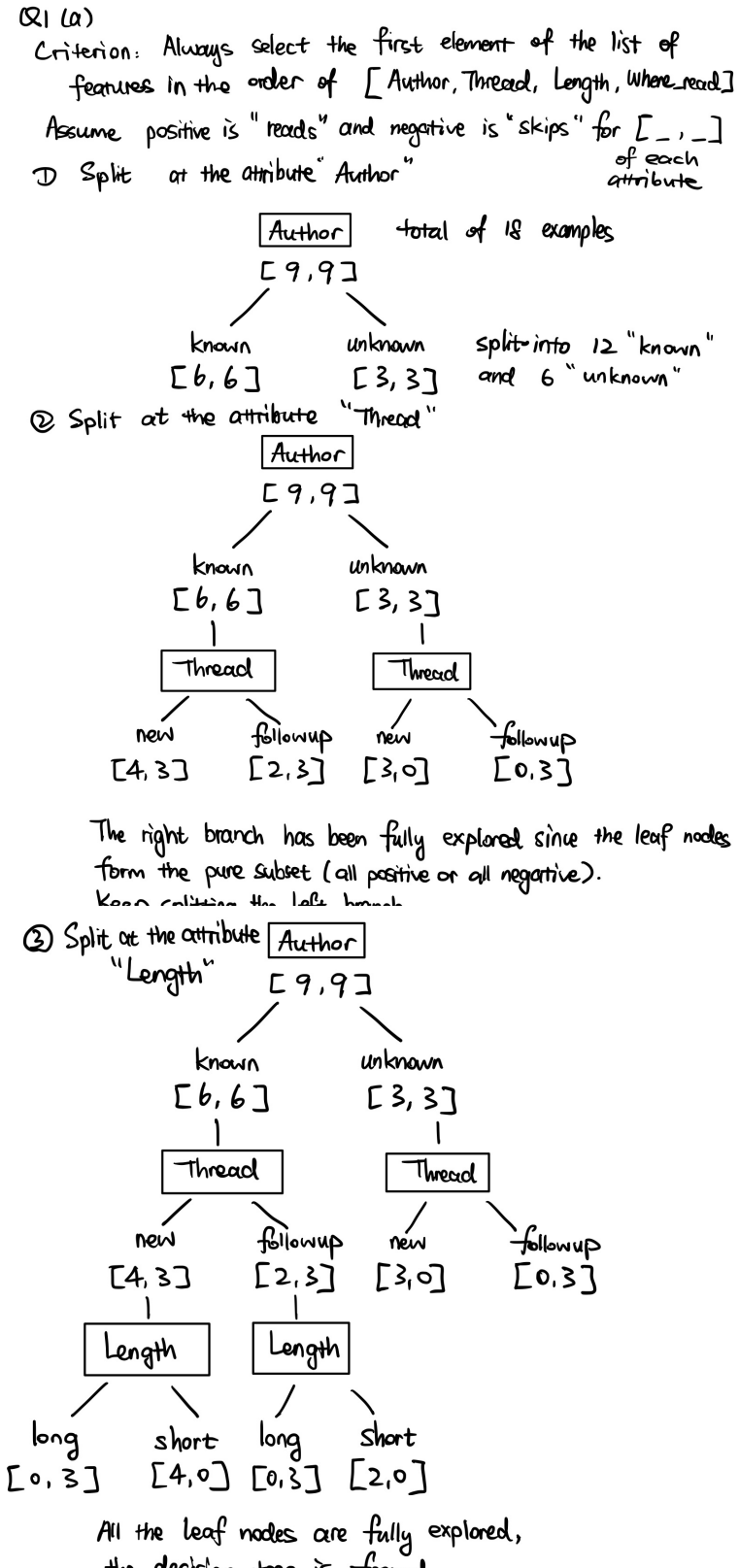
COMP3411-21T1 Assignment 2 Machine Learning

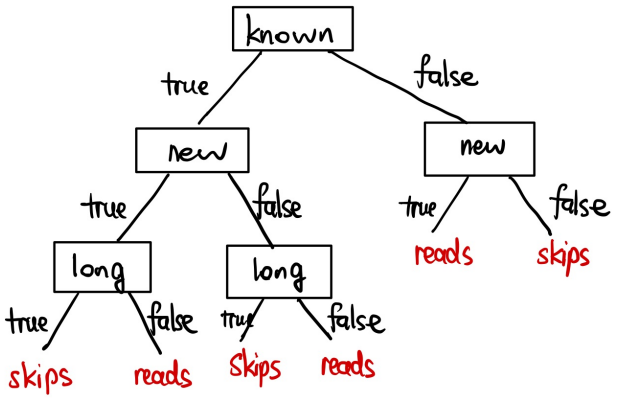
Part 1: Decision Trees

Question 1.1

(a) Suppose you change the algorithm to always select the first element of the list of features. What tree is found when the features are in the order [Author, Thread, Length, WhereRead]? Does this tree represent a different function than that found with the maximum information gain split? Explain.



④ Redraw it to correspond to the form in Figure 7.6



The stopping criterion of all of the examples having the same classification which is either "reads" or "skips" is reached for the above decision tree. The decision tree in Figure 7.6 is showed as below:

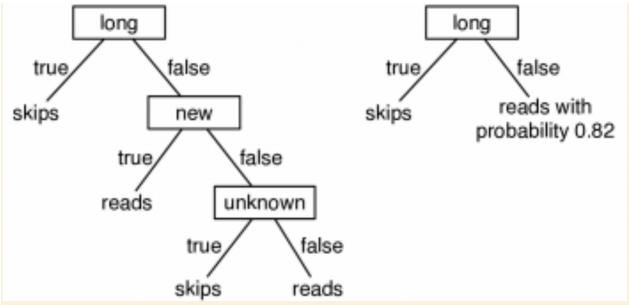
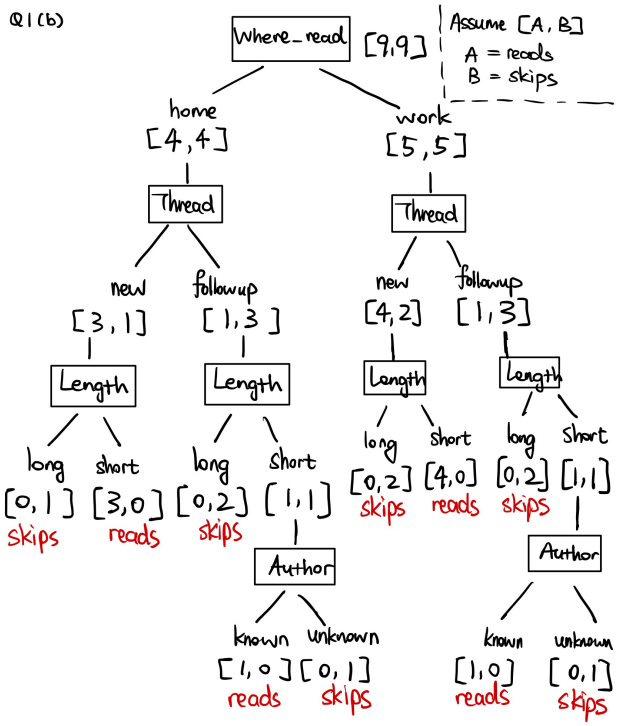
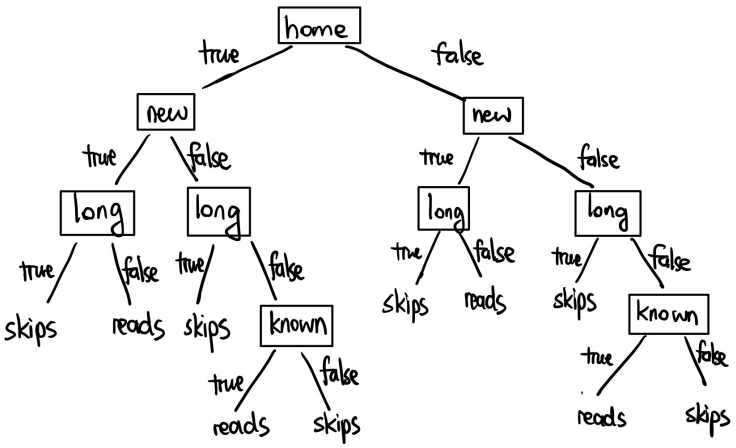


Figure 7.6: Two decision trees

To identify whether the select-the-first-element decision tree represents a different function or not, e19 and e20 were used to compare the outcome from the two decision trees. In e20, two decision trees produced the same outcome, "skips". However, the select-the-first-element tree gave "reads" and the maximum-information-gain tree get "skips" as the output in e19.
Therefore, it is evident that the two decision trees represent different functions due to the different outputs produced.

(b) What tree is found when the features are in the order [WhereRead, Thread, Length, Author]? Does this tree represent a different function than that found with the maximum information gain split or the one given for the preceding part? Explain.





Summaries the rules:

① $\begin{cases} new \wedge long \rightarrow skips, \text{ simplifies as } long \rightarrow skips \\ \neg new \wedge long \rightarrow skips \end{cases}$

② $new \wedge \neg long \rightarrow reads$

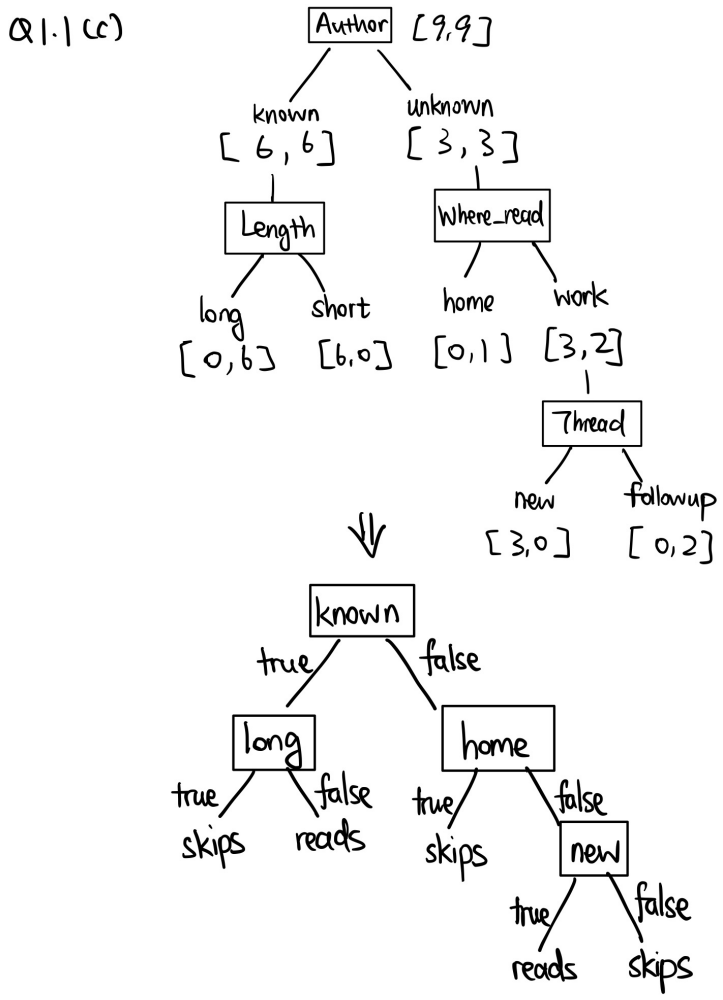
③ $\neg new \wedge \neg long \wedge known \rightarrow reads$

④ $\neg new \wedge \neg long \wedge \neg known \rightarrow skips$

After summarising then generalising the rules in the above decision tree, it is clear that the maximum-information-gain decision tree follows these rules as well. Hence it is evident that they have the same function. However, the decision tree in part (a) does not follow the third and fourth rules above. Therefore, it has a different function than the tree above. Functions are the same if they produce the same outputs for all inputs. To test the validity of the conclusion, we suppose a new set which is different from all of the training and testing sets in Figure 7.1, e.g. "Author = unknown, Thread = new, Length = long, Where_read = home". As a result, the tree in part (a) produces "reads" as our output whereas the trees in Figure 7.6 and part (b), generate "skips" as the outcome.

c) Is there a tree that correctly classifies the training examples but represents a different function than those found by the preceding algorithms? If so, give it. If not, explain why.

In order to build a decision tree which can correctly classify the training set but represents a different function, we need to select an order of the attributes that is different from the orders in the previous question. Based on the rules obtained in question 1.1 (b), for all examples that has "long" as "Length", the class is always "skips". It is concluded that the tree should not have the "Length" attribute as the last attribute of the order since it is too definitive. In addition, making the left and right branch exploring different attributes would be a good direction of approach. After a lot of trail and errors, the tree is formed as showed below.

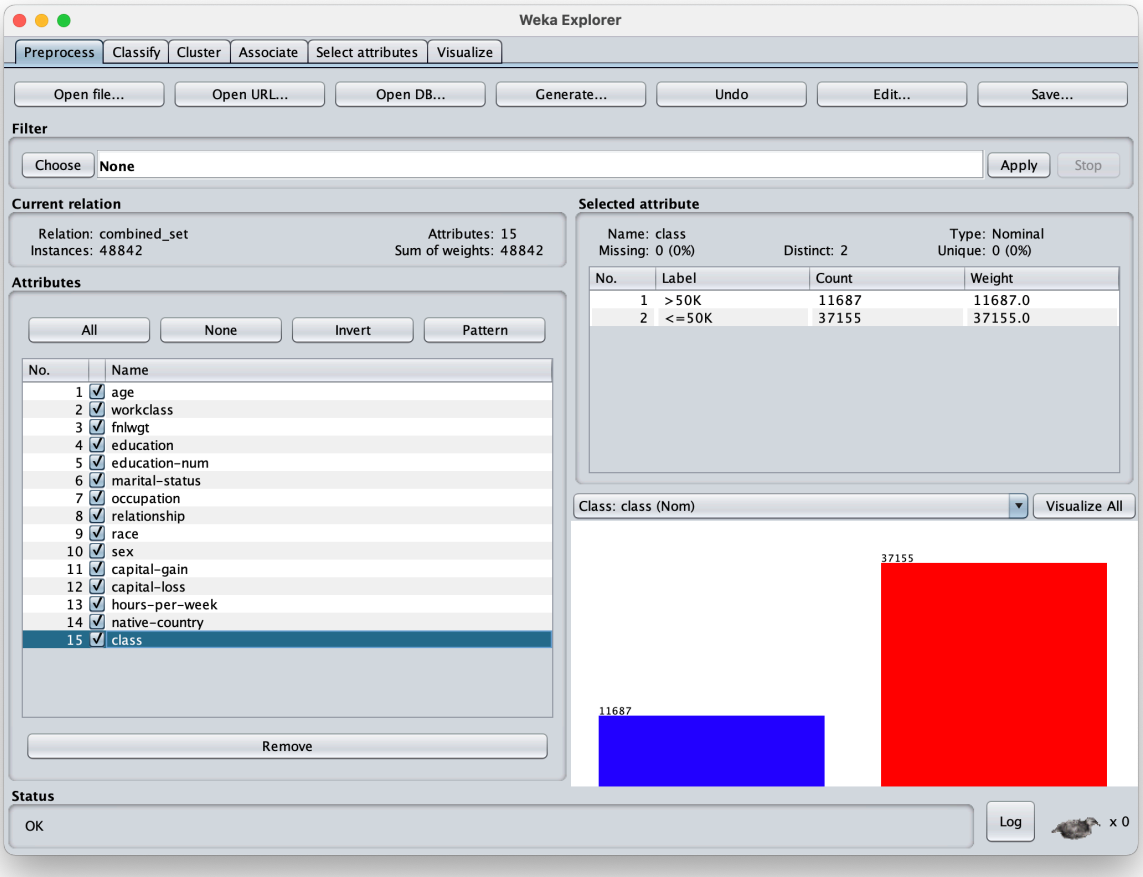


This decision tree correctly classifies all the training examples, but passing in the inputs, which is not in the training set, as [*author=unknown, thread=new, length=short, where_read=home*], this tree outputs "skips" whereas the tree in (a) and (b) classifies as "reads". This indicates this decision tree has a different function than those found by the preceding algorithms.

Question 1.2

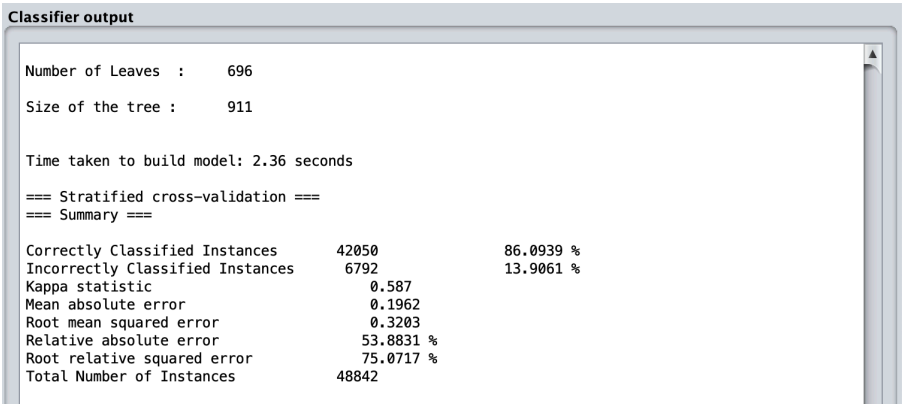
Steps/methods taken:

Since Weka gives the user options to either upload training and testing data separately or use percentage split and cross validation, I planned to test the differences of using both methods. Firstly I converted the adult.data and adult.test into separate arff files, following the instructions in the Weka documentation pages. Then I combined the training data in adult.data and testing data in adult.test into a new arff file. The attributes information to build an arff file were acquired from the adult.names file in the provided adult data directory. Next, I imported the combined data file to Weka Explorer, which is demonstrated below.



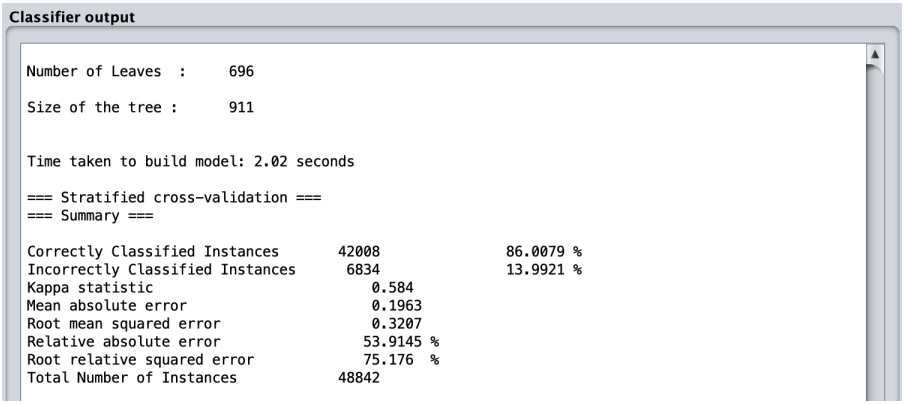
After importing the data, I clicked Classify tab and selected J48 as the classifier. I kept the default parameters unchanged for now but aimed to test whether different extent of cross-validation and percentage split affects the accuracy of the resulting pruned decision tree.

The result for doing a 10 fold cross-validation classification is showed below:



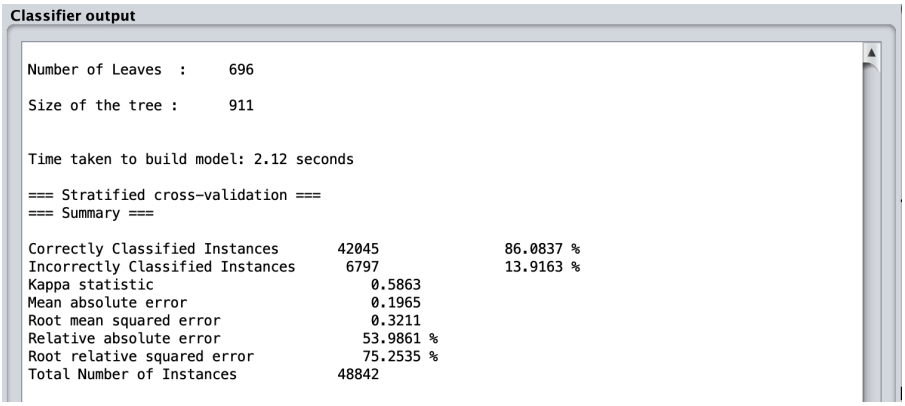
```
Classifier output

Number of Leaves  :     696

Size of the tree :      911


Time taken to build model: 2.36 seconds

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

Correctly Classified Instances       42050               86.0939 %
Incorrectly Classified Instances      6792               13.9061 %
Kappa statistic                          0.587
Mean absolute error                      0.1962
Root mean squared error                  0.3203
Relative absolute error                 53.8831 %
Root relative squared error             75.0717 %
Total Number of Instances            48842
```

15 fold cross-validation:



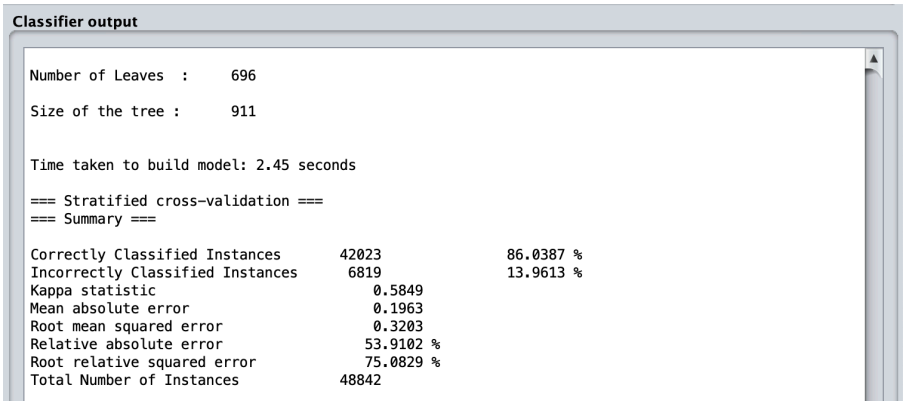
```
Classifier output

Number of Leaves  :     696

Size of the tree :      911


Time taken to build model: 2.02 seconds

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

Correctly Classified Instances       42008               86.0079 %
Incorrectly Classified Instances      6834               13.9921 %
Kappa statistic                          0.584
Mean absolute error                      0.1963
Root mean squared error                  0.3207
Relative absolute error                 53.9145 %
Root relative squared error             75.176  %
Total Number of Instances            48842
```

20 fold cross-validation:



```
Classifier output

Number of Leaves  :     696

Size of the tree :      911


Time taken to build model: 2.12 seconds

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

Correctly Classified Instances       42045               86.0837 %
Incorrectly Classified Instances      6797               13.9163 %
Kappa statistic                          0.5863
Mean absolute error                      0.1965
Root mean squared error                  0.3211
Relative absolute error                 53.9861 %
Root relative squared error             75.2535 %
Total Number of Instances            48842
```

50 fold cross-validation:



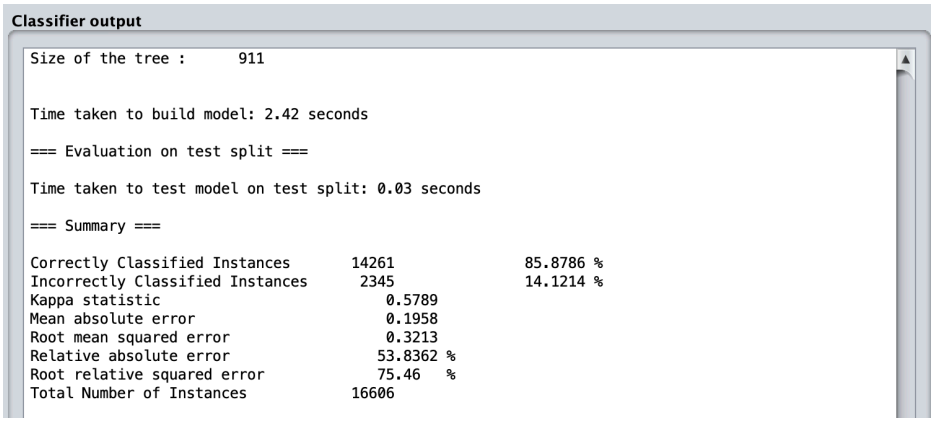
```
Classifier output

Number of Leaves  :     696

Size of the tree :      911


Time taken to build model: 2.45 seconds

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

Correctly Classified Instances       42023               86.0387 %
Incorrectly Classified Instances      6819               13.9613 %
Kappa statistic                          0.5849
Mean absolute error                      0.1963
Root mean squared error                  0.3203
Relative absolute error                 53.9102 %
Root relative squared error             75.0829 %
Total Number of Instances            48842
```

	10 fold cross-validation	15 fold cross-validation	20 fold cross-validation	50 fold cross-validation
Correctly Classified Instances	86.0939%	86.0079%	86.0837%	86.0387%

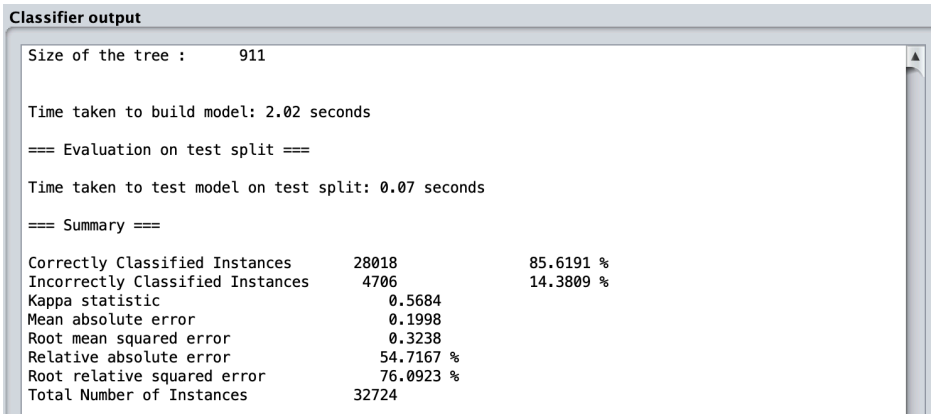
The results show the extent of cross-validation has little effect on the accuracy of a pruned decision tree.

Next, I tested the effect of changing the percentage split on the default pruned decision tree.

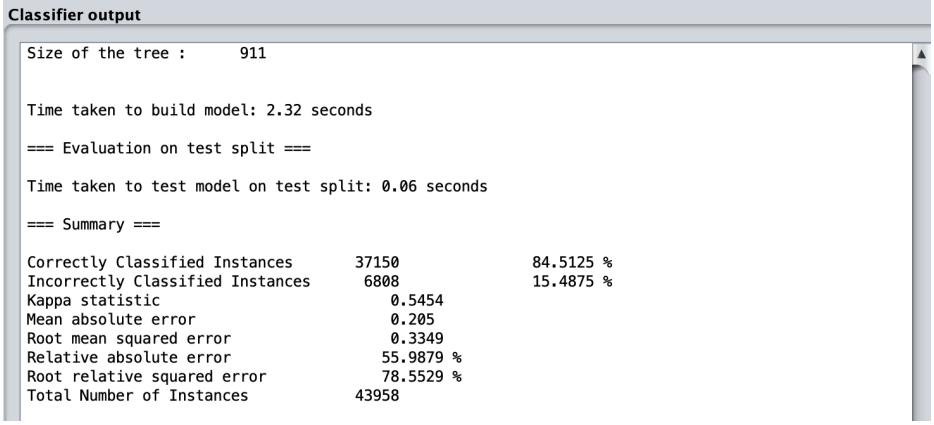
66% percentage split:



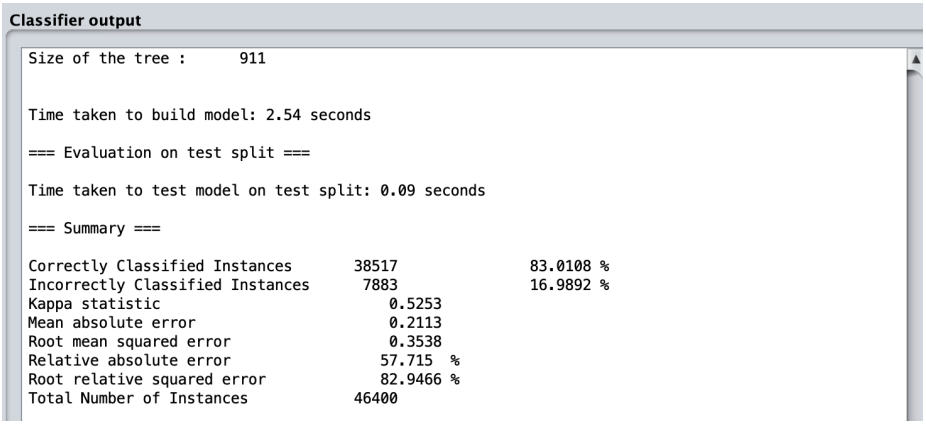
33% percentage split:



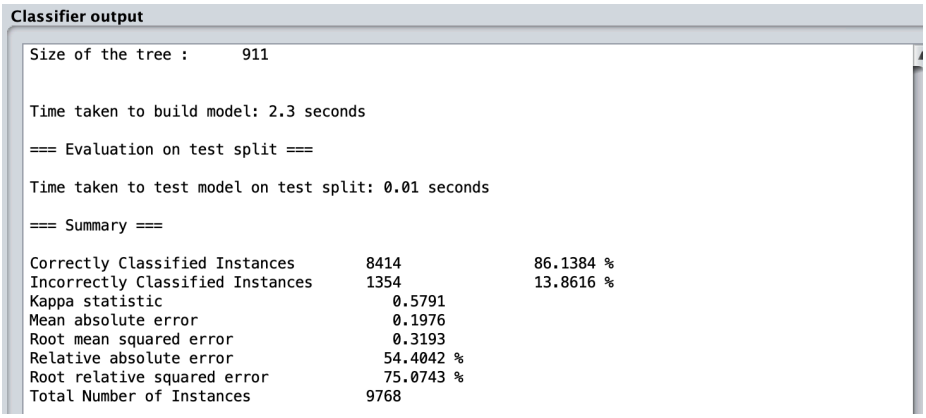
10% percentage split:



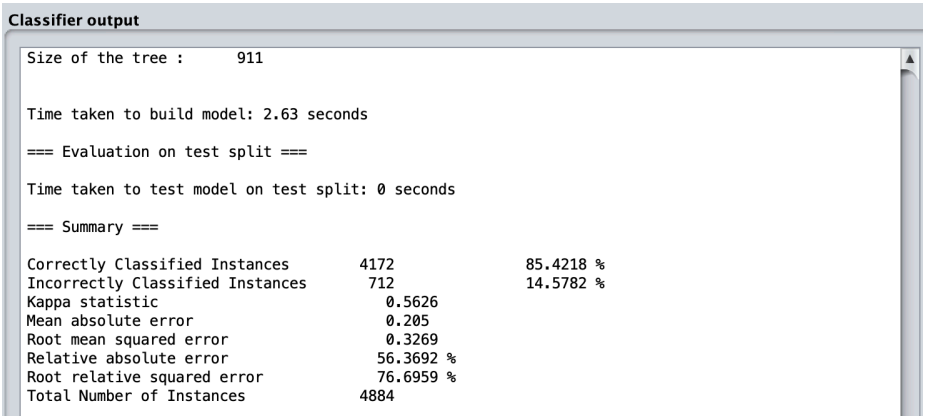
5% percentage split:



80% percentage split:



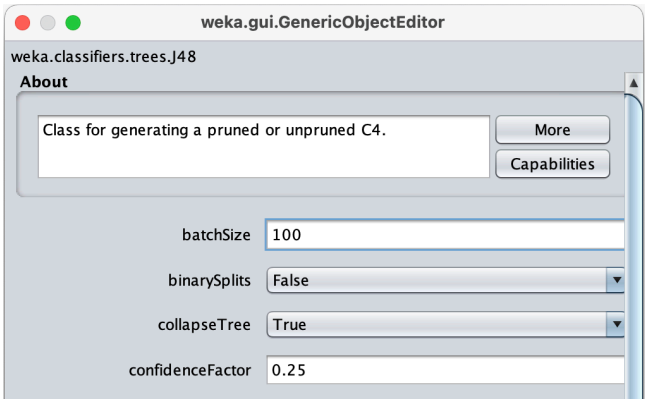
90% percentage split:



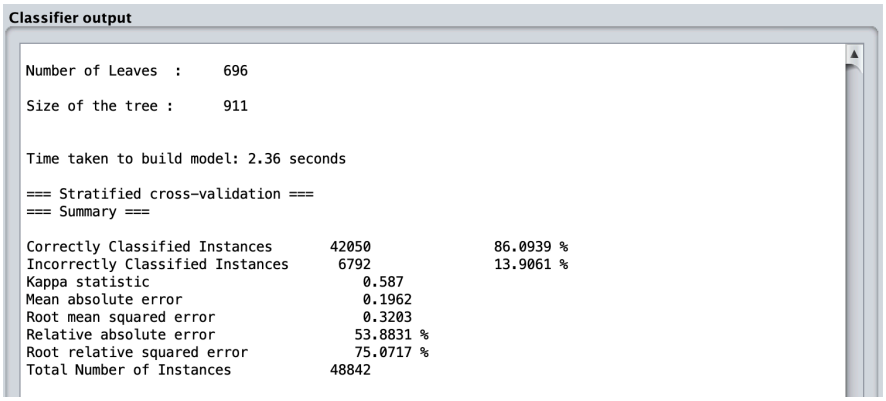
	5% split	10% split	33% split	66% split	80% split	90% split
Correctly Classified Instances	83.0108%	84.5125%	85.6191%	85.8786%	86.1384%	85.4218%

The table shows that the resultant decision tree has a higher accuracy when performing the percentage split of an approximate range of 66% to 80%. If the percentage split is too high it indicates almost all of the data is used to train the tree but there is not much validation for the built tree, which would reduce the accuracy. If the percentage split is too low this implies not much data is used to train the tree but too much data is used for testing, which would clearly cost the accuracy of the performance.

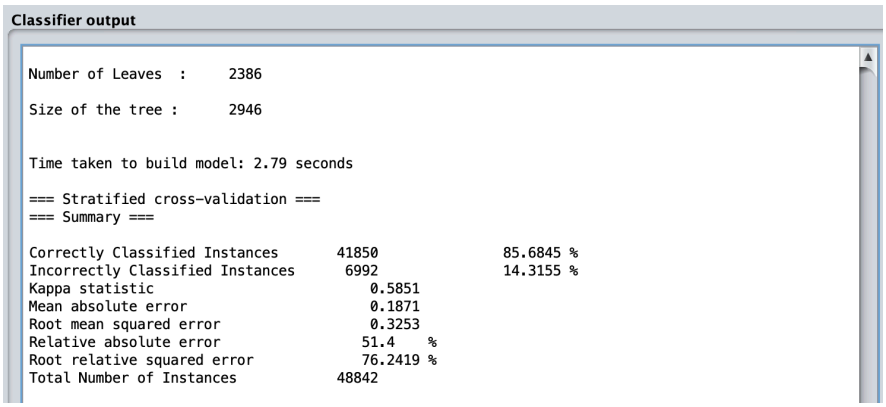
Then I aimed to test the effect of confidence factor on the accuracy of the performance of the tree. I kept Weka to perform 10 fold cross validation but modified the confidenceFactor option in the J48 settings.



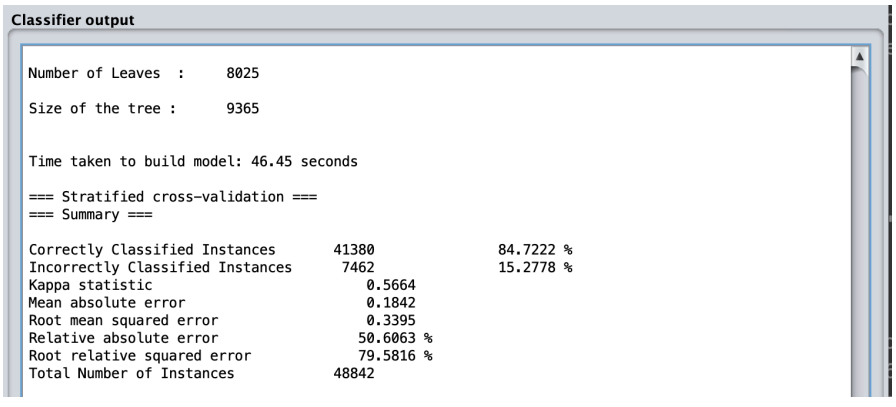
Confidence factor of 0.25:



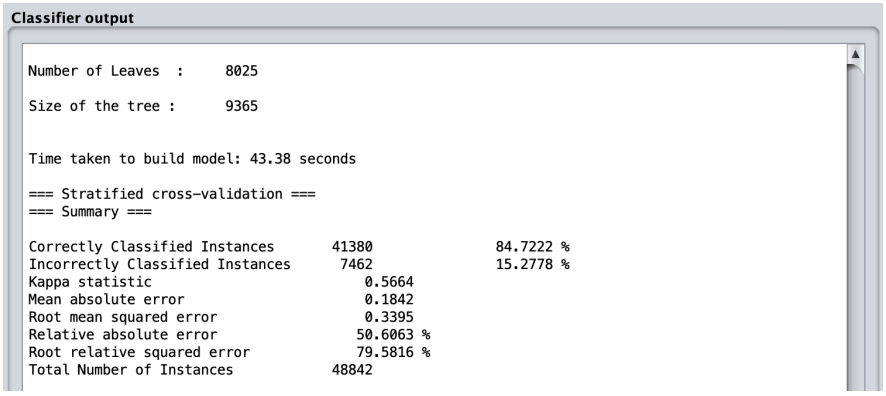
Confidence factor of 0.5:



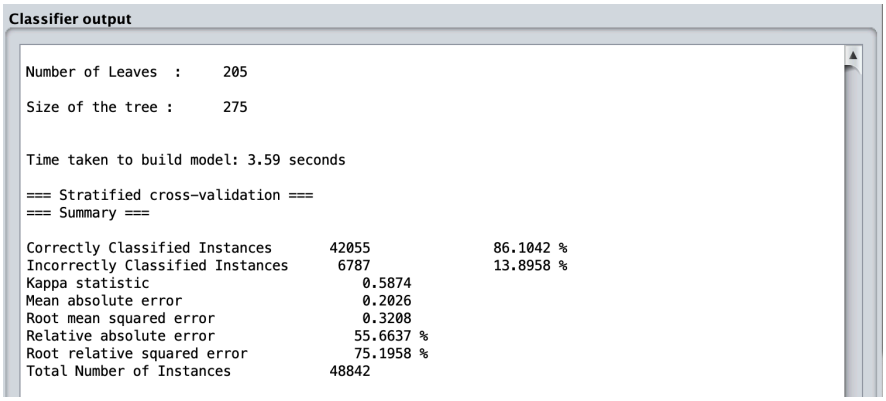
Confidence factor of 0.75:



Confidence factor of 0.99:

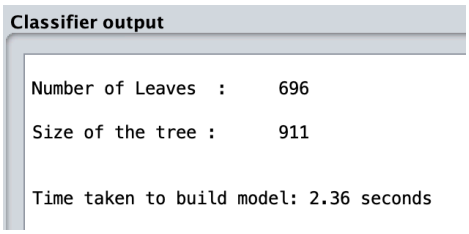


Confidence factor of 0.1:

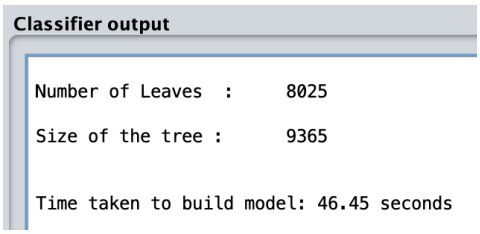


Confidence factor	0.1	0.25	0.5	0.75	0.99
Correctly Classified Instances	86.1042%	86.0939%	85.6845%	84.7222%	84.7222%

The table above showcase the relationship between the accuracy and the confidence factor. The higher the confidence factor, the more unpruned the decision tree is, hence the classification is less accurate. In addition, it was also concluded that the unpruned decision trees result in much larger trees which are harder to understand and take longer to test and validate. As shown below, the tree was approximately 10 times bigger. Therefore, it is better to select low confidence factors which makes the tree having high accuracy while preventing overfitting.



Output using the confidence factor 0.25



Output using the confidence factor 0.75

In addition, the training and testing set were used separately by selecting "Supplied test set" option and importing the test set. Then the confidence factor was modified to test the accuracy of the classification on the test set. The results are showcased below:

Confidence factor	0.1	0.25	0.5	0.75	0.99
Correctly Classified Instances	86.2171%	85.8485%	85.3265%	84.2209%	84.2209%

Using training and test set separately results in the same trend as using the combined set. The difference in accuracy between using a separate test set and using a combined set is small, approximately within 0.5%.

Therefore, I have experienced in the workflow of model building in Weka and practised how to perform importing and cleaning data, splitting the data into train/test or cross-validation sets, preprocessing, transformations and feature engineering.