**Assignment 6**

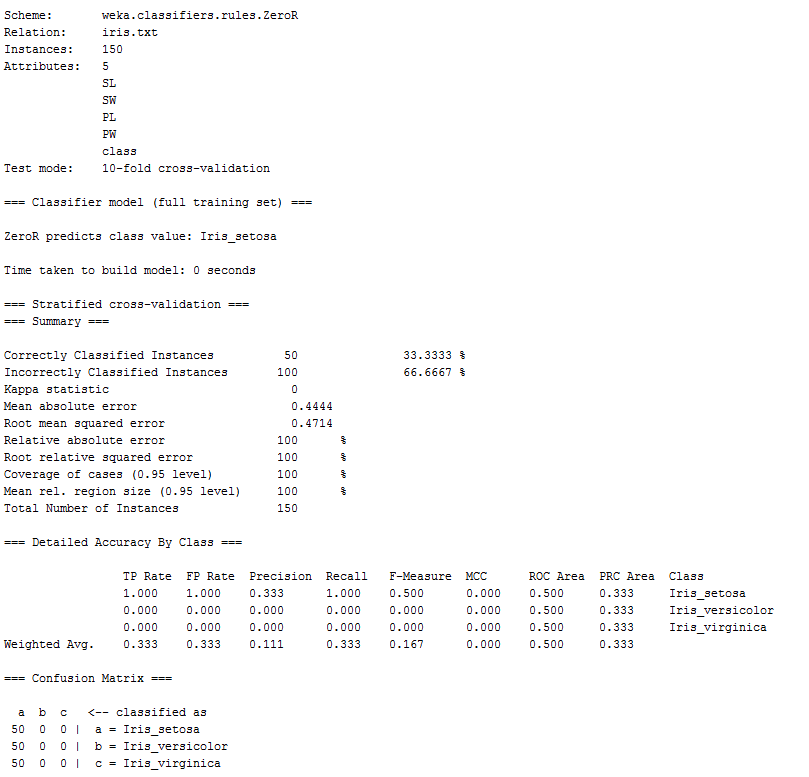
**Katherine Rodgers and John Merranko**

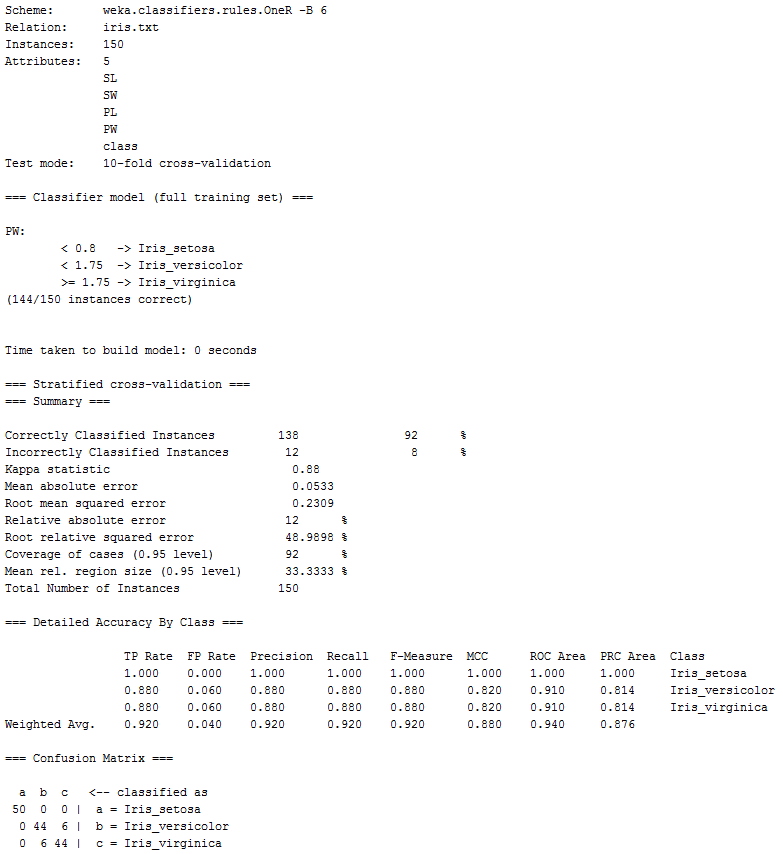
**Iris Data**

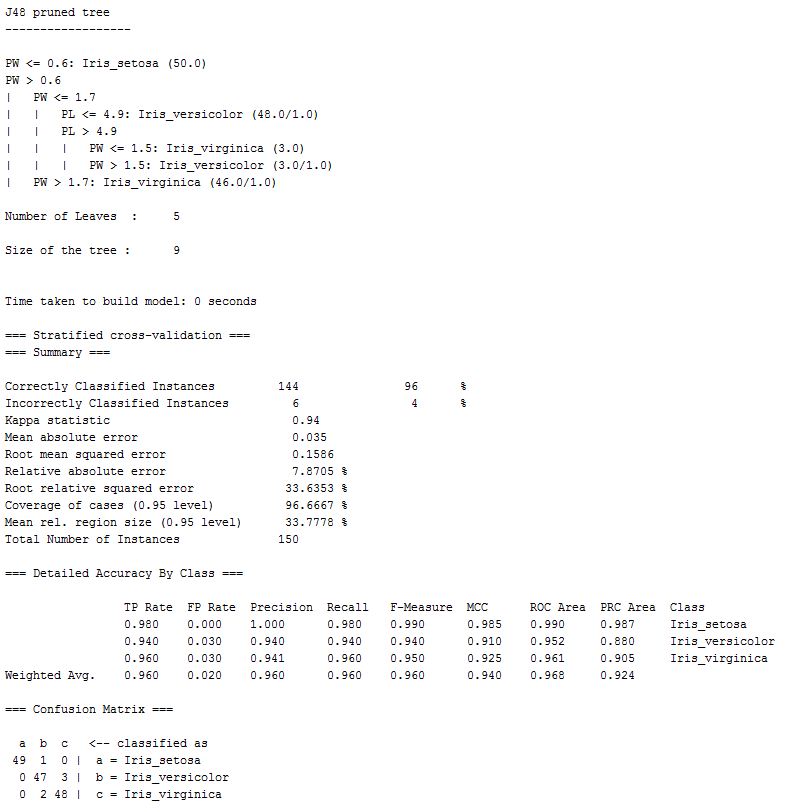
From the Irvine Machine Leaning Repository for this data set, the attribute information is as follows:

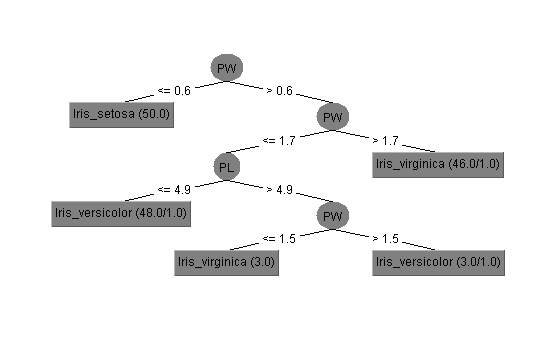
1. sepal length(SL) in cm    
2. sepal width(SW) in cm   
3. petal length(PL) in cm   
4. petal width(PW) in cm   
5. class:  Iris Setosa, Iris Versicolor, Iris Virginica

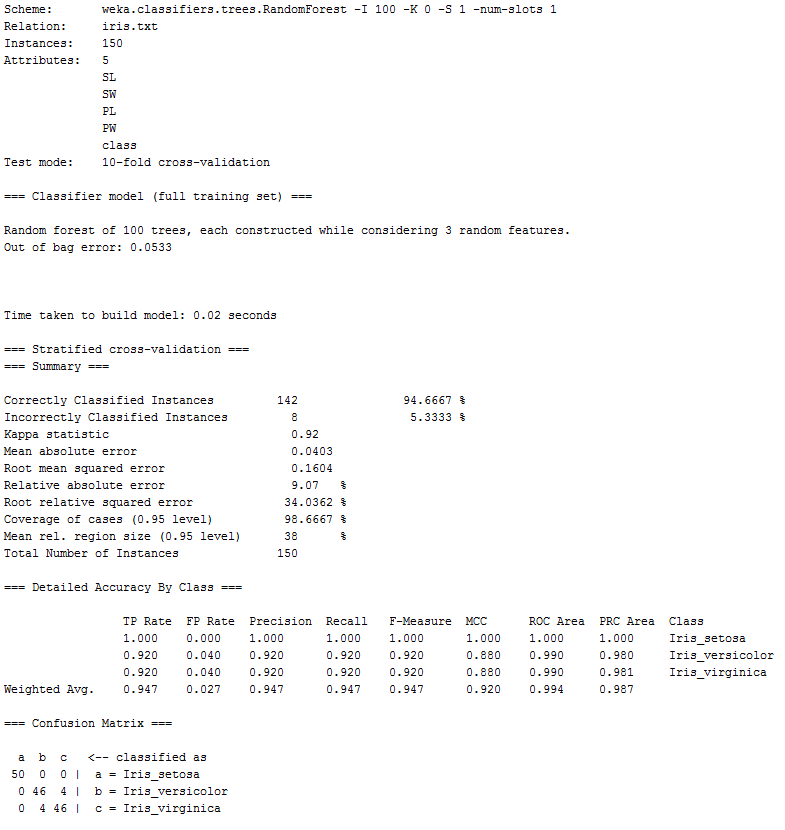
We loaded iris.arff into the Weka program and decided to run a few baseline classification models without doing anything to the data just to see how good the models would be. We decided on 2 algorithms under rules, zeroR and oneR, and 2 under trees, J48 and RandomForest. We used the 10-fold cross validation test option for the models. Here are the results:





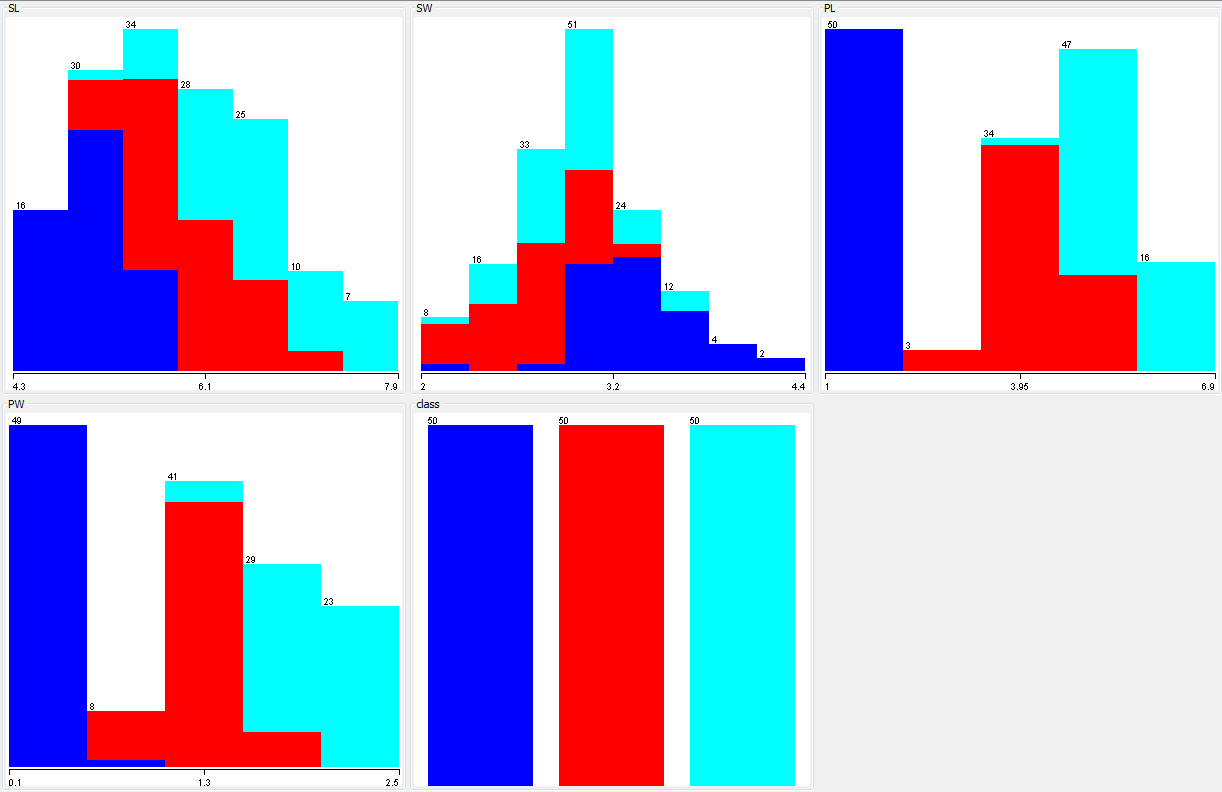




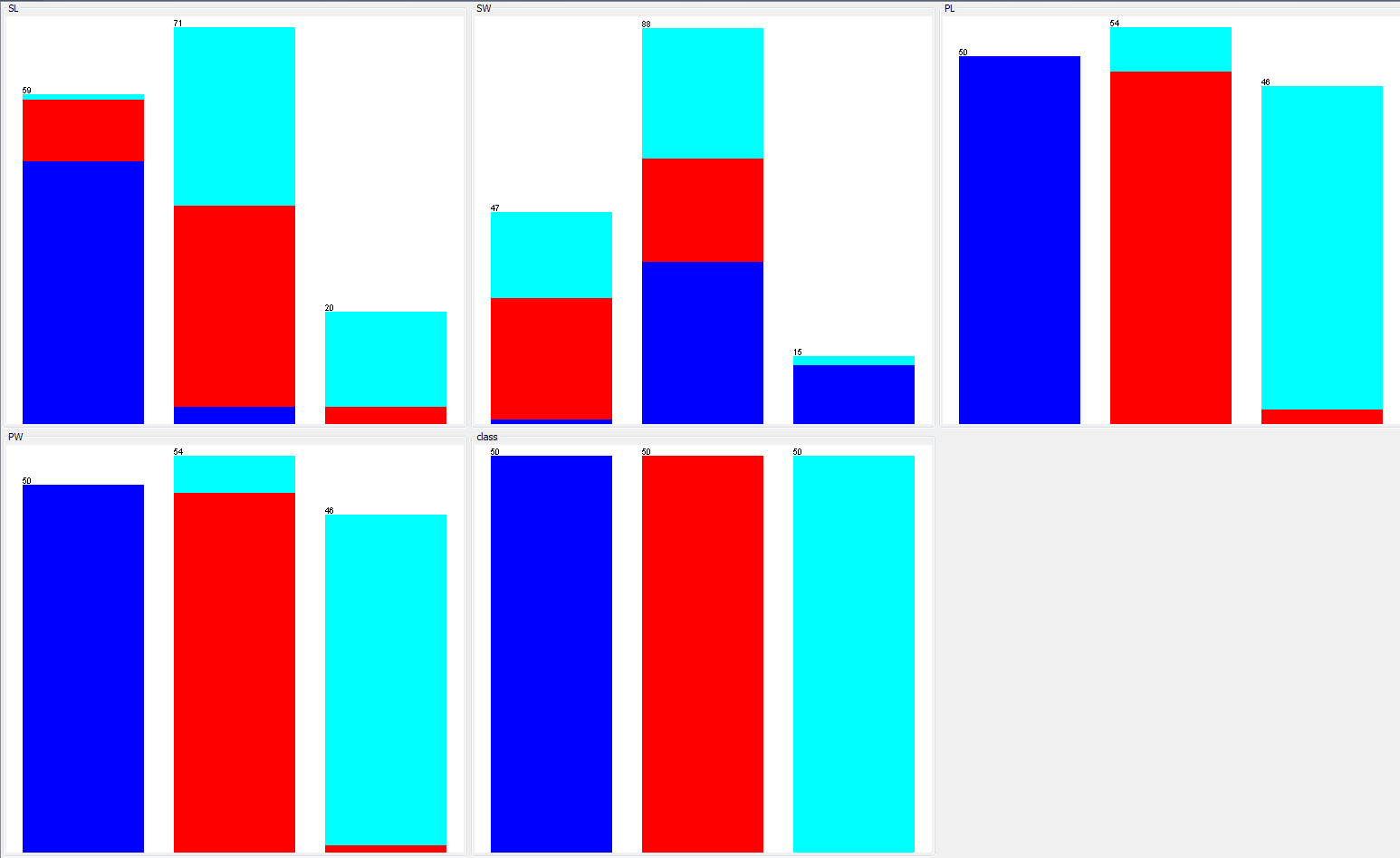


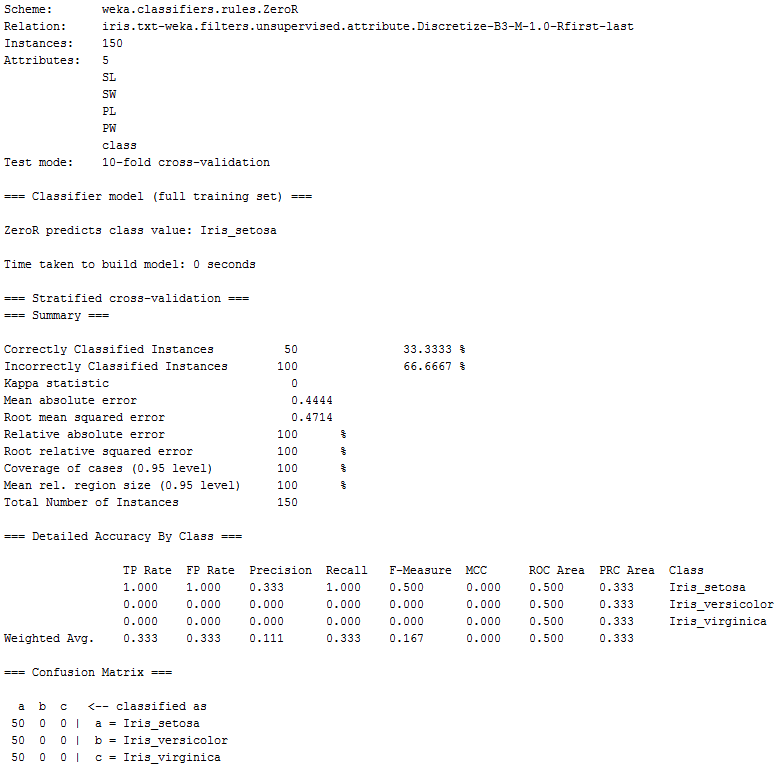
Comparing the accuracy from the weighted average F-measure, we see that zeroR performed much worse than the others. Also, both tree methods performed better than the rules classifier oneR.

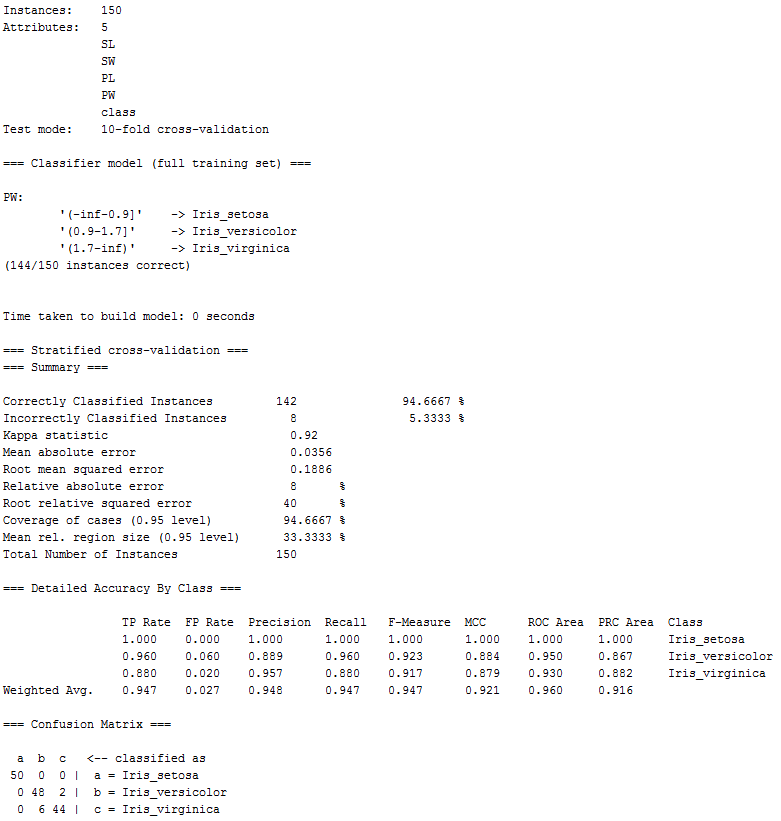
We decided at this point to look at the data. The following is the histogram visualizations the program displayed when the data was loaded:

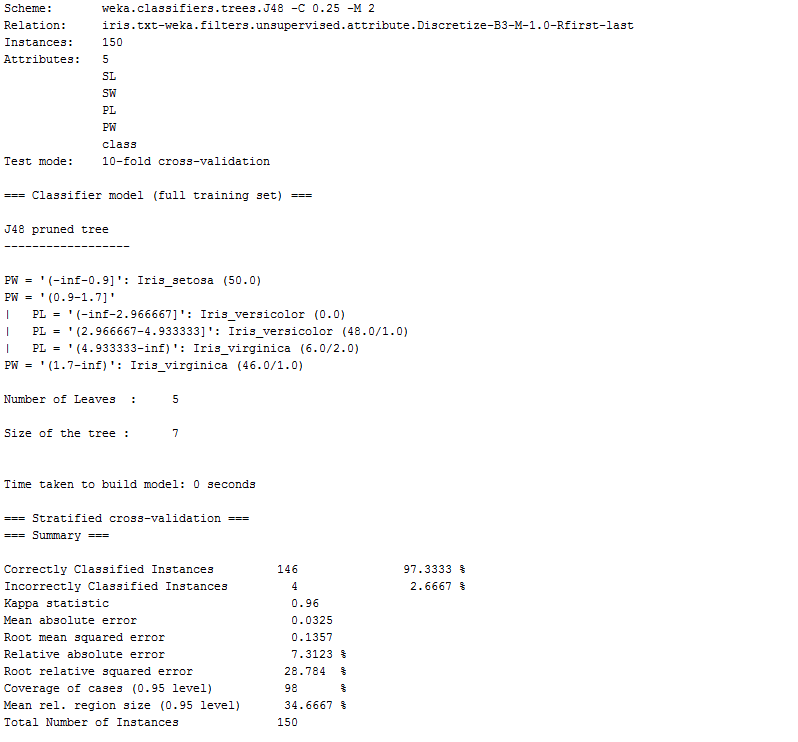


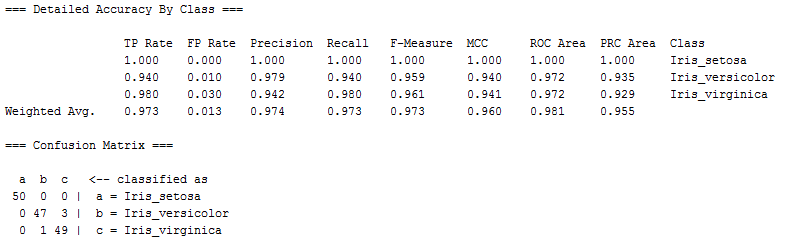
From this data we can see that there are 50 instances of each iris type. Also, the data is very interrelated with SW and somewhat interrelated with SL. PL gives a clear cut for the blue color (Iris Setosa). Additionally PW is almost a clear cut for blue. SL, PL, and PW are ordered in that Iris Setosa falls on the lower end of the measurement scale, followed by the majority of red(Iris Versicolor) falling in the middle, and then Iris Virginica at the end while moving up the scale. Because of this we decided to discretize the numeric attributes in the dataset into nominal attributes using 3 bins. Our data visualizations changes and re-run models are as follows:

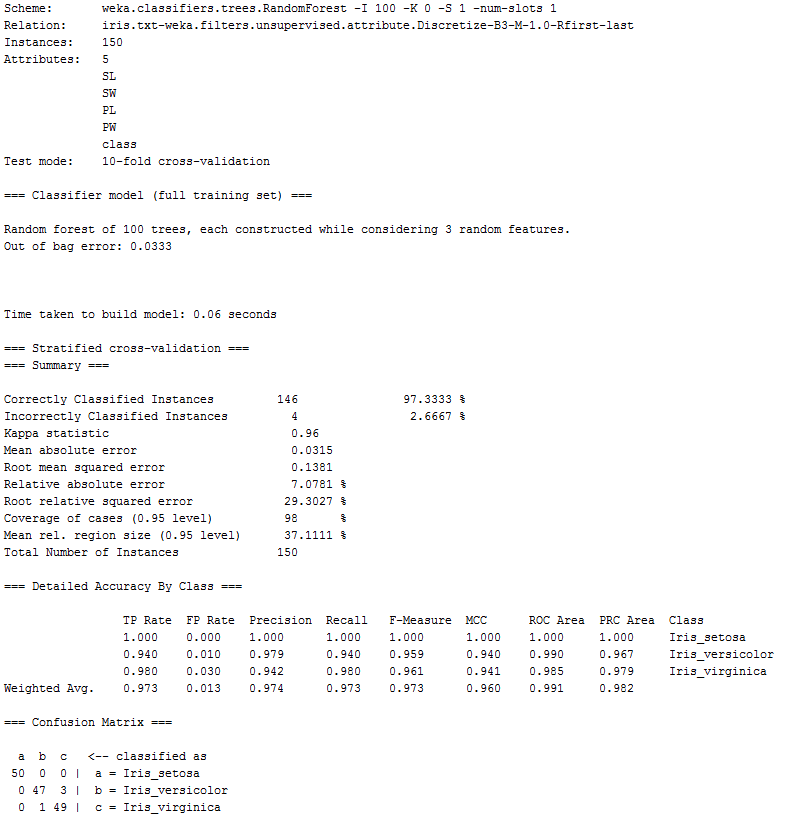




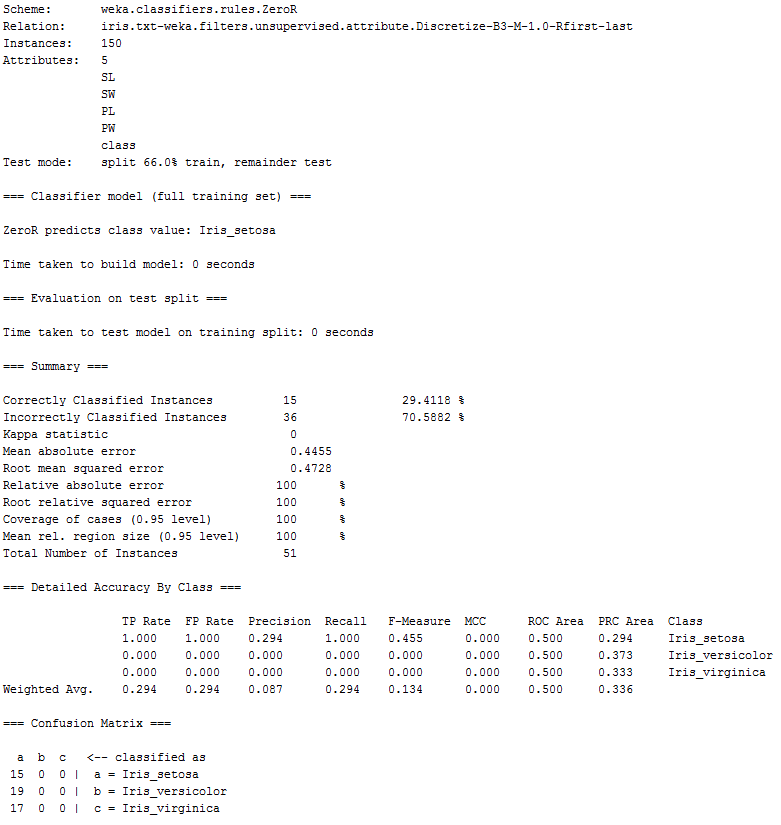


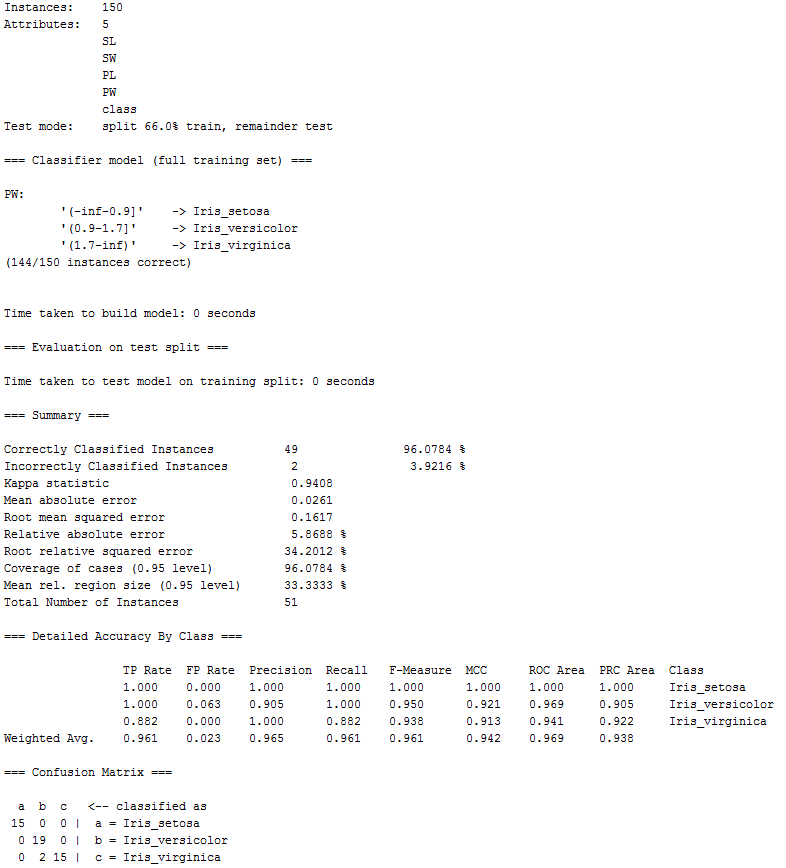


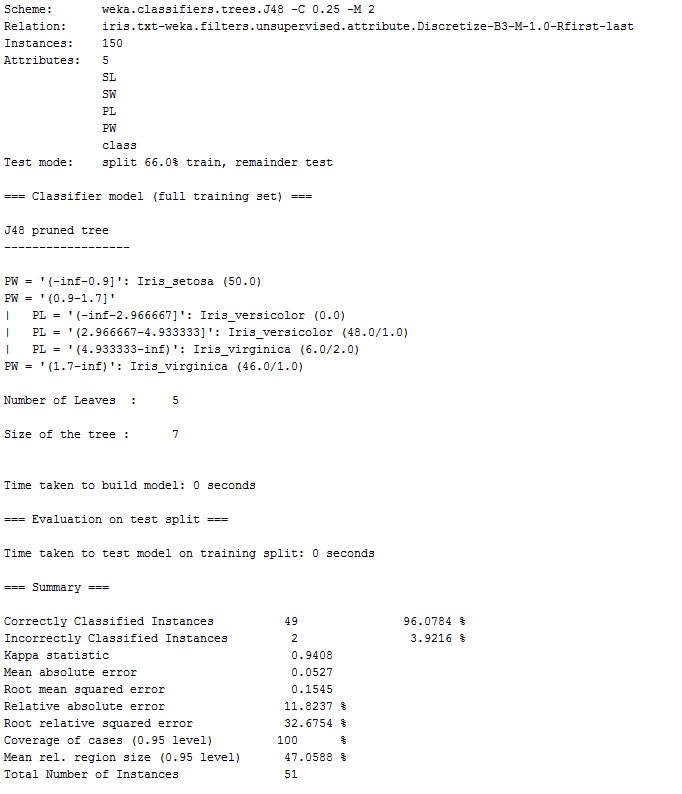


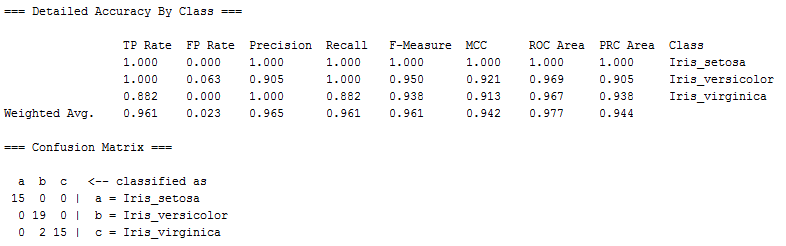


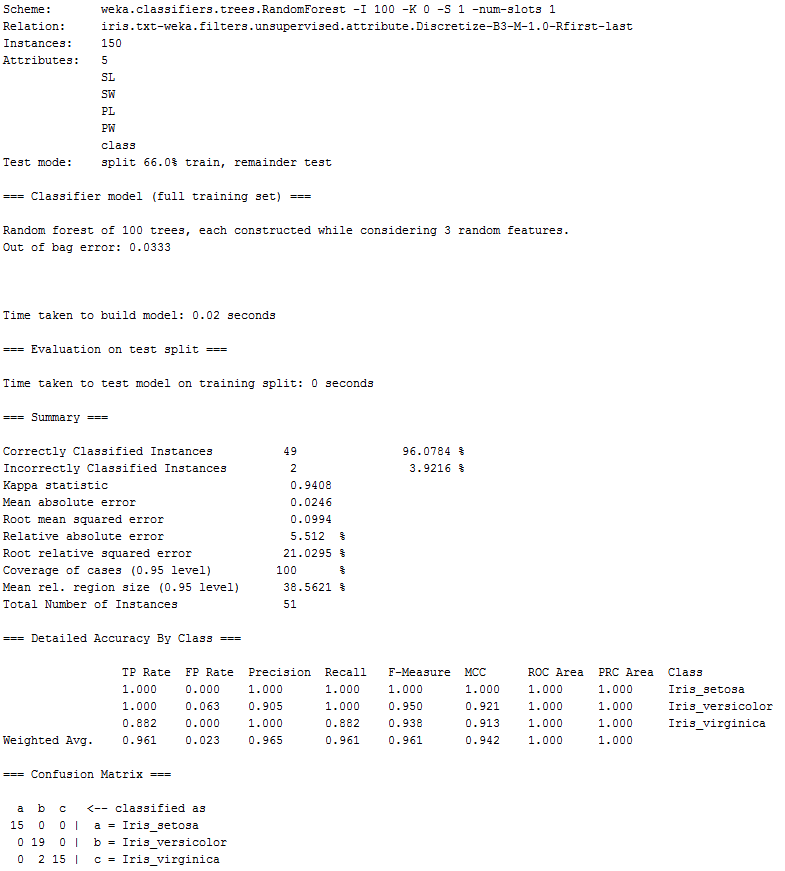
From comparison, discretizing our data had no impact on zeroR but increased the accuracy of the other three models, with RandomForest and J48 both having the highest F-measure at 0.973 with only 4 incorrectly classified instances. We messed around with some of parameters offered within the models themselves, but none changed the overall accuracy. We then re-ran the models by changing the test option to percentage split at 66% which indicates two-thirds training set and one-third test set. Here are the results:





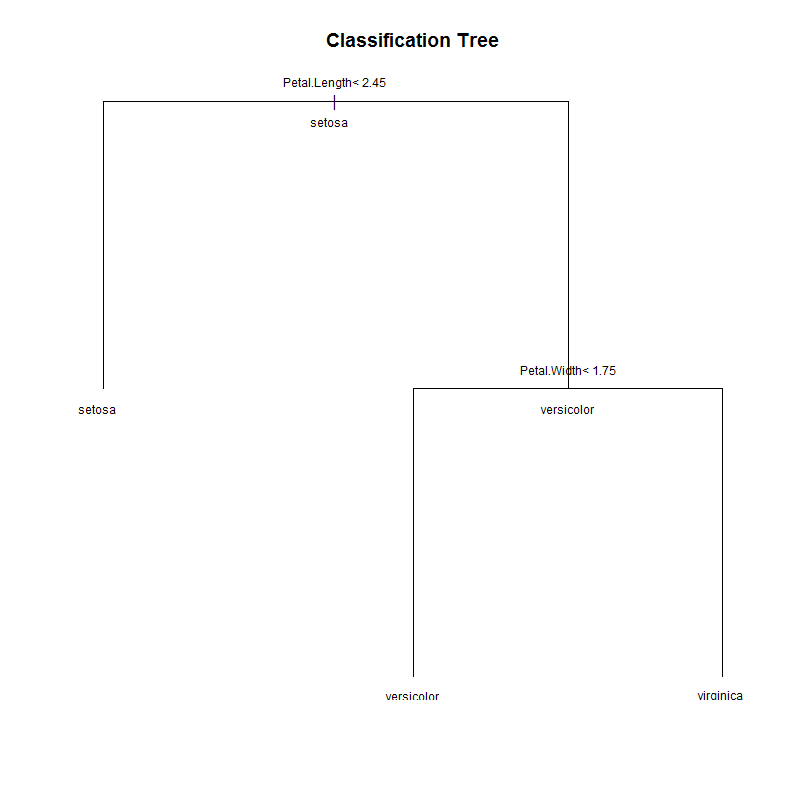




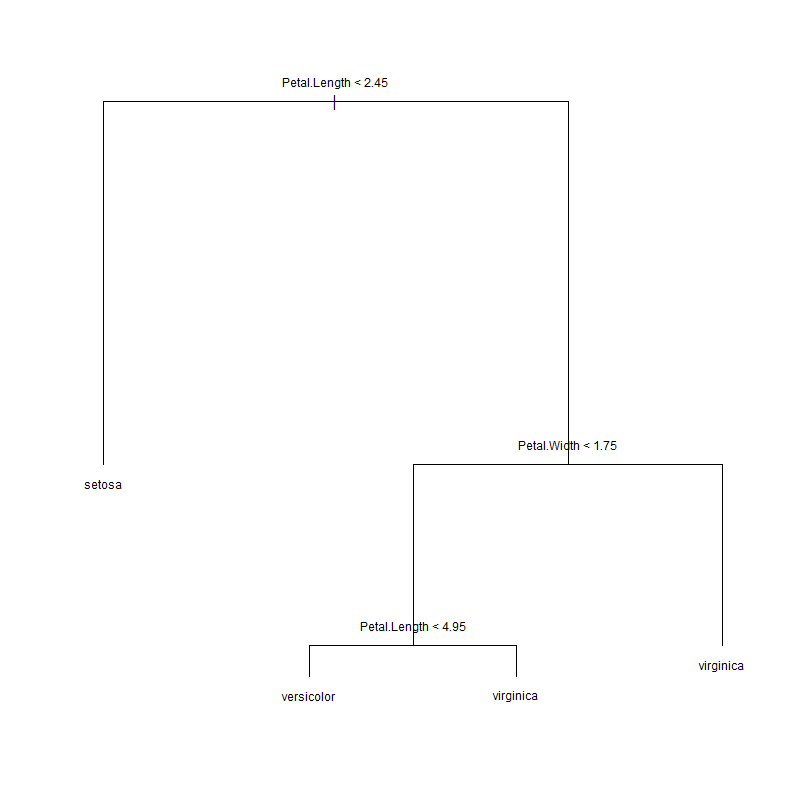
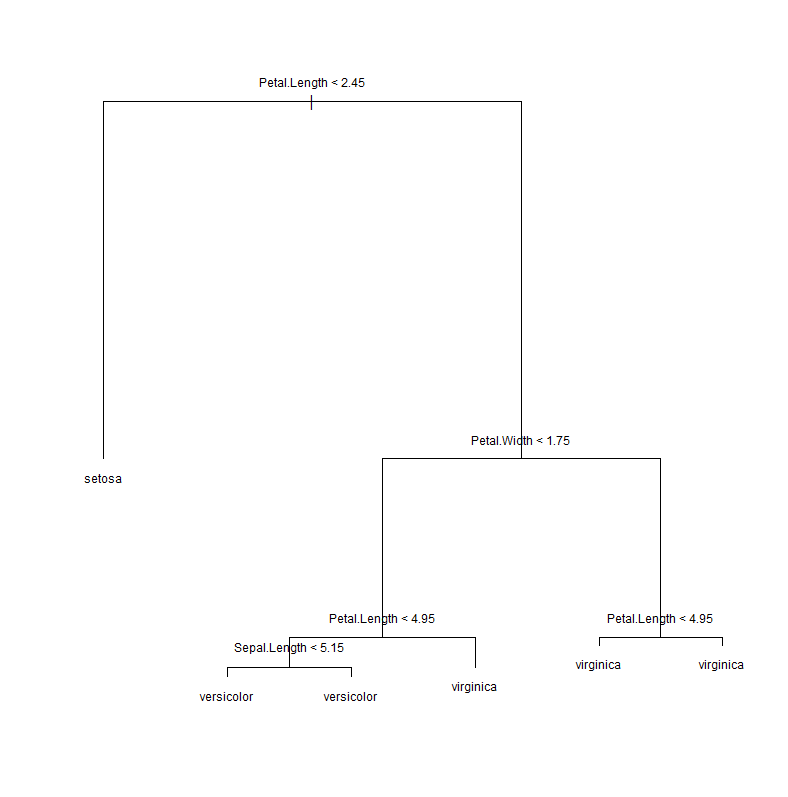


The percentage split decreased the accuracy of all models meaning it estimates the models will perform worse on future data relative to the 10-fold cross-validation testing method. The highest accuracy for the iris data set that we found comes from discretizing the data and running the J48 and RandomForest models with a 10-fold cross validation.

Next we tried running decision tree and random forest models in R. As a first step, we used the {rpart} package, which yielded the model below. Pruning expectedly did not improve the model (given that it is already quite simple). Ten-fold cross-validation yielded accuracy of 93.33%.



As a second step, we used the {tree} package, which yielded the model below at left. Pruning via cross-validation to minimize deviance (using cv.tree function) yielded the model at right. Ten-fold cross-validation running the initial model and allowing pruning to apply separately within fold yielded accuracy of 95.33%.



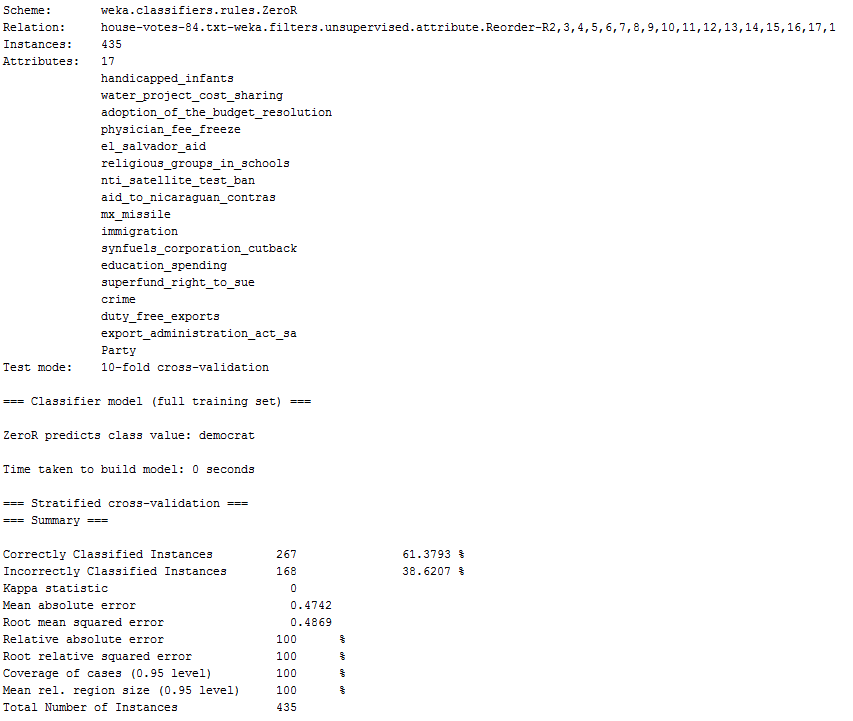
As a final step, we ran random forest models ({randomForest}) with 1000 trees using all possible subsets of the iris predictor variables. The best model specification featured Petal.Length and Petal.Width, and it 10-fold cross-validated with accuracy of 96.00%. This was the most accurate of the three R packages we tried.

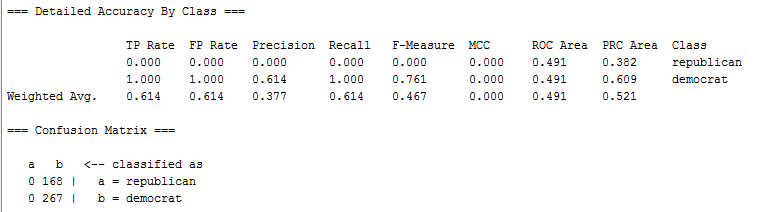
**House Votes Data**

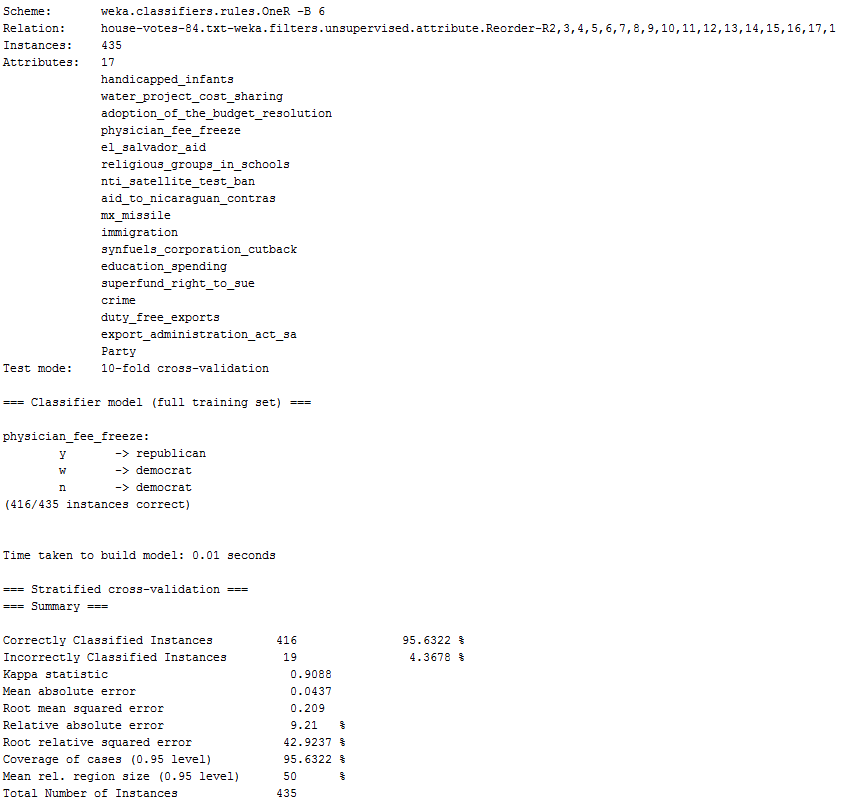
From the Irvine Machine Leaning Repository for this data set, the attribute information is as follows:

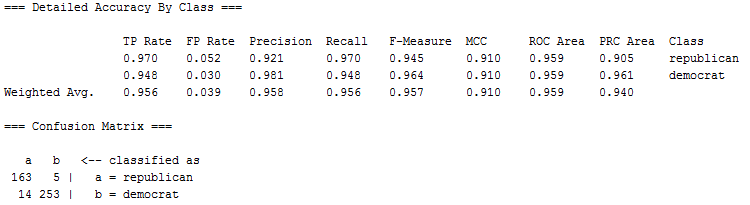
1. Class Name: 2 (democrat, republican)   
2. handicapped-infants: 2 (y,n)   
3. water-project-cost-sharing: 2 (y,n)   
4. adoption-of-the-budget-resolution: 2 (y,n)   
5. physician-fee-freeze: 2 (y,n)   
6. el-salvador-aid: 2 (y,n)   
7. religious-groups-in-schools: 2 (y,n)   
8. anti-satellite-test-ban: 2 (y,n)   
9. aid-to-nicaraguan-contras: 2 (y,n)   
10. mx-missile: 2 (y,n)   
11. immigration: 2 (y,n)   
12. synfuels-corporation-cutback: 2 (y,n)   
13. education-spending: 2 (y,n)   
14. superfund-right-to-sue: 2 (y,n)   
15. crime: 2 (y,n)   
16. duty-free-exports: 2 (y,n)   
17. export-administration-act-south-africa: 2 (y,n)

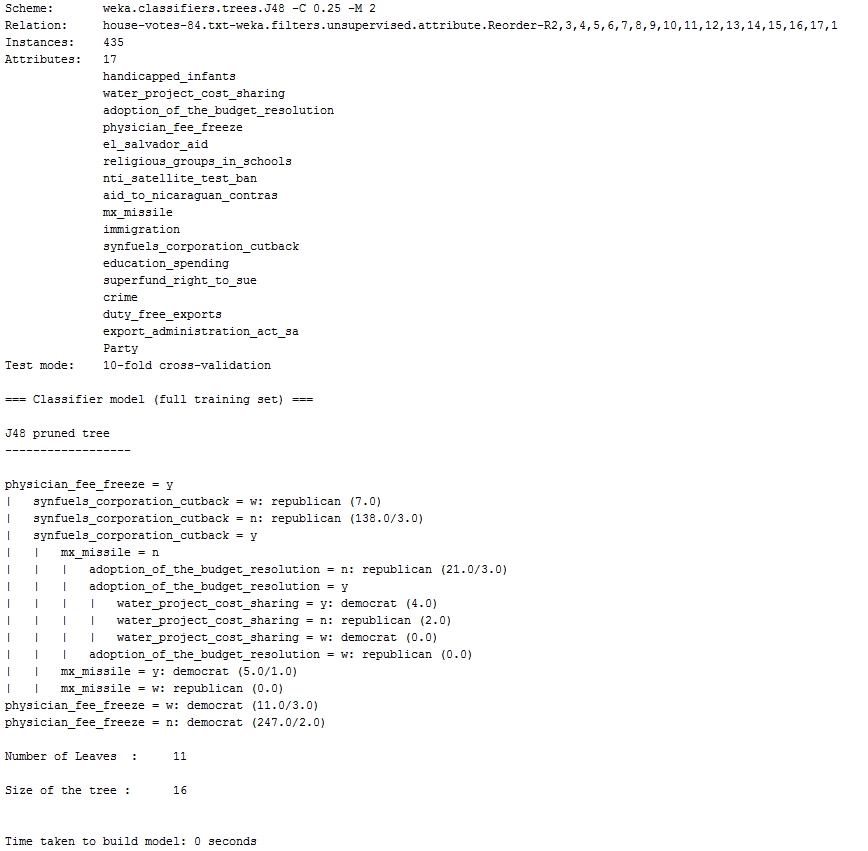
The data file provided for the assignment also used “w” within the voting category to indicate that the politician didn’t say yes or no. We are assuming it stands for withdrew from voting. After loading the house-votes-84.arff file, we first had to edit the data to make Party the classifying attribute. Like before, we ran a few baseline classification models using the 10-fold cross validation option without doing anything else to the data. We decided to use the same models as before. Here are the results:

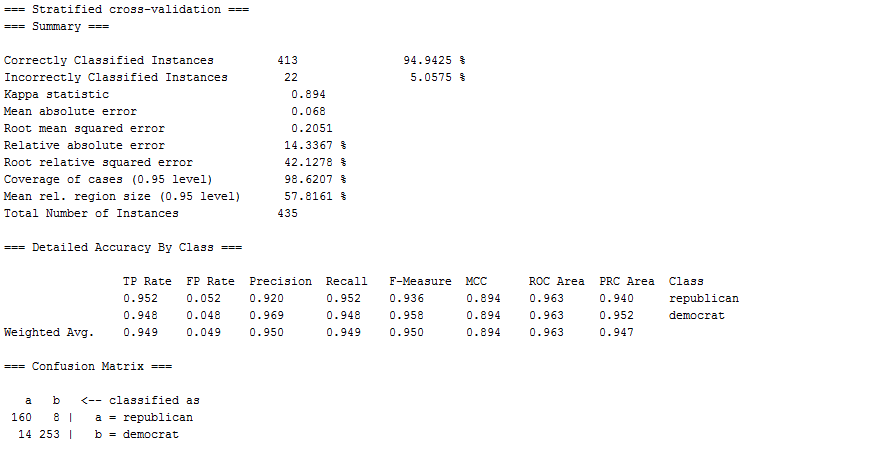


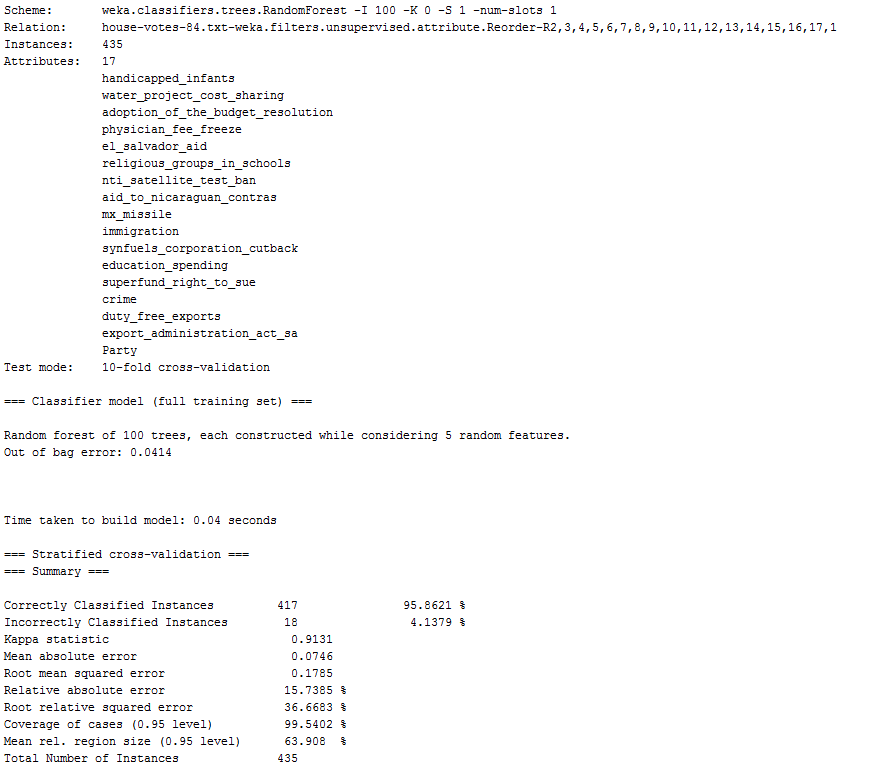


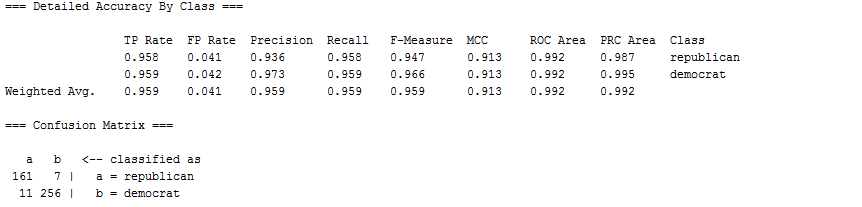




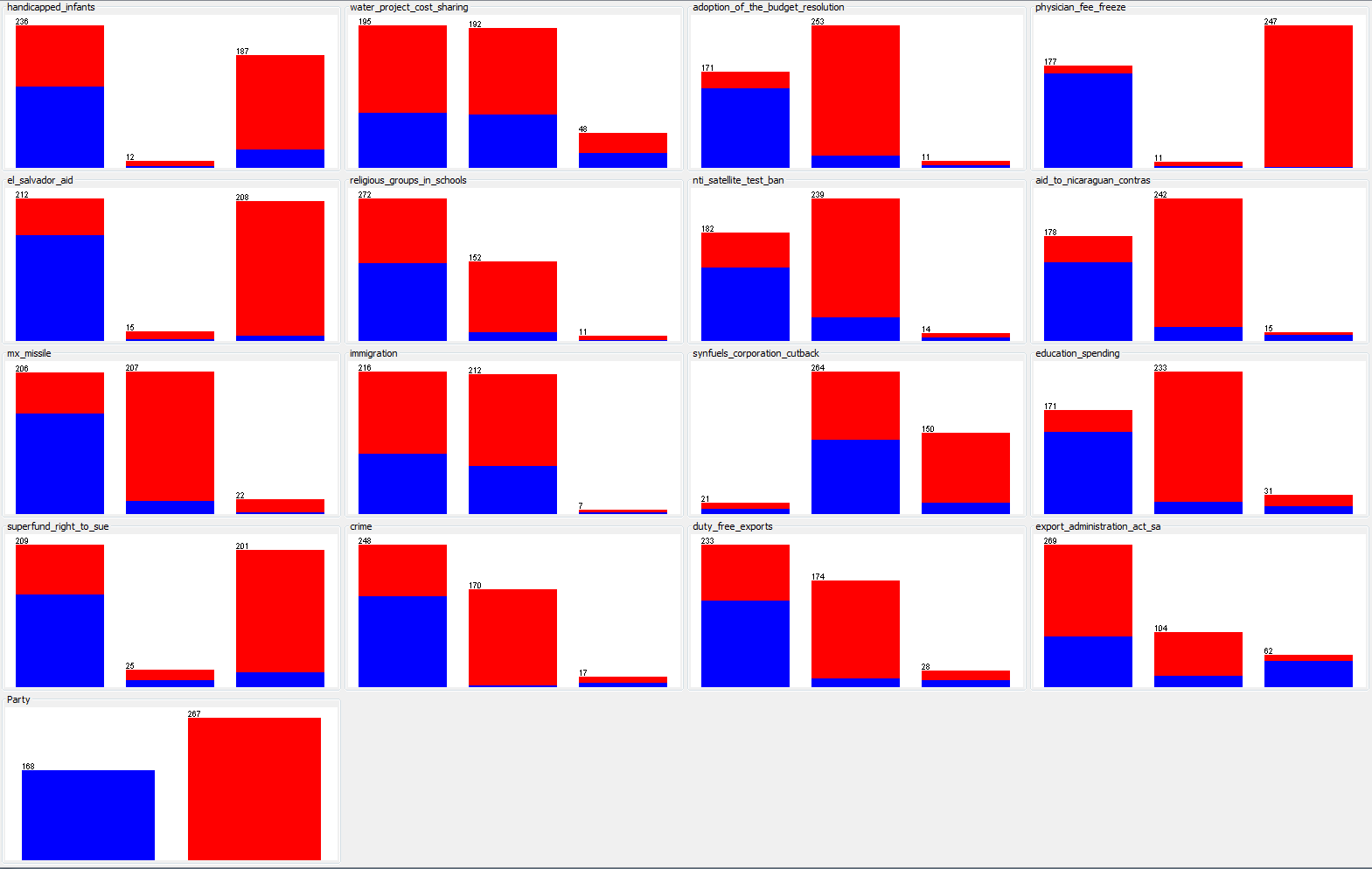








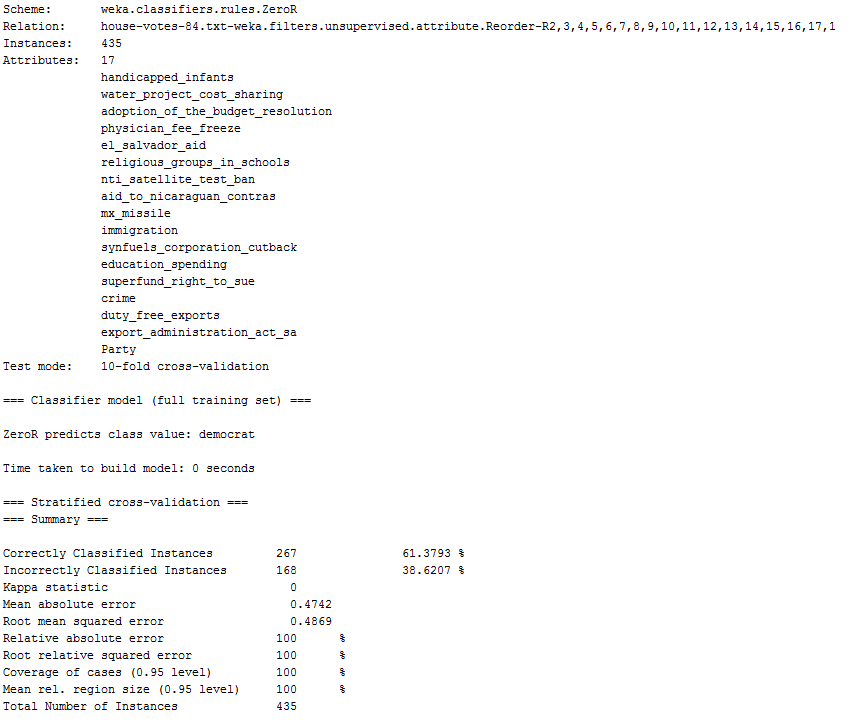
As with the Iris data, the zeroR algorithm was much worse than the others. The oneR performed a little better than the j48 tree algorithm, and the RandomForest tree algorithm performed the best at 0.959. At this point we decided to look at the data. Here are the histograms with the w variable included:

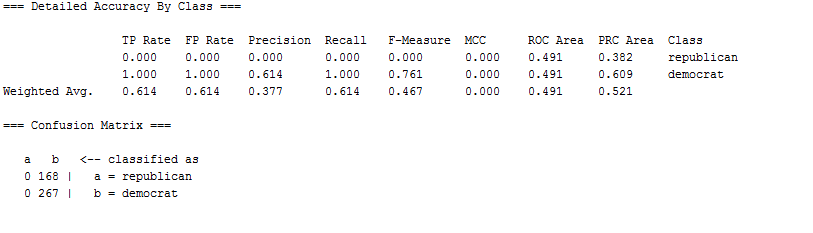


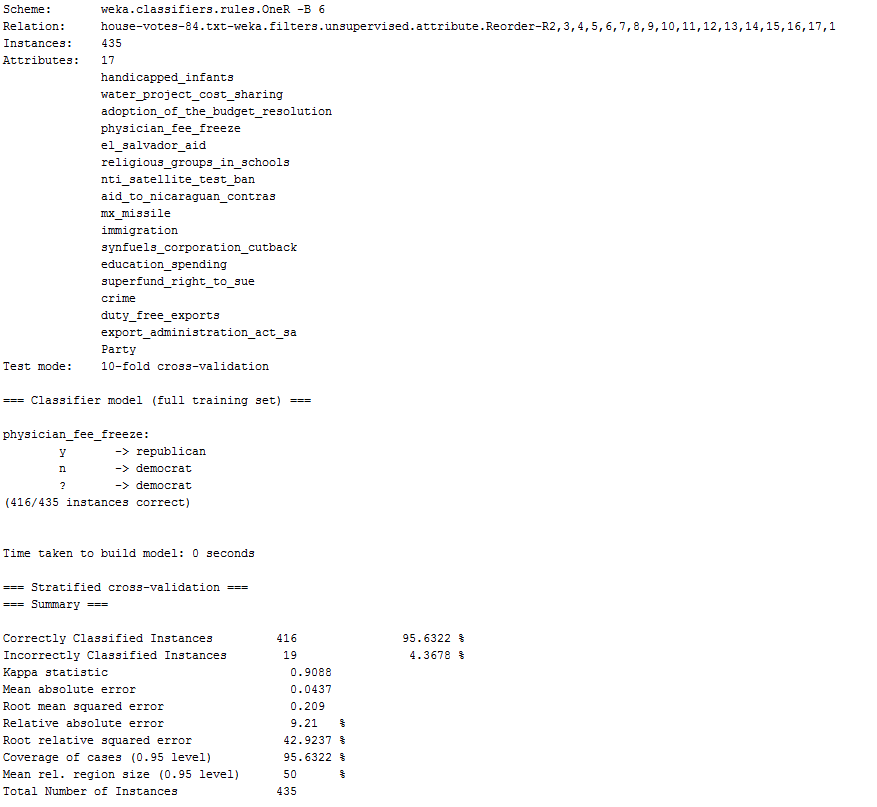
The w variable doesn’t help predict whether a politician will say yay or nay to any particular piece of legislation, so we decided to treat it as a missing value as opposed to a variable. We edited the file in a text editor by replacing any w in the data with a question mark, which Weka treats as a missing value. Here are the histograms for the cleaned up data:

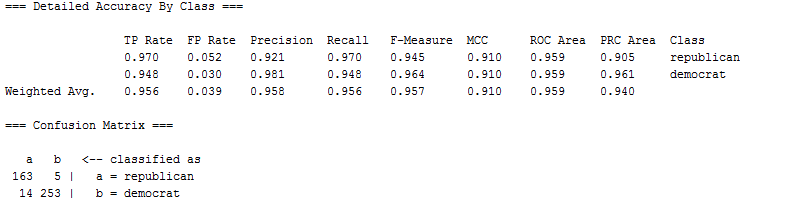


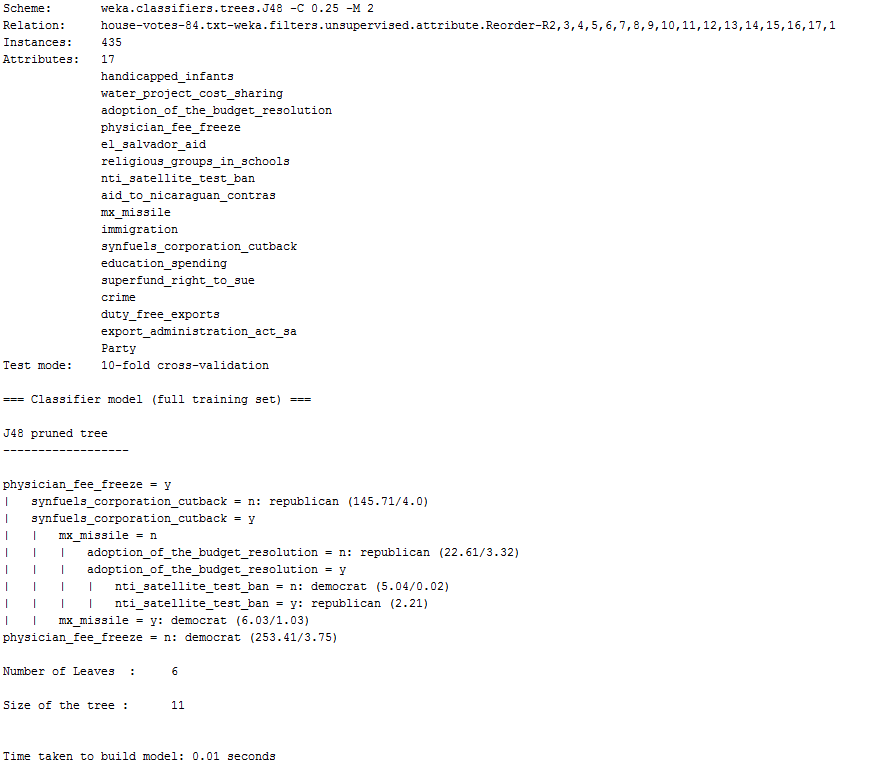
We then re-ran the previous algorithms on this data, and the results are as follows:

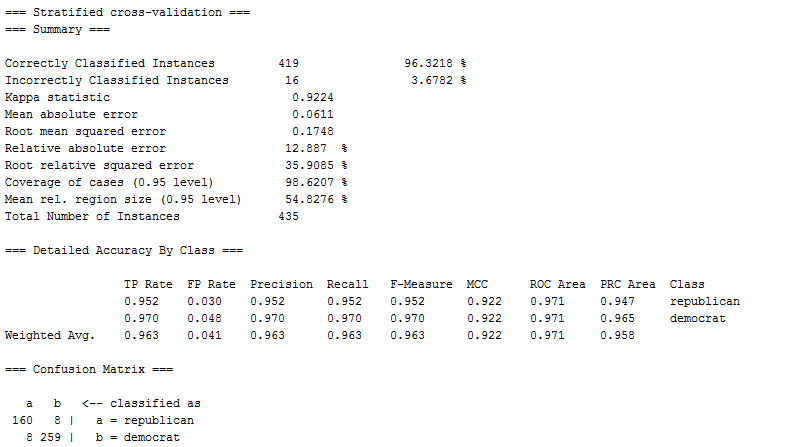


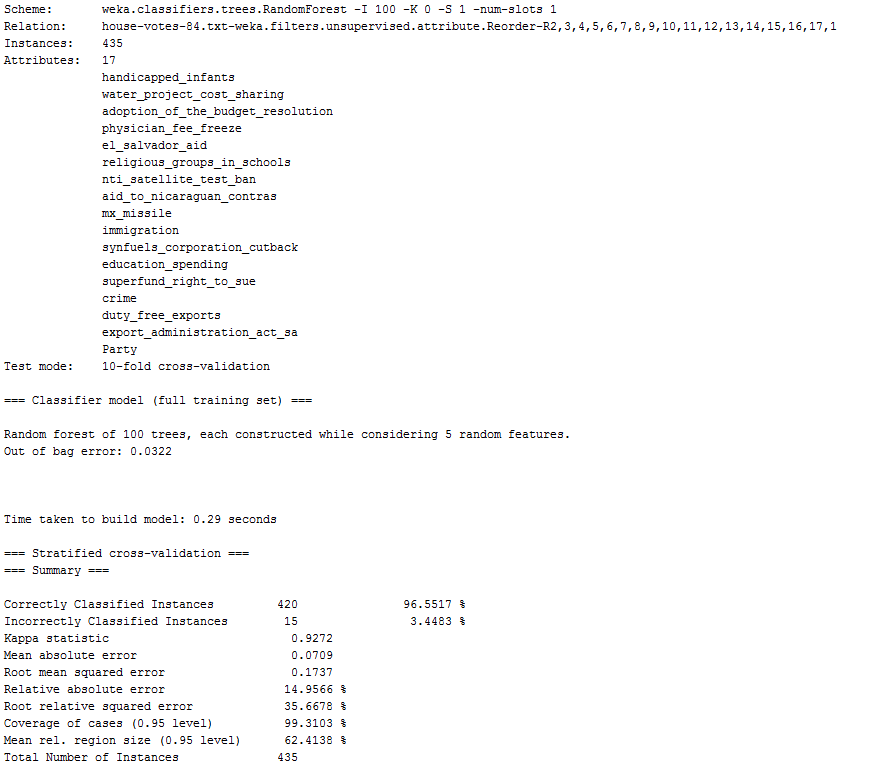


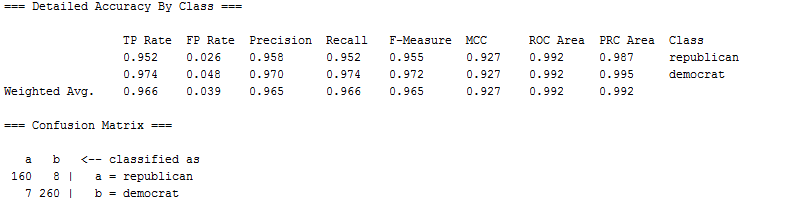




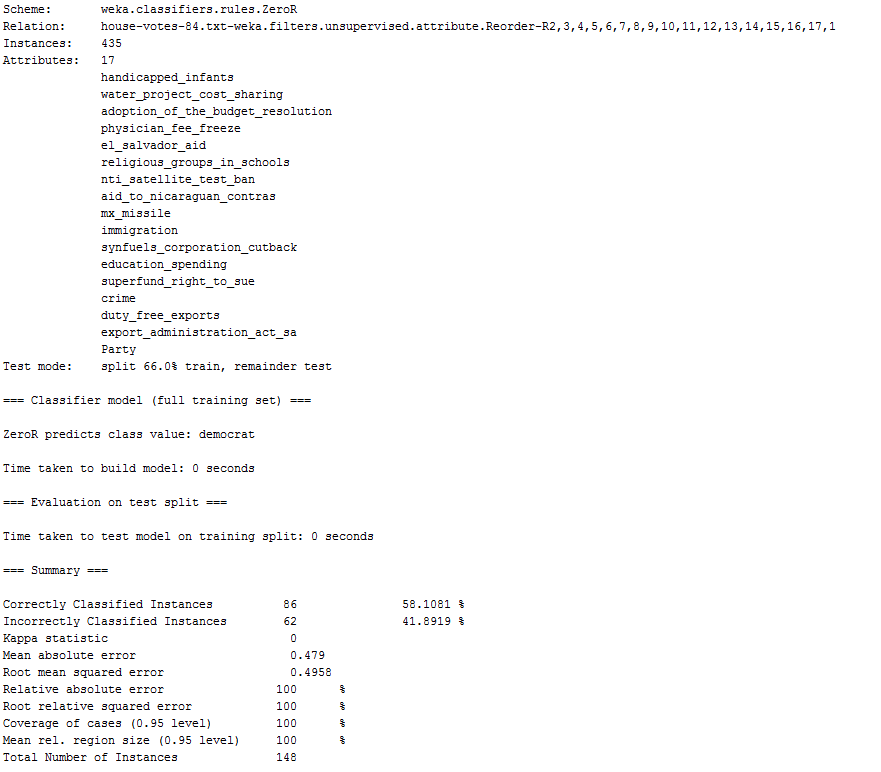


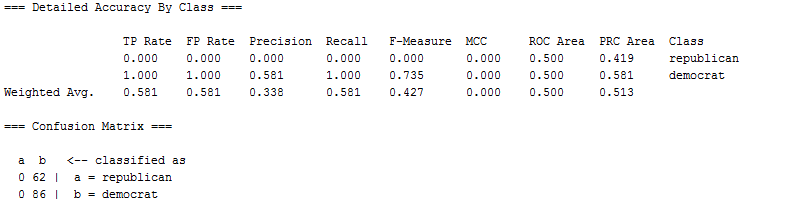


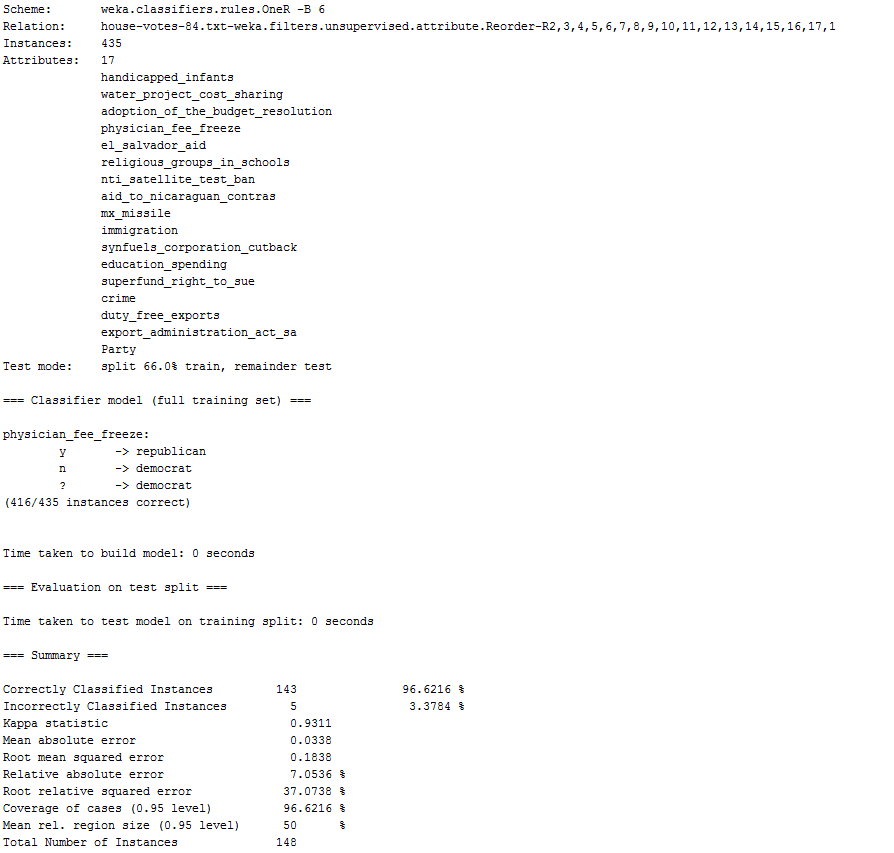


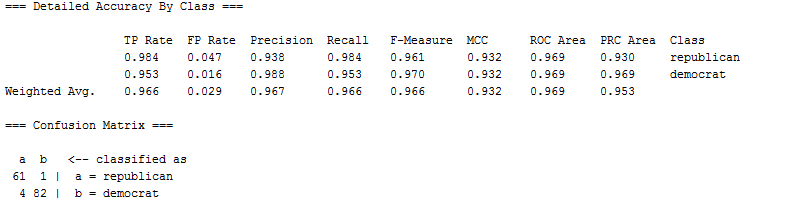


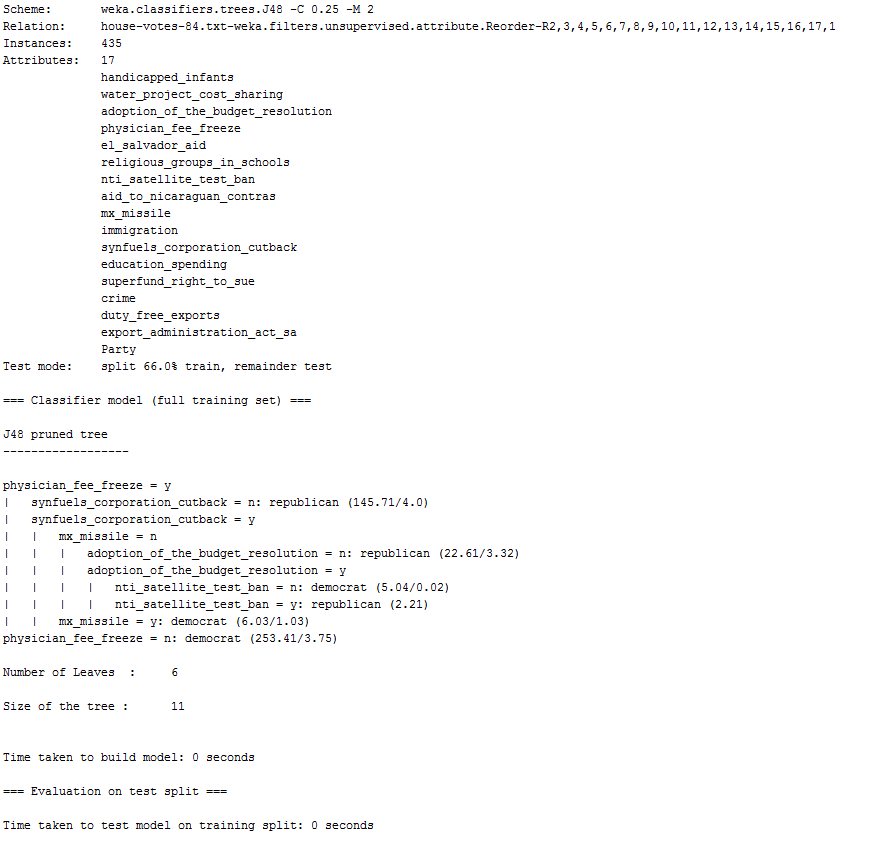
From the results, the accuracy on the rules based algorithms didn’t change, but the accuracy increased for the tree algorithms with the RandomForest tree having the highest accuracy with an F-measure value of 0.965. For symmetry with the analysis of the iris dataset, we re-ran the models by changing the test option to percentage split at 66% to see how that hold-out option would estimate the models. Here are the results:

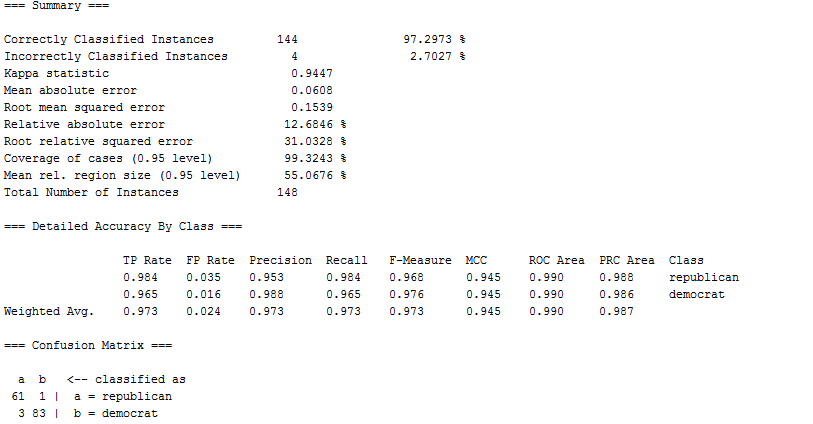


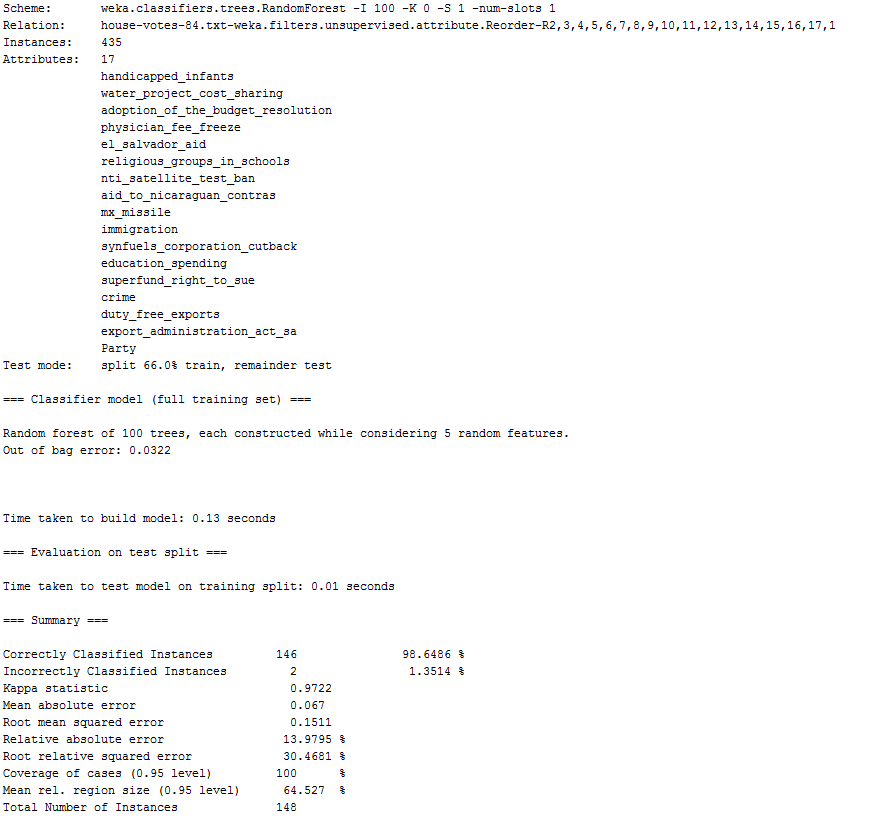


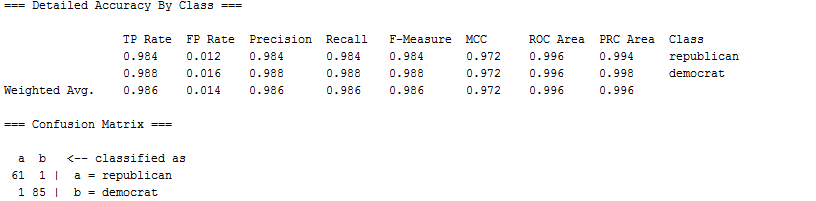








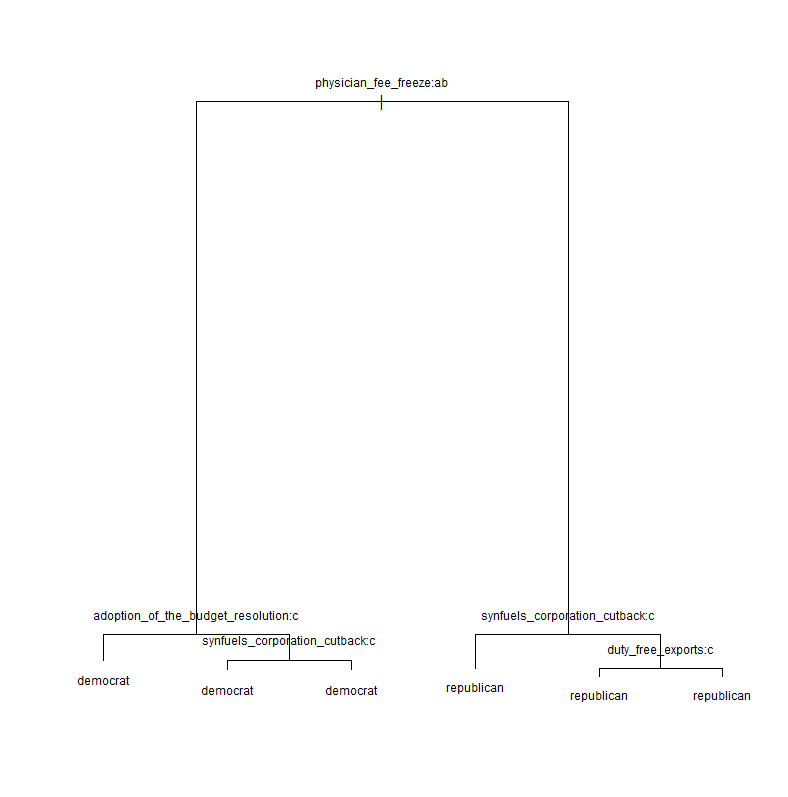
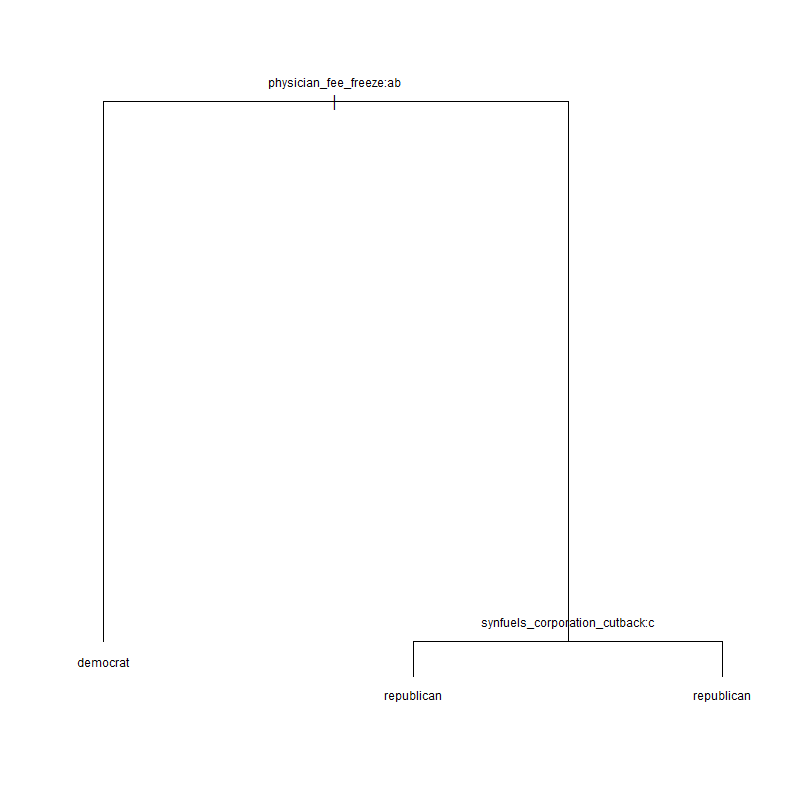




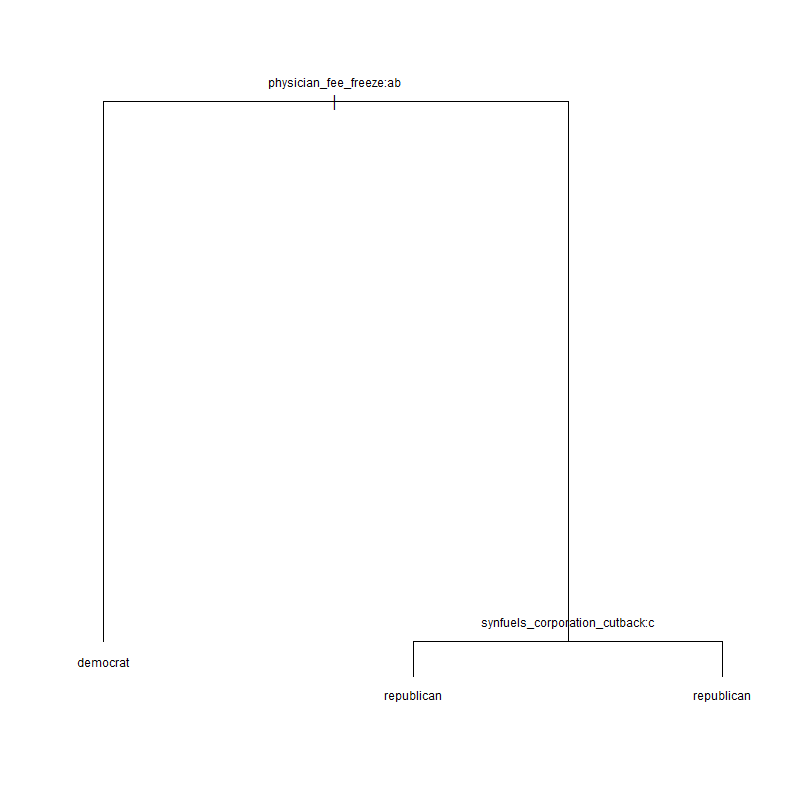
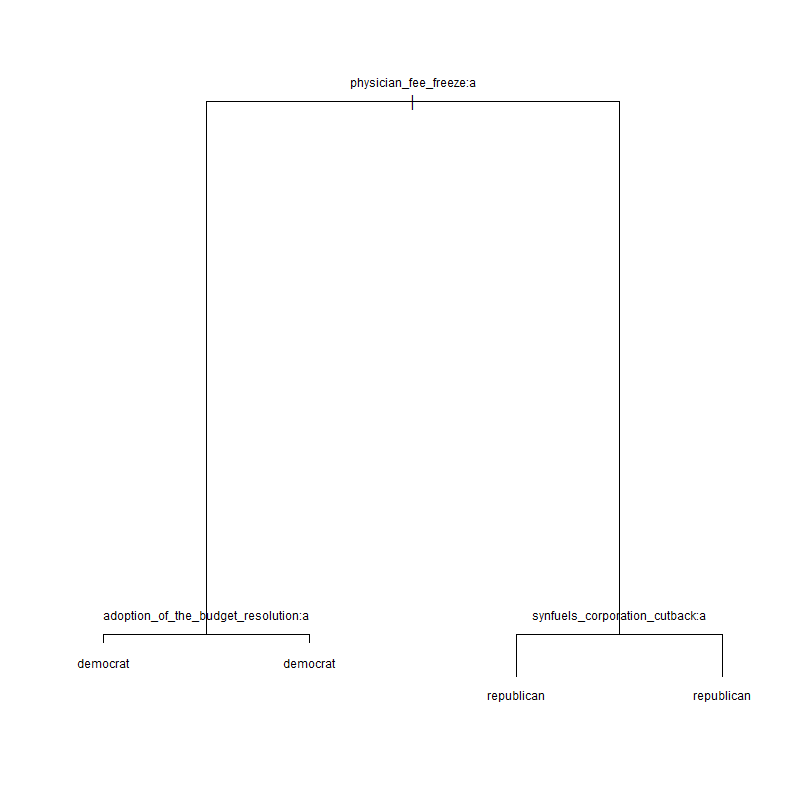
The results show that the percentage split testing option estimates that the zeroR model would be less accurate for predicting future data compared to the 10-fold cross-validation. In the other 3 models, the percentage split estimates the models will perform better. The RandomForest tree had the highest accuracy with an F-measure value of 0.965 when using the 10-fold cross-validation option, but the highest accuracy for the house votes data set that we found using Weka comes from cleaning the data to only include y and n votes and running the RandomForest model with a percentage split testing option at 66% giving an F-measure value of 0.986.

Next we again tried running decision tree and random forest models in R. As a first step, we used the {rpart} package, which yielded a very simple model, featuring only physician\_fee\_freeze such that a vote of yes predicts Party=Republican, and a vote of no or withdrew predicts Party=democrat (obviously this model did not require pruning). Since this model is really more of a rule than a model, accuracy could determined simply by observing the accuracy of the rule in the full dataset, which was 95.63%. We next reran the {rpart} model counting withdrawn votes as missing, which resulted in an identical model, though resulting observed accuracy did increase very slightly to 96.23%.

We then used the {tree} package, which yielded the model below at left. Pruning via cross-validation to minimize deviance (using cv.tree function) yielded the model at right, however the unpruned model cross-validated better than the pruned model, resulting in mean accuracy of 95.42%.

Rerunning this model type counting withdrawn votes as missing yielded a simpler tree (shown below at left), which pruned the same way as the previously pruned model. Again, the unpruned model cross-validated with the best accuracy at 95.65%; so as was the case in the previous model, removing the withdrawn votes from the data slightly improved accuracy.



Last of all, we ran random forest models (number of trees = 1000). Doing so on the full set of features cross-validated with accuracy of 96.57%. We then used a hill-climbing algorithm to perform wrapper feature subset selection in order to find the subset of features that predicted with the strongest 10-fold cross-validated accuracy, again using a 1000 tree random forest. The best subset featured the set of 10 predictors below and resulted with mean accuracy of 97.48%.

[1]"water\_project\_cost\_sharing" "adoption\_of\_the\_budget\_resolution" "physician\_fee\_freeze"

[4]"el\_salvador\_aid" "nti\_satellite\_test\_ban" "aid\_to\_nicaraguan\_contras"

[7]"mx\_missile" "immigration" "synfuels\_corporation\_cutback"

[10] "duty\_free\_exports"

[1] 0.9748414

Running the random forest model procedures above counting withdrawn votes as missing reduced accuracy (best mean accuracy was 96.97%).

Thus, the best of three r packages we used was again the random forest, this time using a hill-climbing algorithm to select the optimal predictor subset to maximize classifier accuracy.