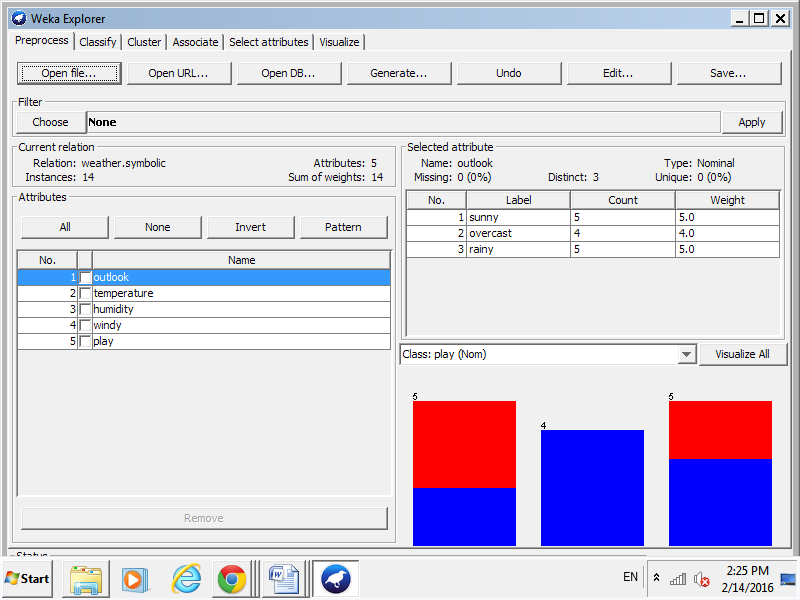
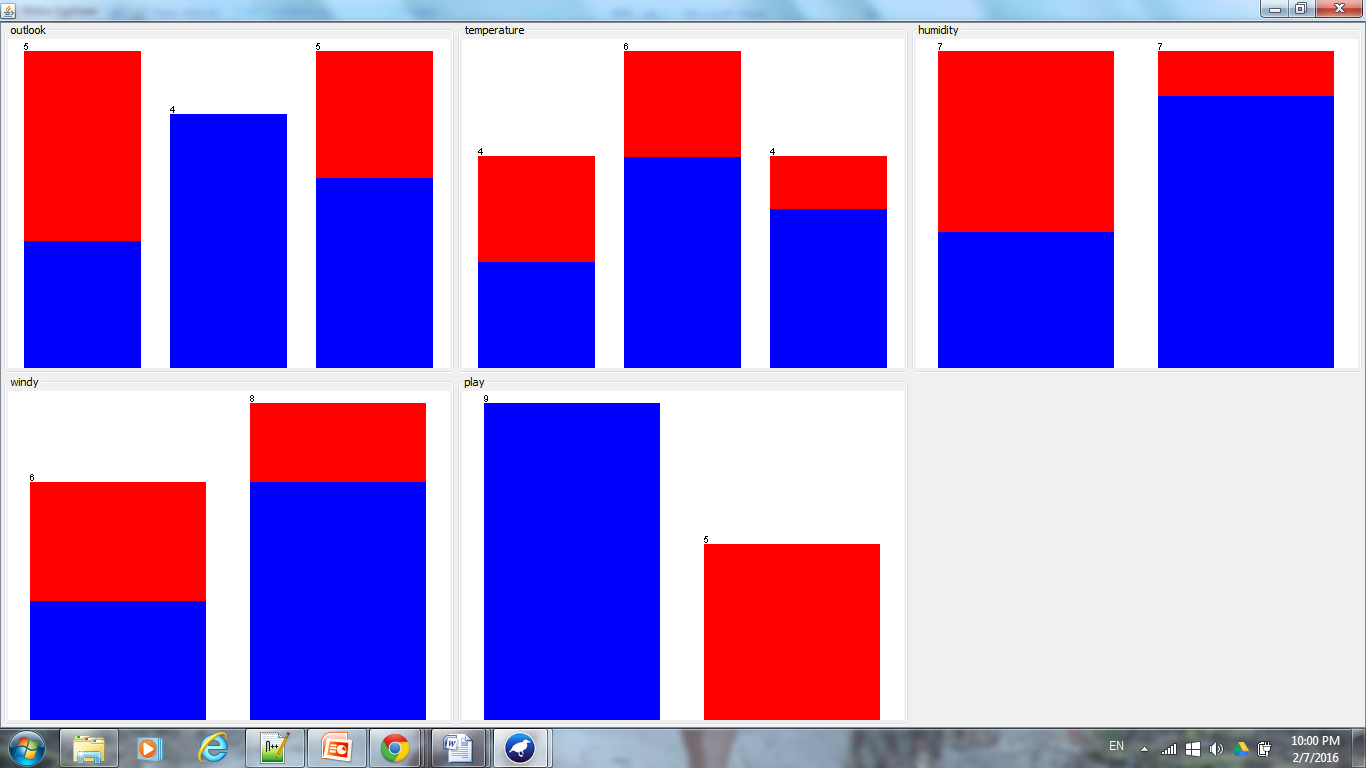
### **Data Mining I: Basic Methods and Techniques**

***Laboratory Assignment #3:***

1. Use the Classification rule production method PRISM (weka.classifiers.Prism) on the Weather.nominal data set. How many rules did it produce? Compare this to the Decision tree produced on the same data. What is the difference between the two models?

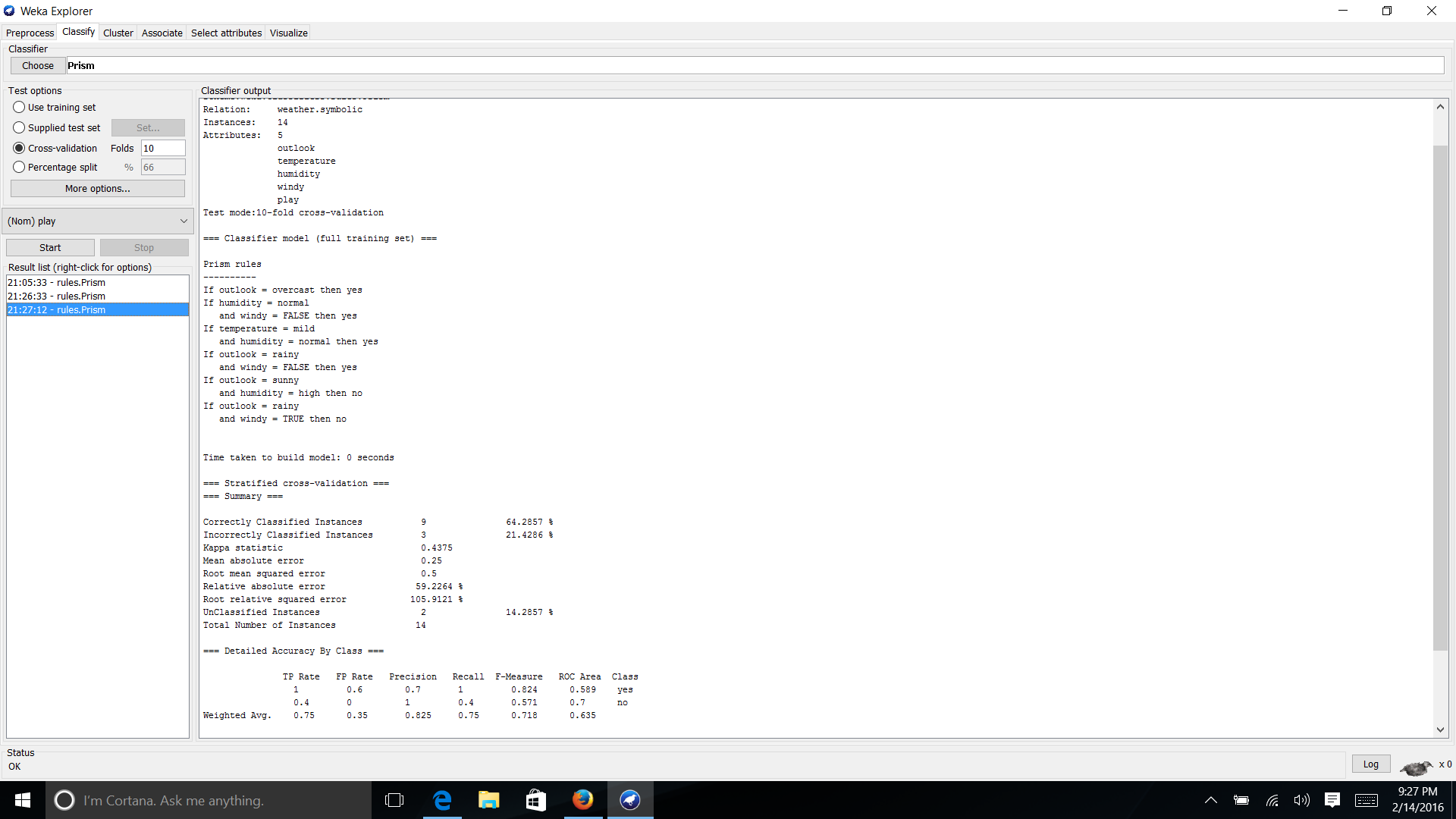
A. The file Weather.nominal.arff was opened in Weka. The data set contains 5 attributes, with supervised learning to be completed in order to determine the outcome of the class attribute “play”, which provides a nominal answer (yes/no) as to whether or not to play base do weather information. No further filters were necessary to apply, since all of the values are nominal. 

B. Selecting “Visualize All” in order to see the distribution of all the attributes for 14 instances.

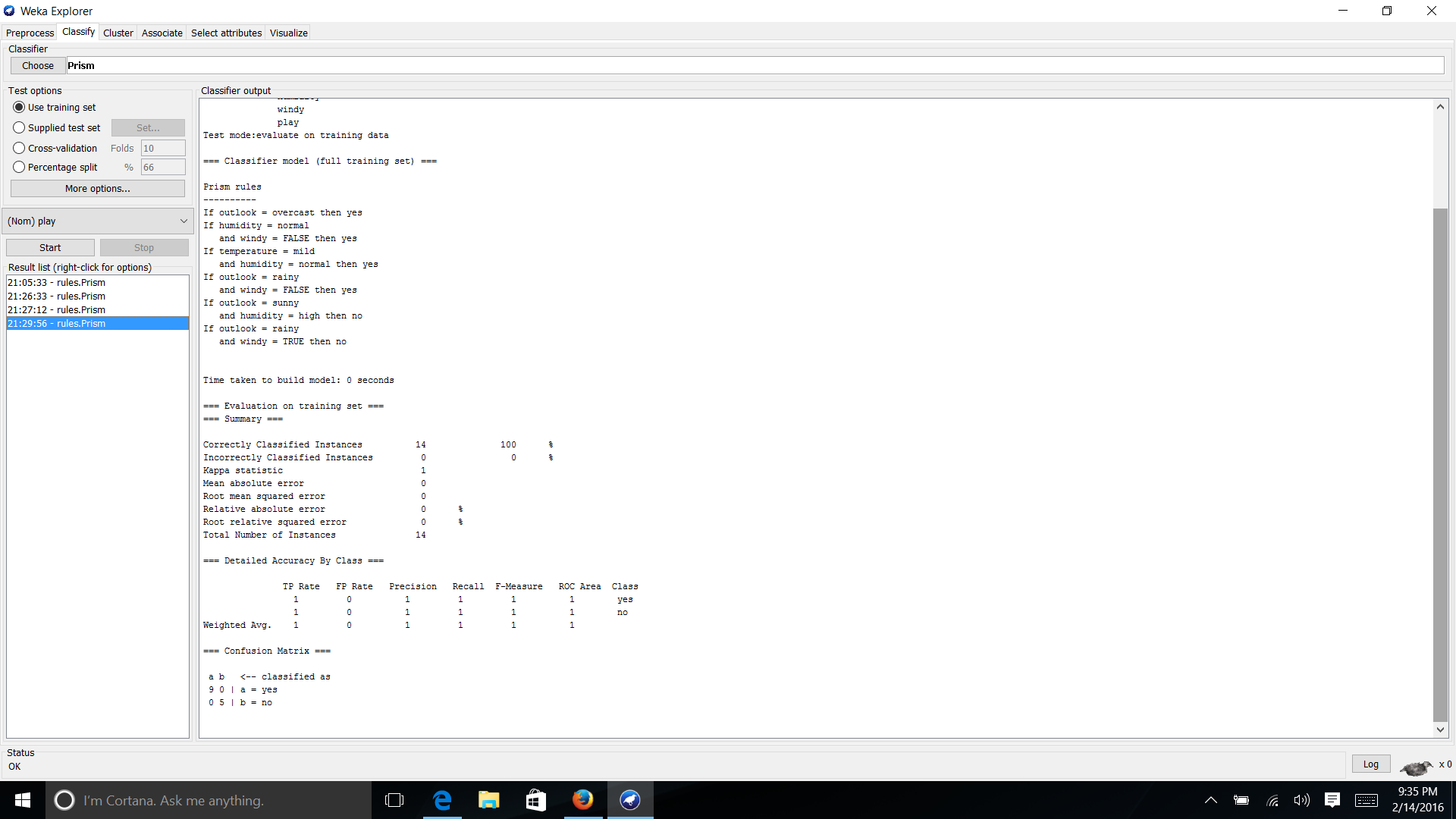


C. Note: Weka 3-6-13 was installed vs. Weka 3-9; for some reason the most recent version of Weka does not contain the PRISM classifier. Under “Classify”, classifiers/rules/PRISM was chosen to create a PRISM rule on the data set, where “perfect” rules are attained with an expectation that all instances will be classified.

Under “Test options”, “Cross-validation” at “Fold 10” was selected in order to accomplish the most believable evaluations where the data is split into training and test data sets 10 times with models being created, iterated, and polished between each training and testing. The model tree was generated.



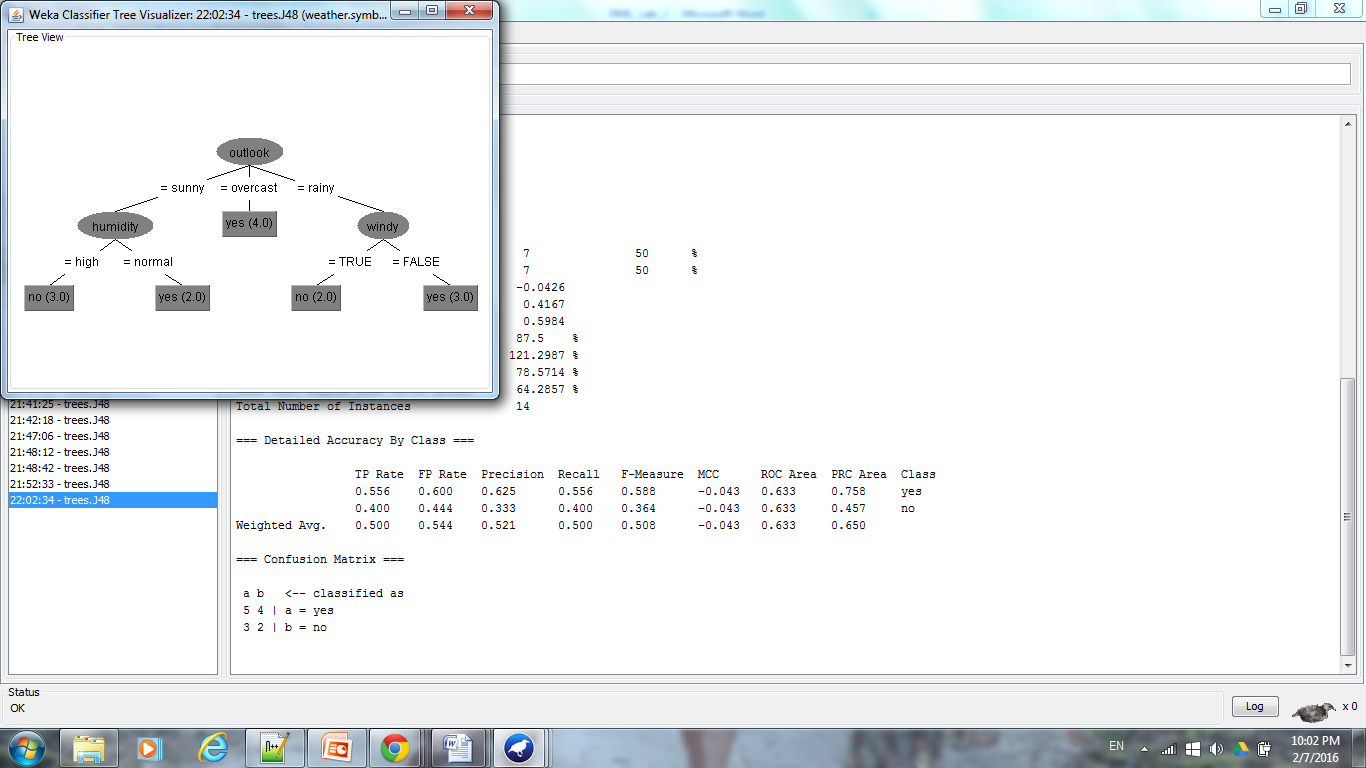
Yet, these 6 rules generated did not classify all the instances. Therefore under “Test options”, “use training set” was selected to see if certain rules are excluded from above when using the entire 14 instances to build the model.



All 14 instances were classified, yet analysis of the 2 sets of PRISM rules above are identical. Therefore, cross validation is preferred in modeling and although not all instances were classified, the PRISM rules attained should be used with a mean absolute error of 0.25.

D. To compare with the Decision tree:

In selecting the “Classify” tab, the J48 decision tree was generated using “Cross validation” at “Fold 10” in order to create the best performing model. It should be noted that based on the nominal data set, the numeric dataset can be further discretized for the humidity and temperature values to reach a similar model as shown below.



From the generated model and summary information, it can be seen that 7 instances were classified correctly, 7 were classified incorrectly, and the mean absolute error of 0.4167. The PRISM rules have a lower mean absolute error.

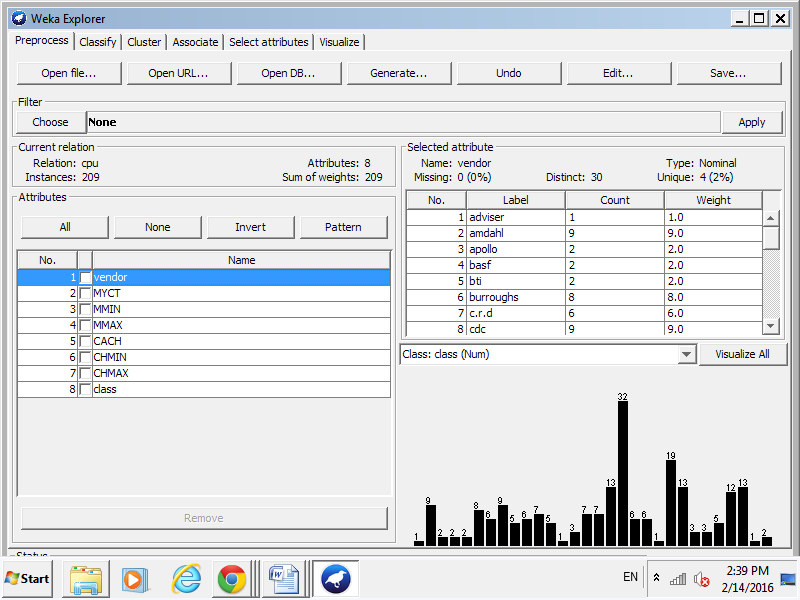
Comparing the Prism rules with the Decision Tree - Each PRISM rule was read & followed down the tree to determine where the decision tree did not agree with the PRISM rules, thus determining the course of error in the Decision Tree:

1. If outlook = overcast then yes DT agrees
2. If humidity = normal and windy = FALSE then yes DT agrees
3. If temperature = mild and humidity = normal then yes DT does not consider temperature
4. If outlook = rainy and windy = FALSE then yes DT agrees
5. If outlook = sunny and humidity = high then no DT agrees
6. If outlook = rainy and windy = TRUE then no DT agrees

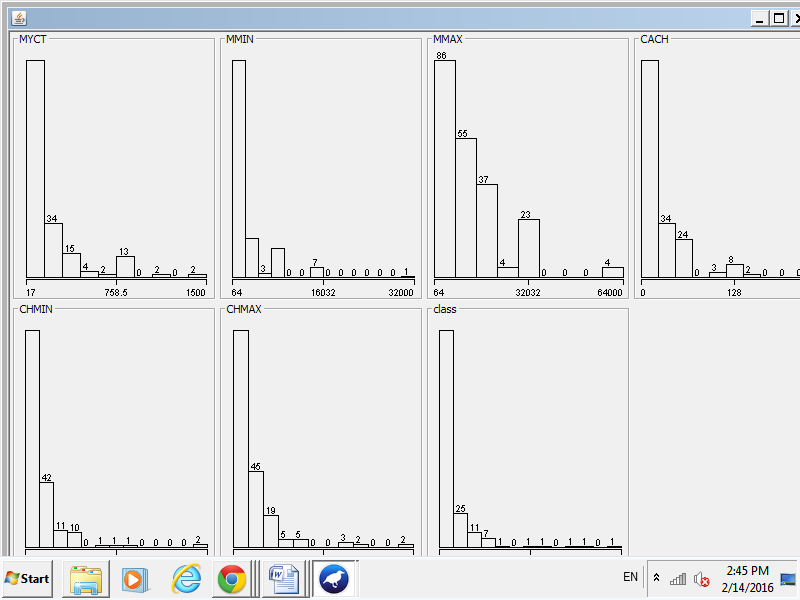
If the attributes are mentioned in the PRISM rules, but not in the decision tree then the decision tree cannot agree and thus this shows that not all of the instances can be correctly classified using the decision tree.

1. Use the Regression tree learning scheme (weka.classifiers.M5') to analyze the CPU.arff. Evaluate the difference between a model tree and a regression tree. Experiment with the available parameters to understand their significance and discuss how they influence the model?

A. . The file cpu.arff was opened in Weka. The data set contains 8 attributes, with 7 containing numeric attribute values and 1, vendor, containing nominal values.

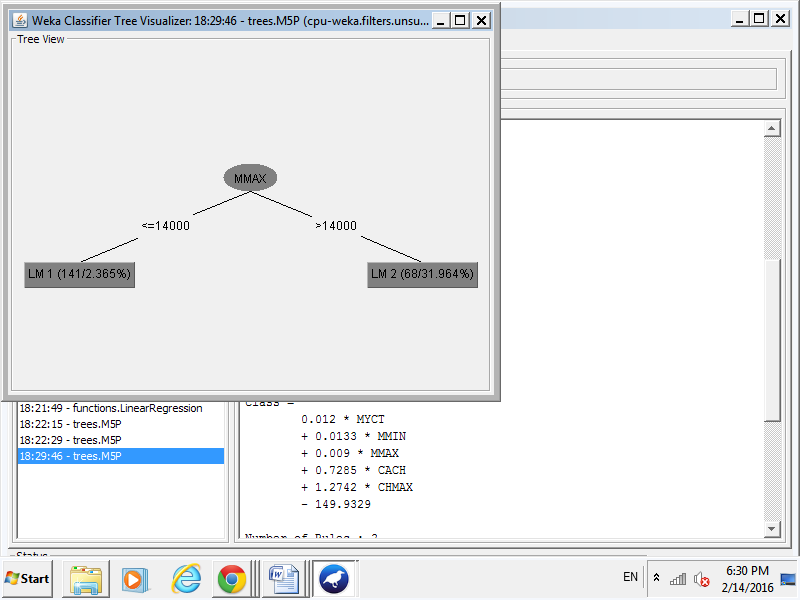


B. In order to properly run a regression tree, the nominal attribute, vendor, was removed from the dataset and when clicking “Visualize All” the distribution for the 7 remaining trees was analyzed.

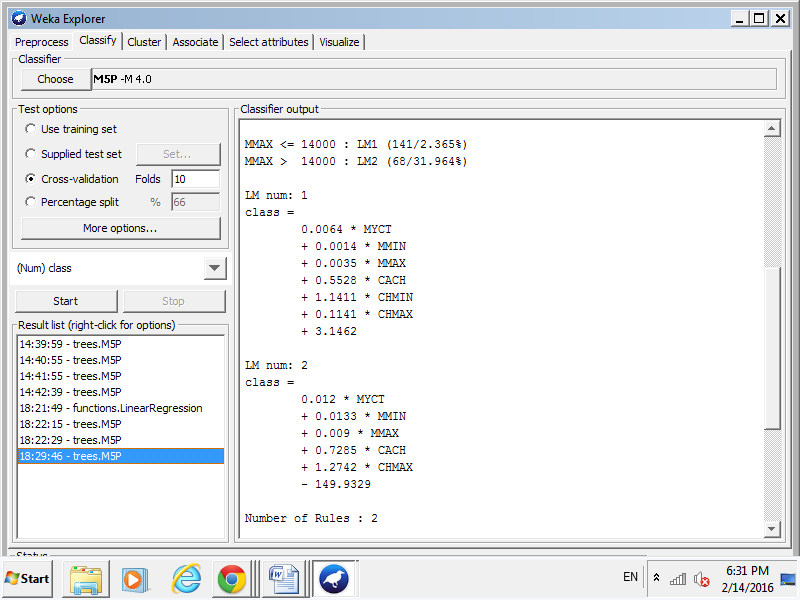


C. Note: A linear model tree is a decision tree with a linear functional model in each leaf, whereas in classical regression tree it is the sample mean of the response variable for statistical units in each leaf (hence, a constant) that is being considered. Linear model trees can be seen as a form of locally weighted regression, while regression tree are piecewise-constant regression.

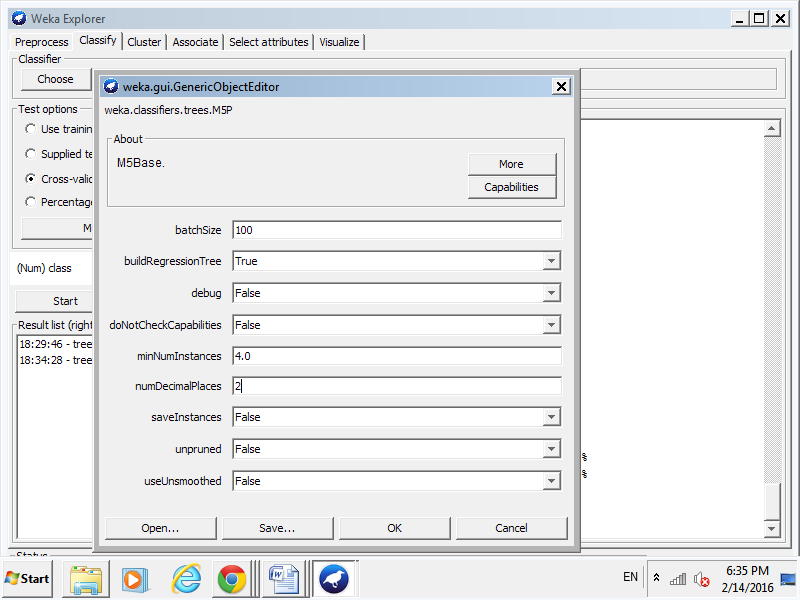
In order to create a model tree, under the “Classify” tab, classifier/trees/MP5 was selected. Under “Test options”, “Cross-validation” at “Fold 10” was selected in order to accomplish the most believable evaluations where the data is split into training and test data sets 10 times with models being created, iterated, and polished between each training and testing. The model tree was generated.



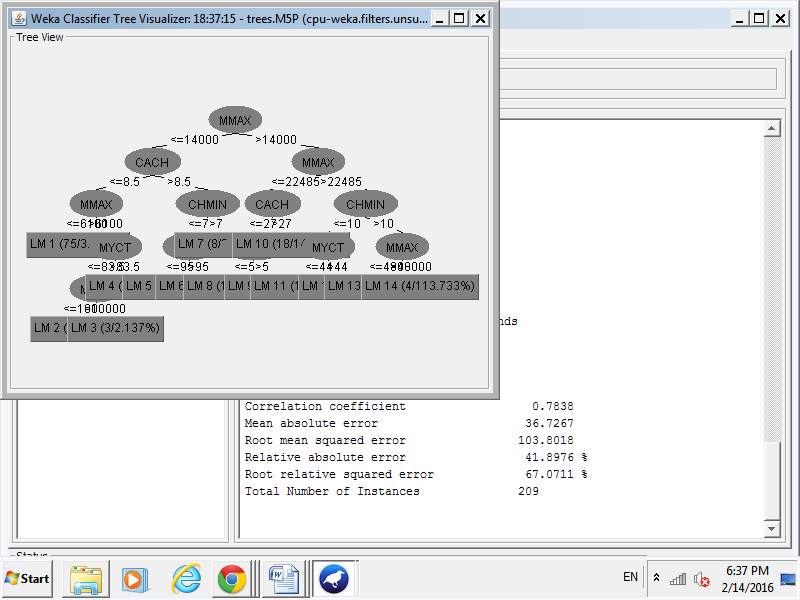
With the following 2 rules for the tree generated:



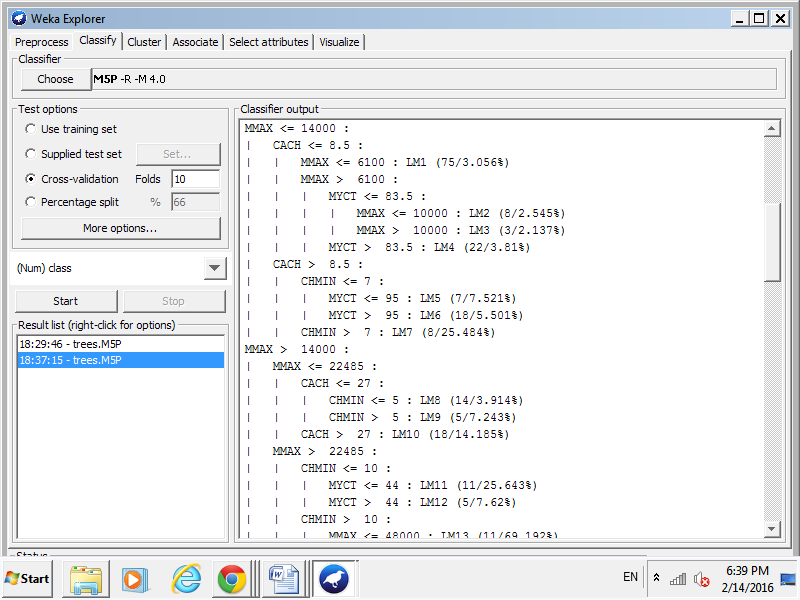
D. In order to create a regression tree, under the “Classify” tab, classifier/trees/MP5 was selected. Under “Test options”, “Cross-validation” at “Fold 10” was selected in order to accomplish the most believable evaluations where the data is split into training and test data sets 10 times with models being created, iterated, and polished between each training and testing. The model tree was generated. Furthermore, changing “buildRegressionTree” to “True” allows Weka to built a regression tree.



Which looks like:



With a total of 14 rules for the tree generated:



Analysis/Comparison of model tree versus regression tree:

In the model tree only a subset of attributes are used in the model, in this case MMAX was used to predict the class, LM1 or LM2. After reviewing the variable information for cpu.arff on Notepad, Weka used the maximum main memory in kilobytes to predict the 2 numeric classes to classify the data set into. The attribute, MMAX had the lowest standard deviation therefore was used in the model tree for classification.

In the regression tree, more rules were generated to determine which numeric class, LM1-LM14, the instance belonged to. This model looks to be a lot more complex in comparison to the model tree. In a model tree, the class indicator is binary while in a regression tree the class indicator is not. Recursively, the error is determined at each node of the regression tree and by the tree which produced the lowest root mean squared error is used. Note: the model tree has a root mean squared error of 35.1885 and the regression tree has a root mean squared error of 103.8018.

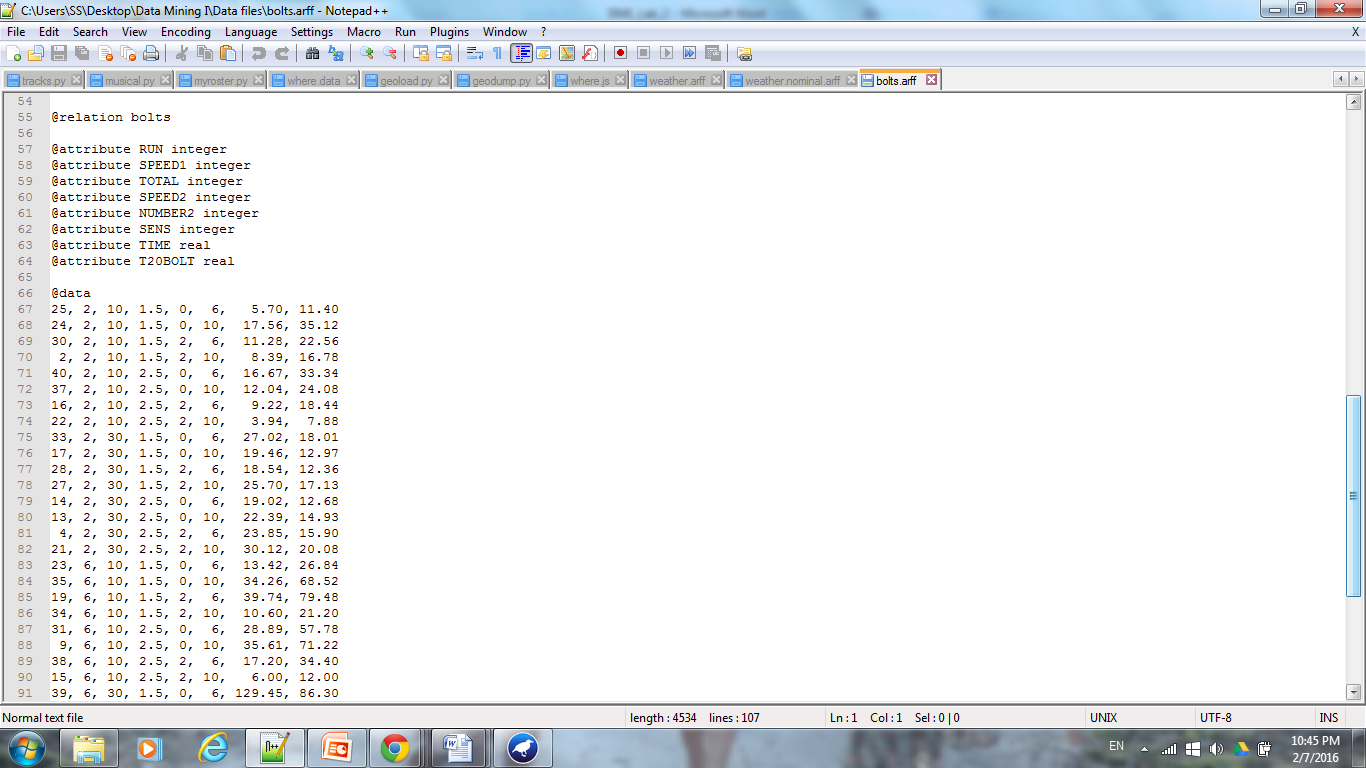
Experimenting with available parameters:

If “unpruned” is chosen to be “True”, both trees become larger with more rules generated, this causes overfitting as Weka tried to accommodate rules for all of the instance. If “useUnsmoothed” is chosen to be “True”, the trees do not really change but errors for the trees do. Furthermore, decreeasing the “minNumInstance” will delay splitting and termination.

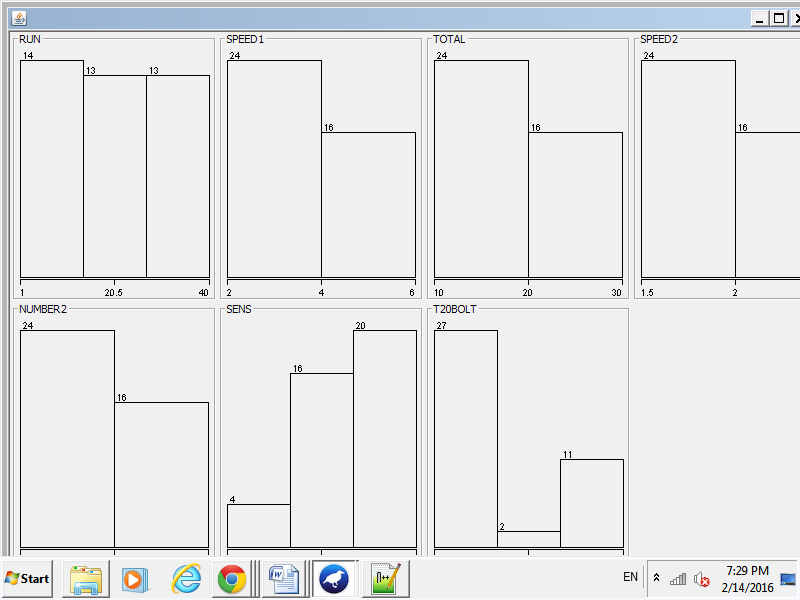
1. Use the Regression tree learning scheme (weka.classifiers.M5') to analyze the bolts data (bolts.arff without the TIME attribute):

* Analyze the data. What adjustments have the greatest effect on the time to count 20 bolts?
* How does this model differ from the Decision Tree induced tree?

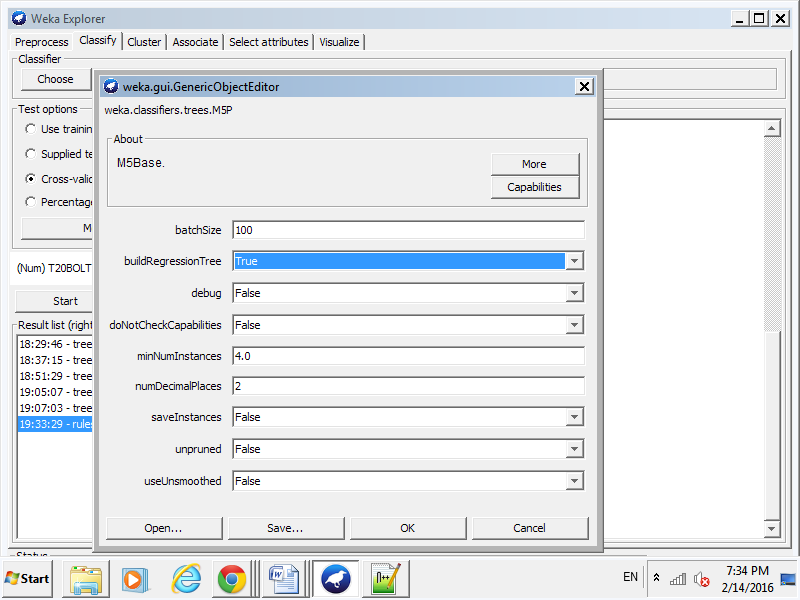
A. Analysis of the bolts.arff data set showed that the attribute values are associated with integer or real values. Therefore, when inputting the data set into Weka for a decision tree, the proper filter parameters must be placed.



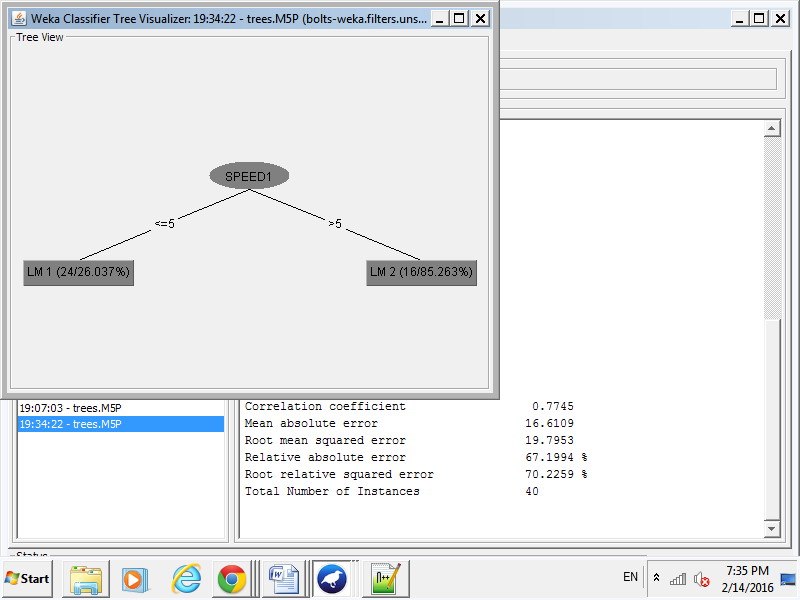
B1. The bolts.arff data was opened in Weka. First during the Pre-process phase, “removing” the Time attribute from the data set would create a data set which did not depend on the value of Time to determine the time needed by a machine to produce and count 20 bolts. Selecting “Visualize All”, showed the numeric distributions for the remaining attributes. Note: T20BOLT will be the class for the data set, the distribution looks like there are 3 class attributes for T20BOLT.



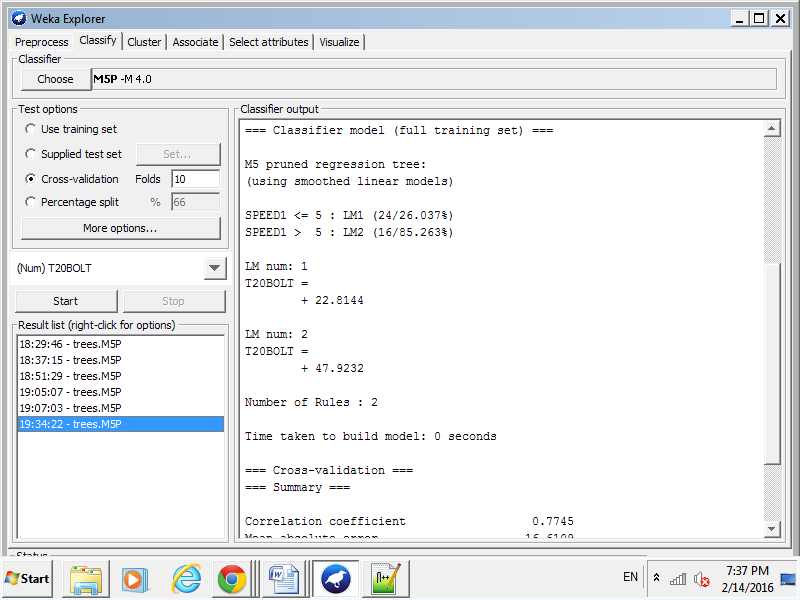
C1. In order to create a regression tree (not model tree), under the “Classify” tab, classifier/trees/MP5 was selected. Under “Test options”, “Cross-validation” at “Fold 10” was selected in order to accomplish the most believable evaluations where the data is split into training and test data sets 10 times with models being created, iterated, and polished between each training and testing. The model tree was generated. Furthermore, changing “buildRegressionTree” to “True” allows Weka to built a regression tree.



And the following regression tree was built:



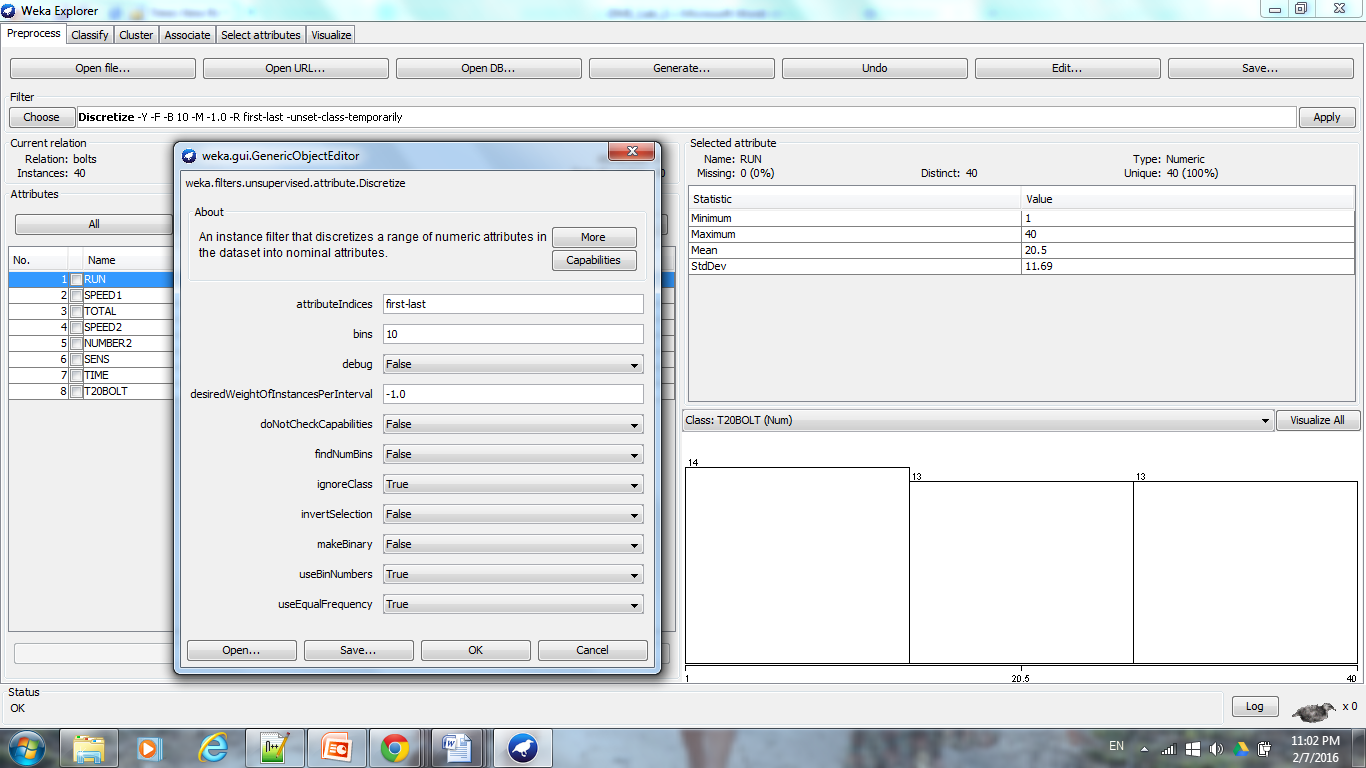
SPEED1 is the singular attribute chosen in the regression model to classify the class of the dataset, T20BOLT. Therefore SPEED1 clearly has the greatest effect on determining the outcome of T20BOLT.



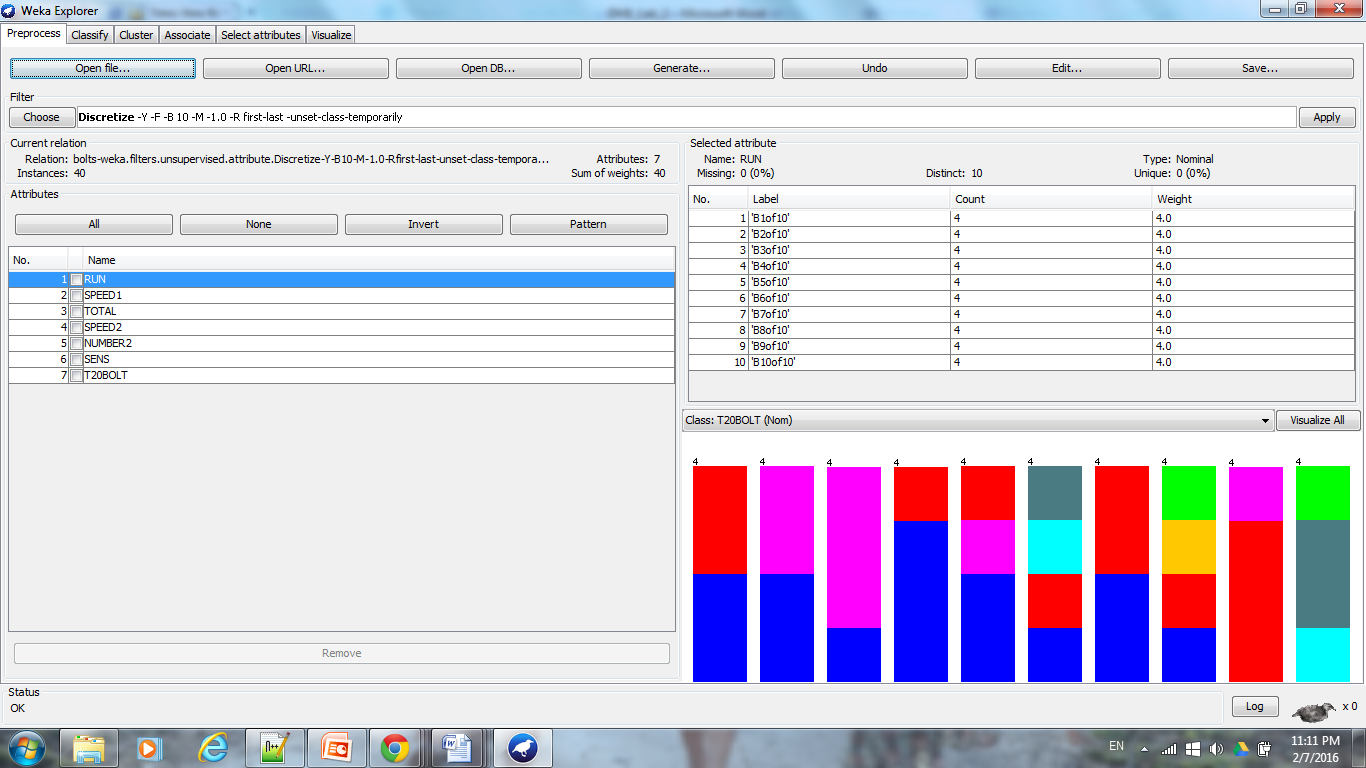
Comparison to J48 Decision Tree:

B2. As completed last week: The bolts.arff dataset was uploaded into Weka and filtered through the “Preprocess” phase. 8 attributes were listed for the data set, all of which are numeric. In order to change the numeric values into nominal values, “unsupervised”, “attribute”, and “discretized” was selected. Next, clicking on the heading produced a popup window (shown below) and the number of bins was selected to be 10, “ignoreClass” was changed to true, and “useBinNumber” was changed to true. Applying the filter onto the data set changed the numeric values to nominal values (as shown below). Finally, “removing” the Time attribute from the data set would create a data set which did not depend on the value of Time to determine the time needed by a machine to produce and count 20 bolts.

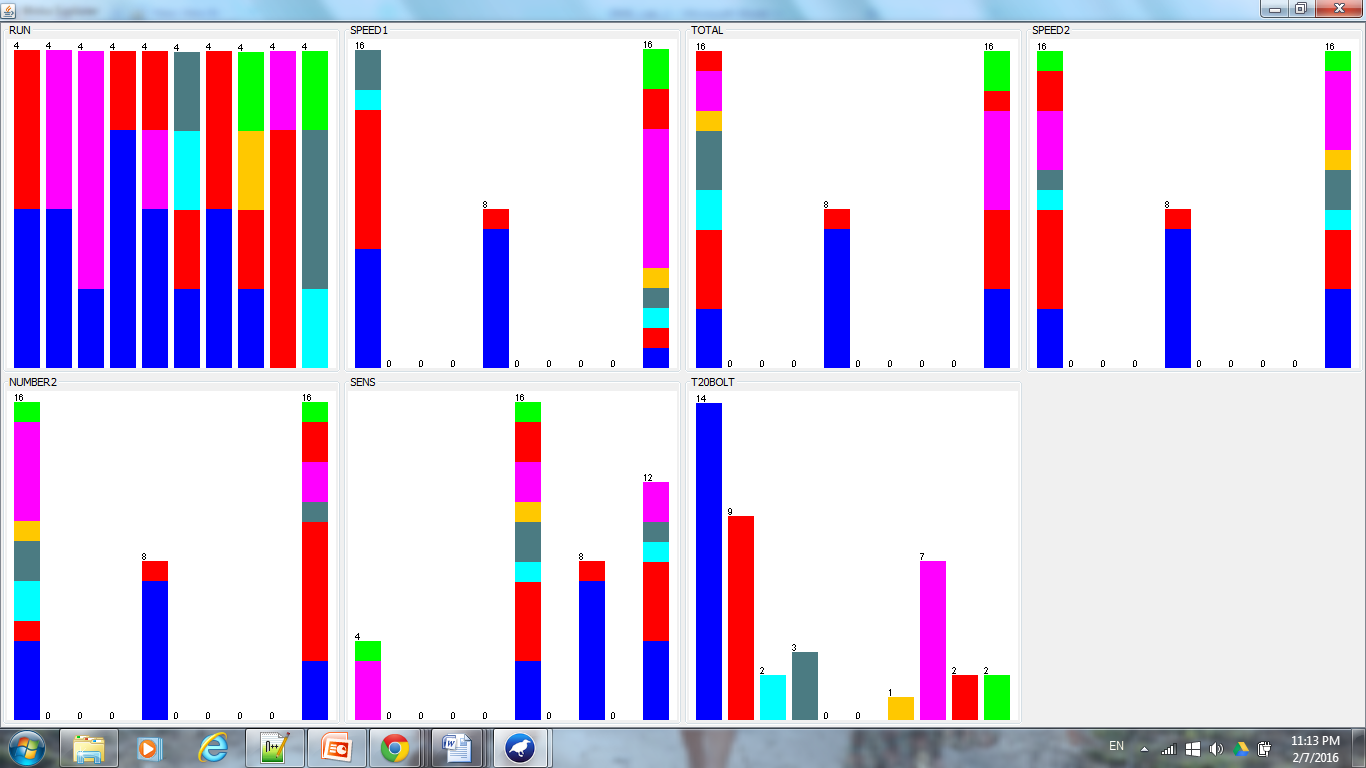
Before applying:



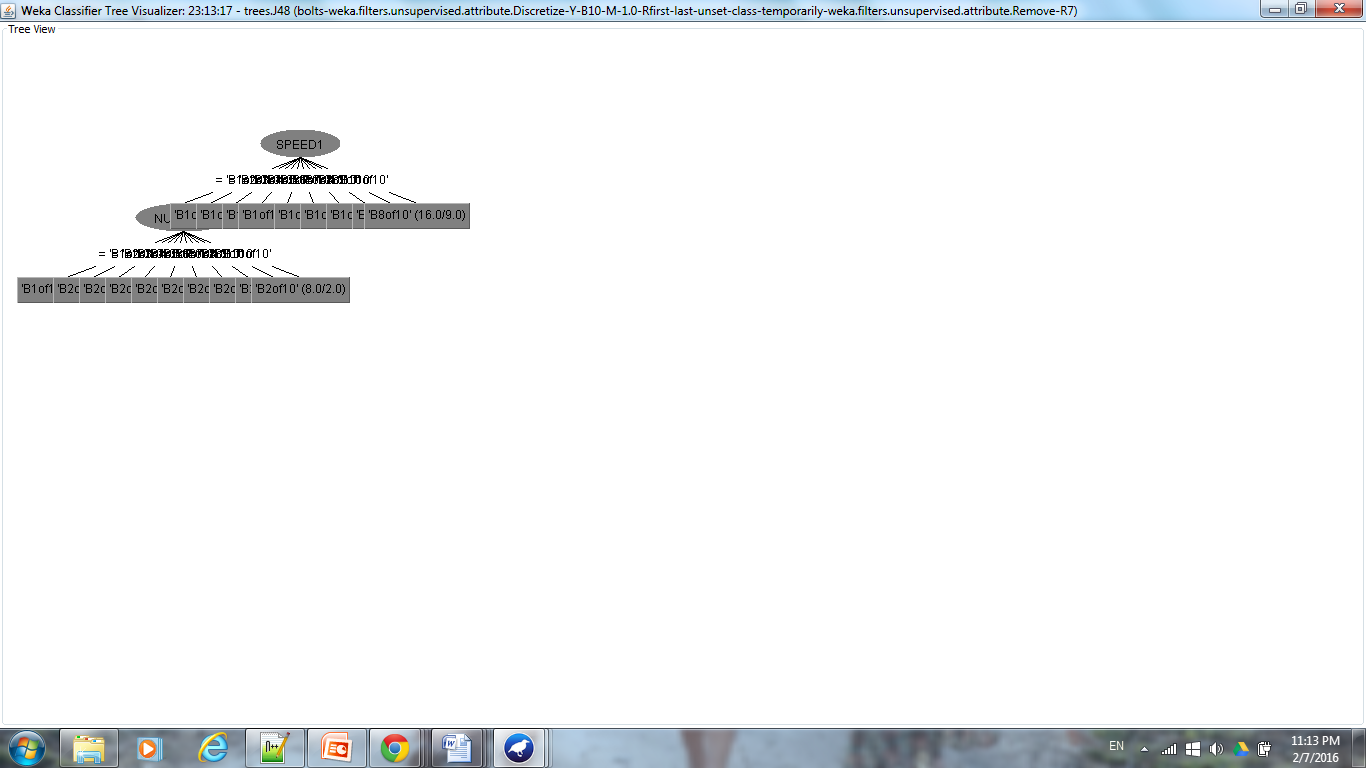
After applying (Note: type of data under “Selected Attribute” changed from “Numeric” to “Nominal”):



C2. “Visualize All” showed the attribute distributions for each of the instances. Below, it can be seen that each of the attributes are divided into 10 bins.



D2. Applying the J48 decision tree learning scheme, shows that in the absence of the Time attribute, SPEED1 and Number2 have the largest effect on determining which “Bin#” for T20BOLT an instance will be classified into. Therefore, it can be seen that the speed setting that controls the speed of rotation (SPEED1) of the plate at the bottom of the dish and the number of bolts to be counted at the second speed (NUMBER2) affect T20BOLT the most. According to the decision tree, by shortening the value of SPEED1, the shortest T20BOLT can be achieved.

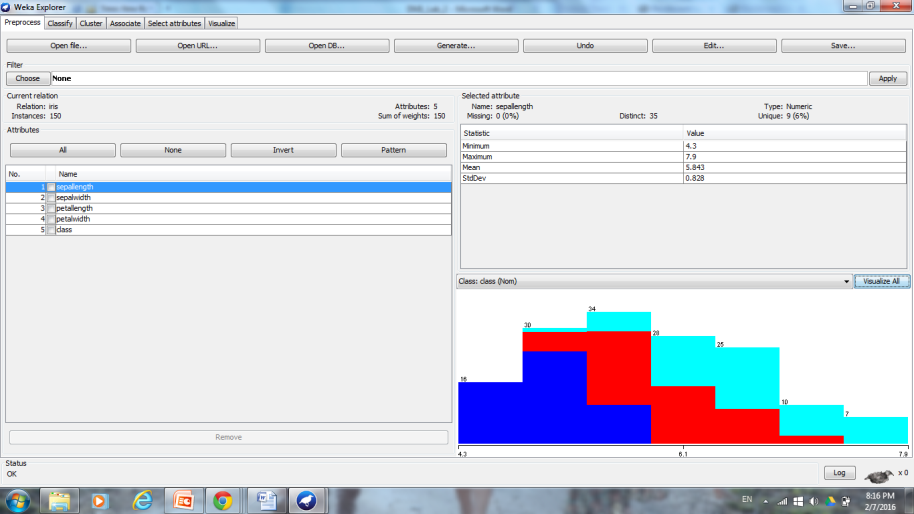


Comparison between numeric regression tree and nominal decision tree:

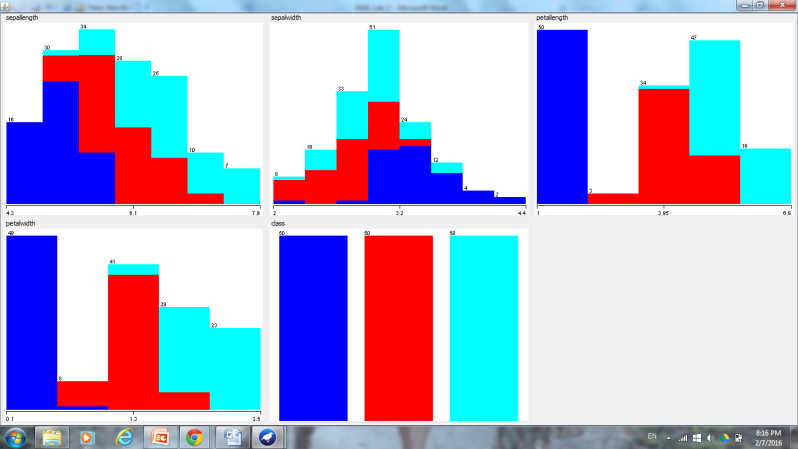
In comparing the 2 trees formed, the regression tree is simpler compared to the nominal decision tree. This is in part due to the fact that the raw data of bolts.arff are numeric therefore to create a simple and reliable model, a numeric learning scheme should be employed as opposed to a nominal learning scheme. Still, for both of the trees SPEED1 is the most important attribute.

1. Use a k-means clustering technique to analyze the iris data set. What did you set the ***k*** value to be? Try several different values. What was the random seed value? Experiment with different random seed values. How did changing of these values influence the produced model?

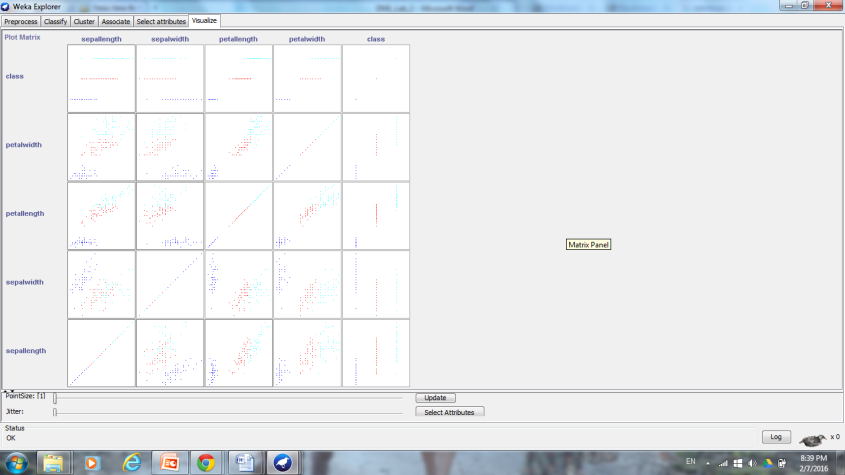
A. The iris.arff was uploaded into Weka, there was no need to further filter the data during the “Preprocess” phase. Note: 5 attributes exist in this data set.



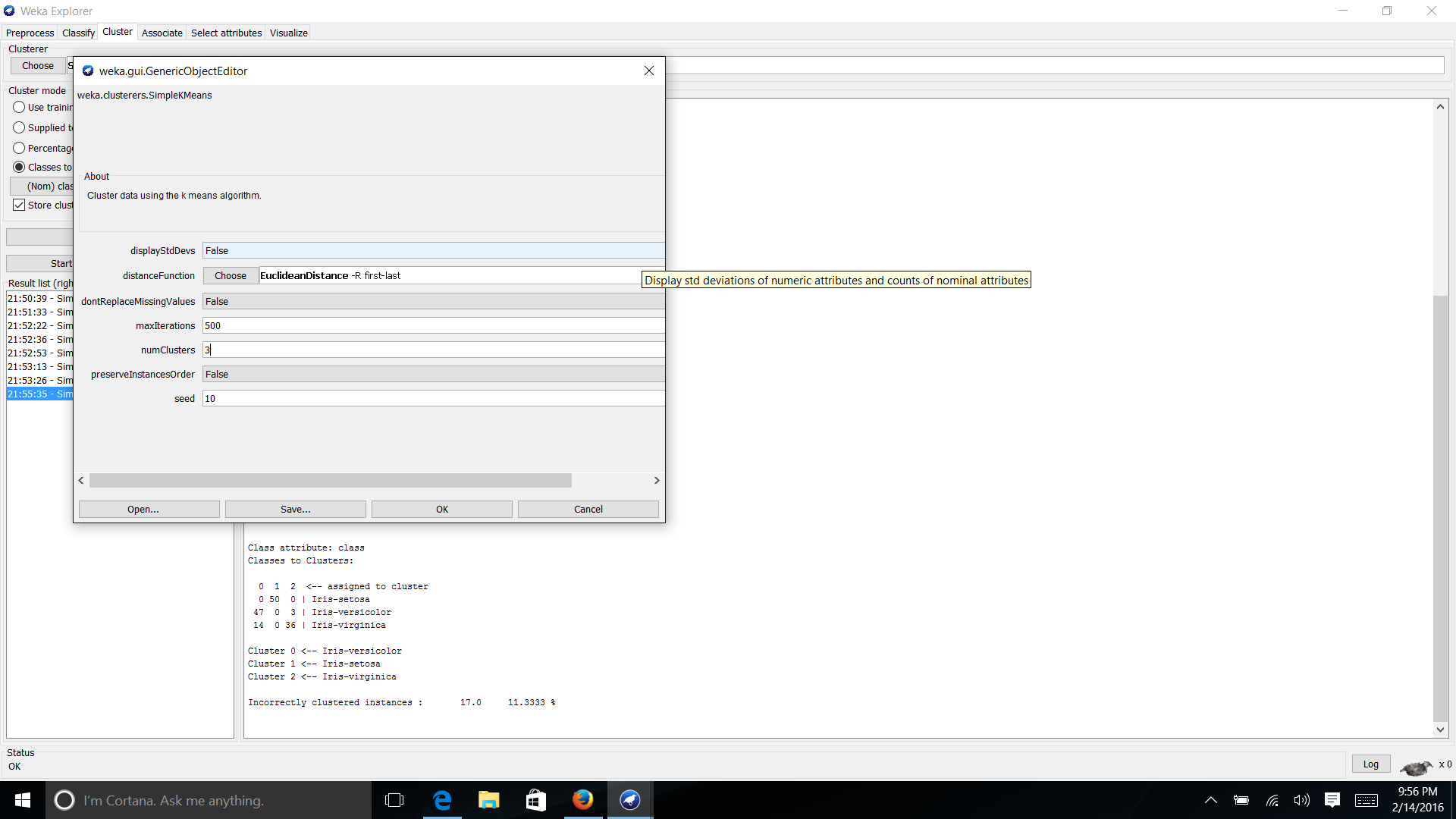
B. “Visualize all” was used to visual the distributions for each class. Where, attributes “sepalength” and “sepalwidth” follow positive skewed Gaussian distributions, the attributes “petallength” and “petalwidth” follow no concrete distribution, and the “class” attribute distributions are equal.



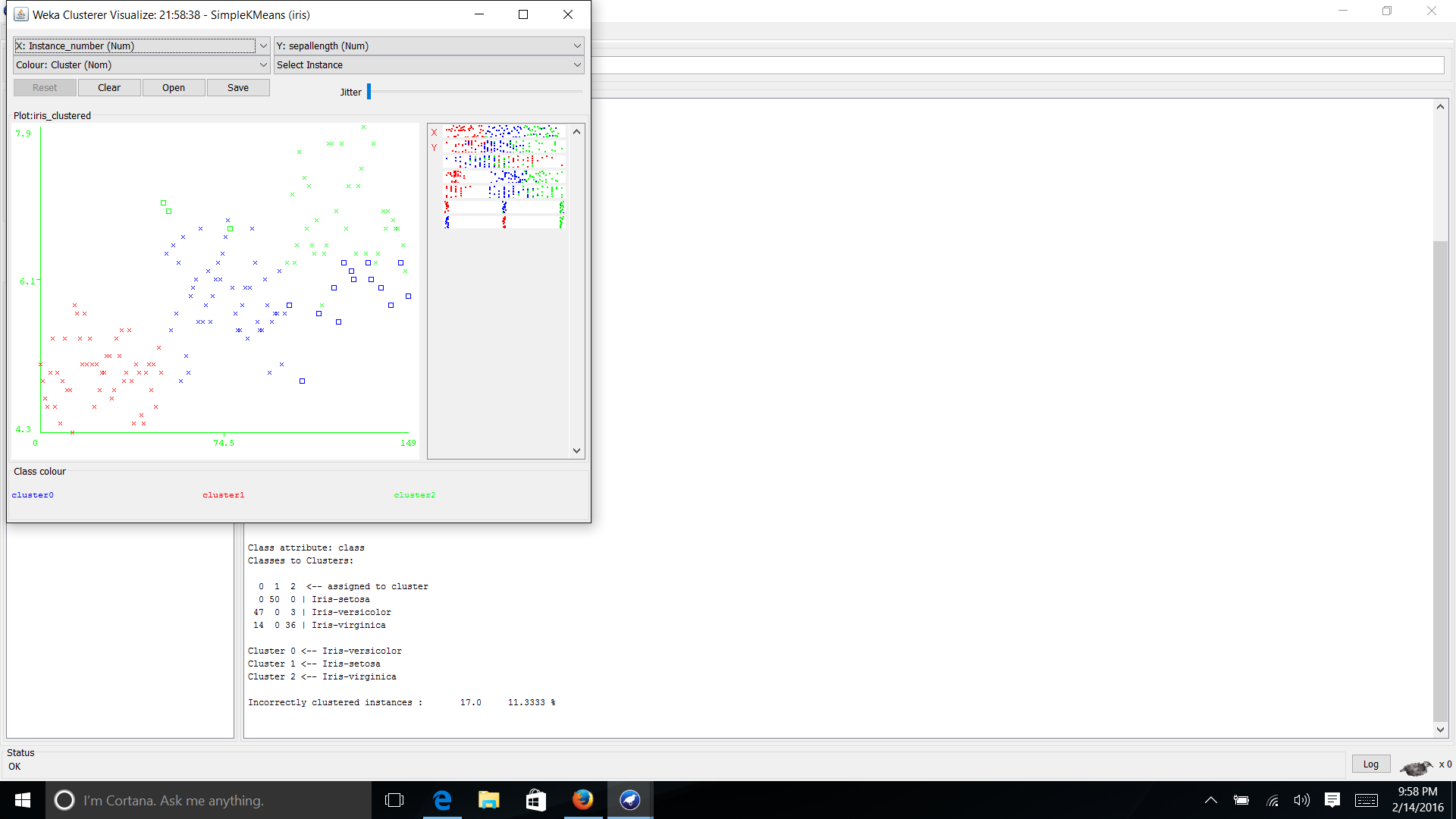
C. Utilizing the “Visualize” tab, clusters for attribute values can be seen with the most apparent clusters seen for petalwidth vs. sepalwidth/petallength and petallength vs. sepallength/sepalwidth/petalwidth. These attributes will be utilized in order to determine the class (Iris-setosa, Iris-versicolor, or Iris-virginica).



D. To apply the k-means clustering technique, under the “Cluster” tab Clusterers/SimpleKMeans is selected. At the pop-up window, “numClusters” was changed to 3 since the number of class attributes is 3, “seed” was maintained at 10.

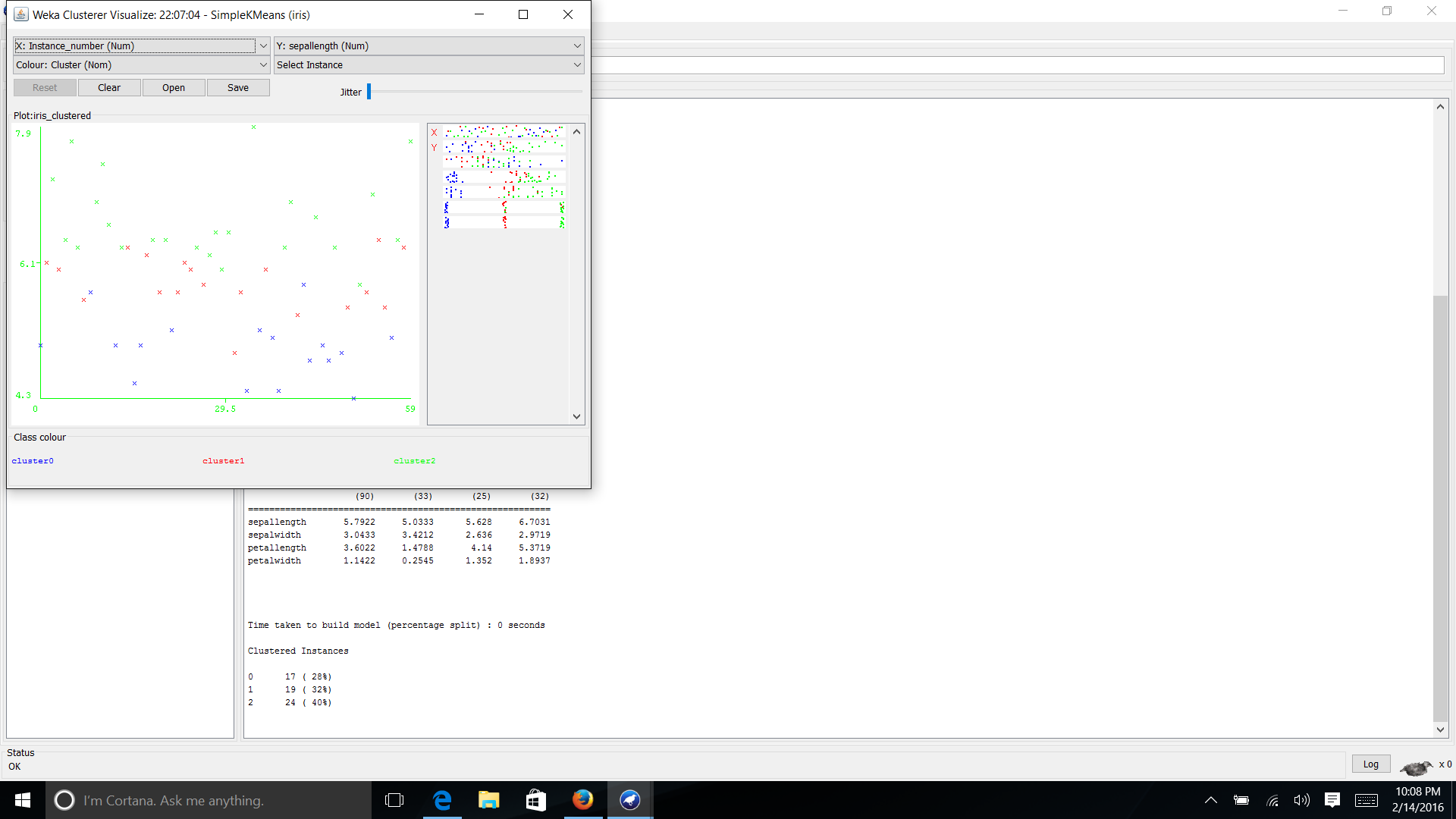


Note: for the first means of analysis, the class attribute was not ignored and the “classes to cluster evaluation” was set to “class (nom)”.



Three clusters were created with the “Clustered Instances” containing 61 (41%) in cluster 0, 50 (33%) in cluster 1, and 39 (26%) in cluster 2. Note: in the class distribution in Part B, Iris-setosa, Iris-versicolor, and Iris-virginica had equal distributions.

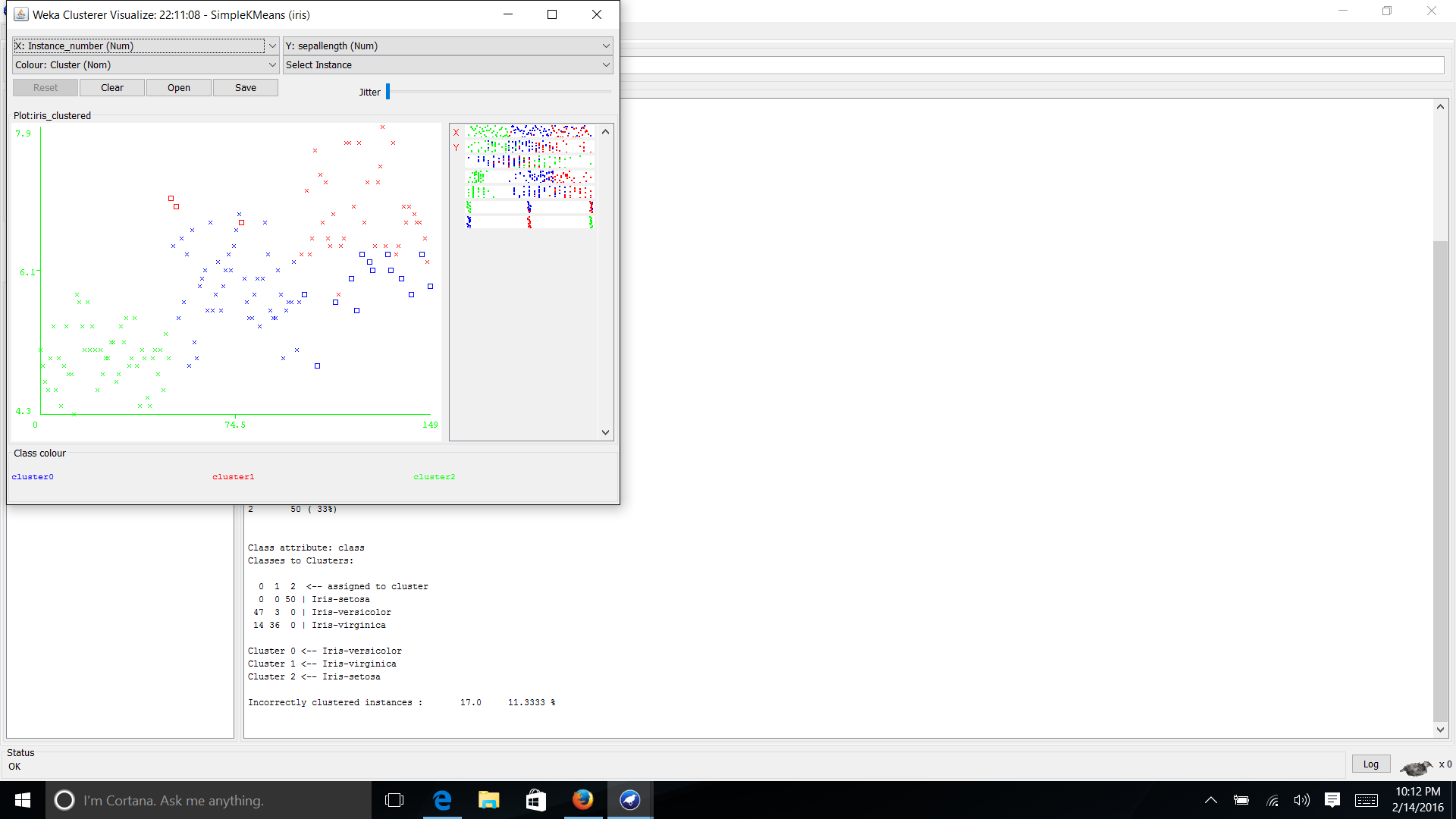
But as clustering is unsupervised, next the class attribute was ignored and only the four attributes - sepallength, sepalwidth, petallength, and petalwidth were considering in clustering to compare the clustering. Class (nom) was ignored and “percentage split” at “60%” was used.



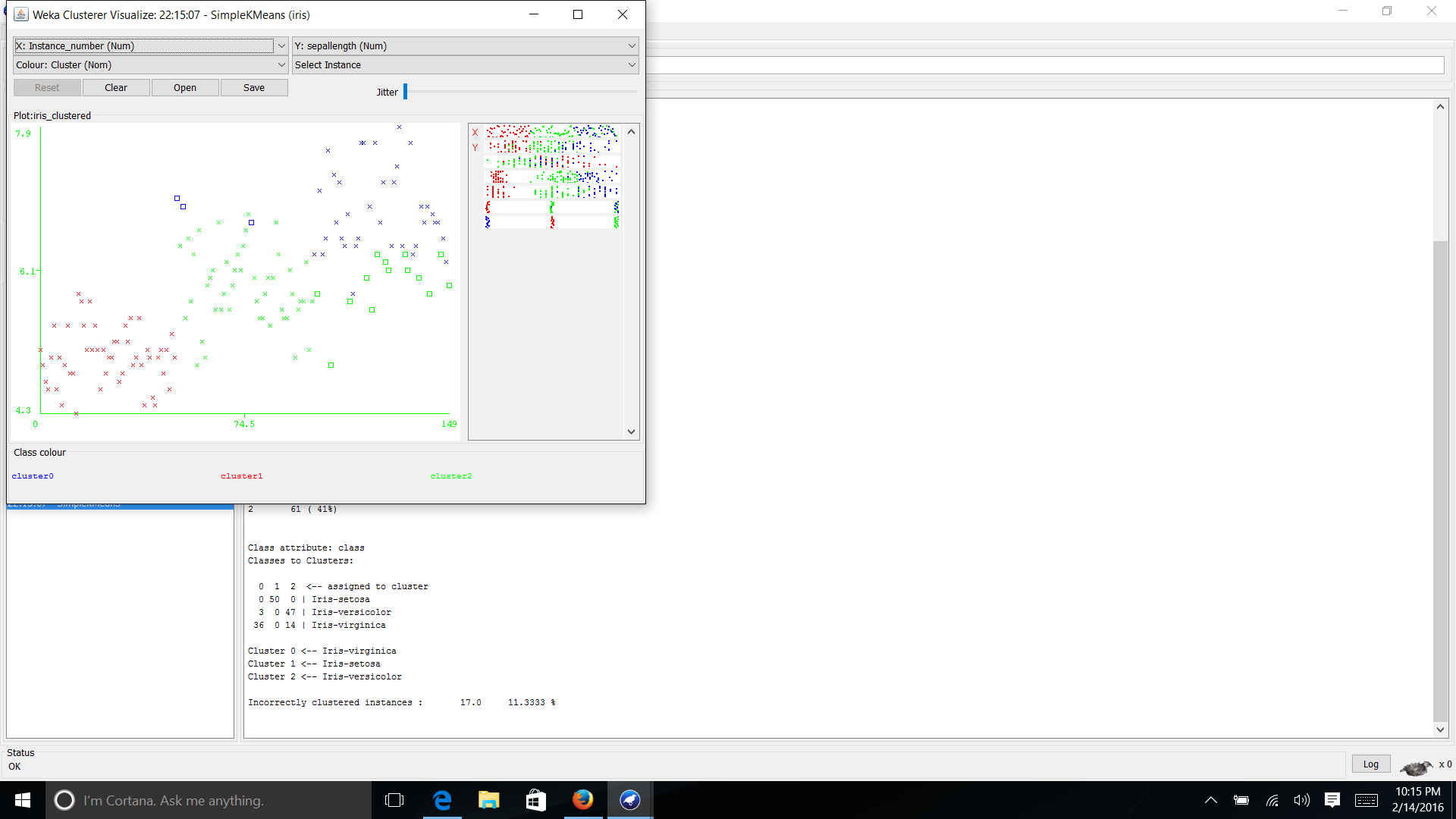
Making the above adjustments, changed the clustering vastly in comparison with the cluster centroids being located in different places then before.

Furthermore, parameter variations for the first cluster produced, where “class (nom)” is not ignored were made.

When increasing the “seed” from 10 to 50, the clusters looked similar to that shown above with 61 (41%) in cluster 0, 39 (26%) in cluster 1, and 50 (33%) in cluster 2.



When decreasing the “seed” from 10 to 5, the clusters looked similar to that shown above with 39 (26%) in cluster 0, 50 (33%) in cluster 1, and 61 (41%) in cluster 2.



Note: It is interesting that the ratio 39:50:61 is conserved when changing the “seed” number. The seed number (any integer) is the randomization for your initial K points. K represents the number of clusters. Because K-means is sensitive to initial points, you will have to try experimentation on the stability of your clusters with different seeds. However, K is user defined which could also be guided by domain knowledge and other practical factors.

Changing the number of clusters from 3 to 10, does not create nice clean clusters for all. Nicer clusters can be created after recursive measures of visualizing the clustering and determining how much variation between instances are acceptable.

