The University of Texas at San Antonio™

The Cyber Center for Security and Analytics

UNIVERSITY OF SOUTH CAROLINA

ZEEK INSTRUSION DETECTION SERIES

Lab 12: Developing Machine Learning Classifiers for Anomaly Inference and Classification

Document Version: 03-13-2020

Award 1829698
"CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers"

Contents

Overview

This lab introduces the application of machine learning in the network security field. The lab explains how to generate a decision table and decision tree to infer scan-related network traffic. The lab is designed to train and test a machine learning classifier using network traffic dataset.

Objectives

By the end of this lab, students should be able to:

1. Train a decision table to classify scan-related network traffic.
2. Train a decision tree to classify scan-related network traffic.
3. Test and modify the trained classifiers and review their output classifications on a test dataset.

Lab topology

Figure 1 shows the lab topology. The topology uses 10.0.0.0/8 which is the default network assigned by Mininet. The *zeek1* and *zeek2* virtual machines will be used to generate and collect network traffic.

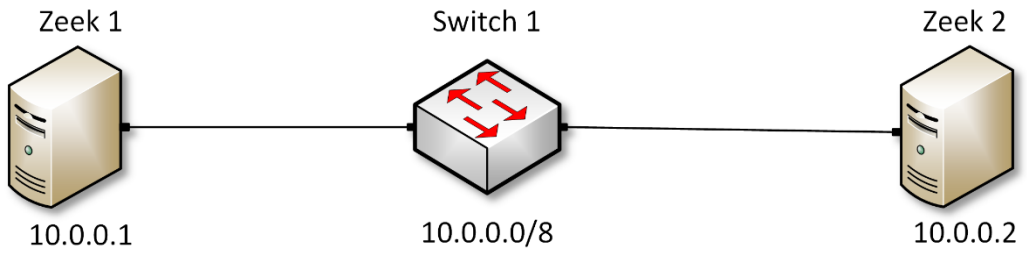


Figure 1. Lab topology.

Lab settings

The information (case-sensitive) in the table below provides the credentials necessary to access the machines used in this lab.

Table 1. Credentials to access the Client machine

Device	Account	Password
Client	admin	password

Table 2. Shell variables and their corresponding absolute paths.

Variable Name	Absolute Path
$ZEEK_INSTALL	/usr/local/zeek
$ZEEK_TESTING_TRACES	/home/zeek/zeek/testing/btest/Traces
$ZEEK_PROTOCOLS_SCRIPT	/home/zeek/zeek/scripts/policy/protocols

Lab roadmap

This lab is organized as follows:

1. Section 1: Introduction to Weka.
2. Section 2: Building a decision classifier with Weka.
3. Section 3: Reviewing the classifier's predictions on a test dataset.

1 Introduction to Weka

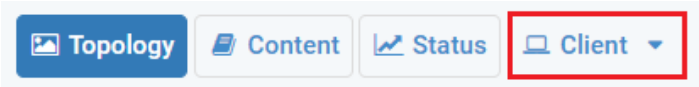
After formatting Zeek output logs into the ARFF files, Weka is now able to process them. Weka contains the algorithms necessary to develop a number of machine learning classifiers. More information on the Weka software can be found on their documentation pages. To access the following link, users must have access to an external computer connected to the Internet, because the Zeek Lab topology does not have an active Internet connection.

```
https://www.cs.waikato.ac.nz/ml/weka/documentation.html
```

In the following sections, we train a *DecisionTable* and a *J48 Decision Tree* classifier.

1.1 Starting Weka

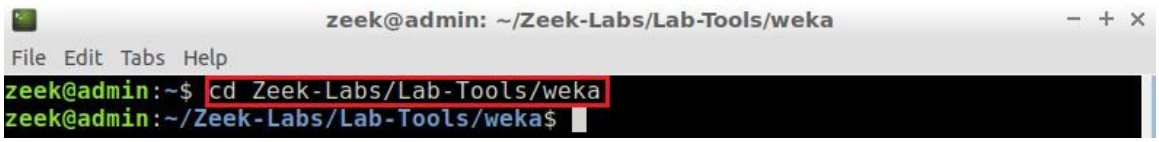
Step 1. From the top of the screen, click on the *Client* button as shown below to enter the *Client* machine.



Step 2. The *Client* machine will now open, and the desktop will be displayed. On the left side of the screen, click on the LXTerminal icon as shown below.

Step 3. Navigate to the Weka workspace directory.

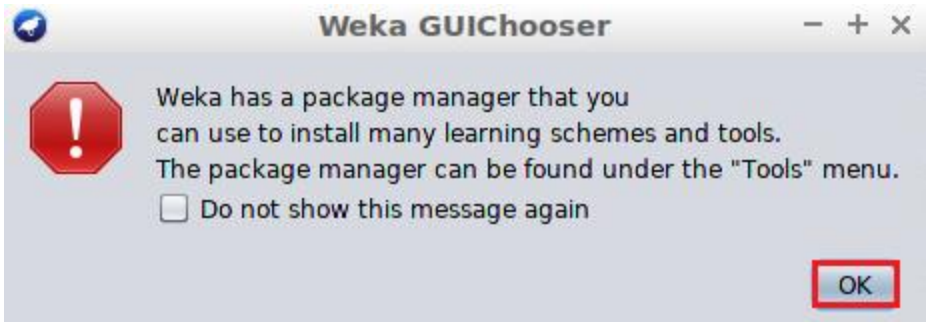
```
cd Zeek-Labs/Lab-Tools/weka
```



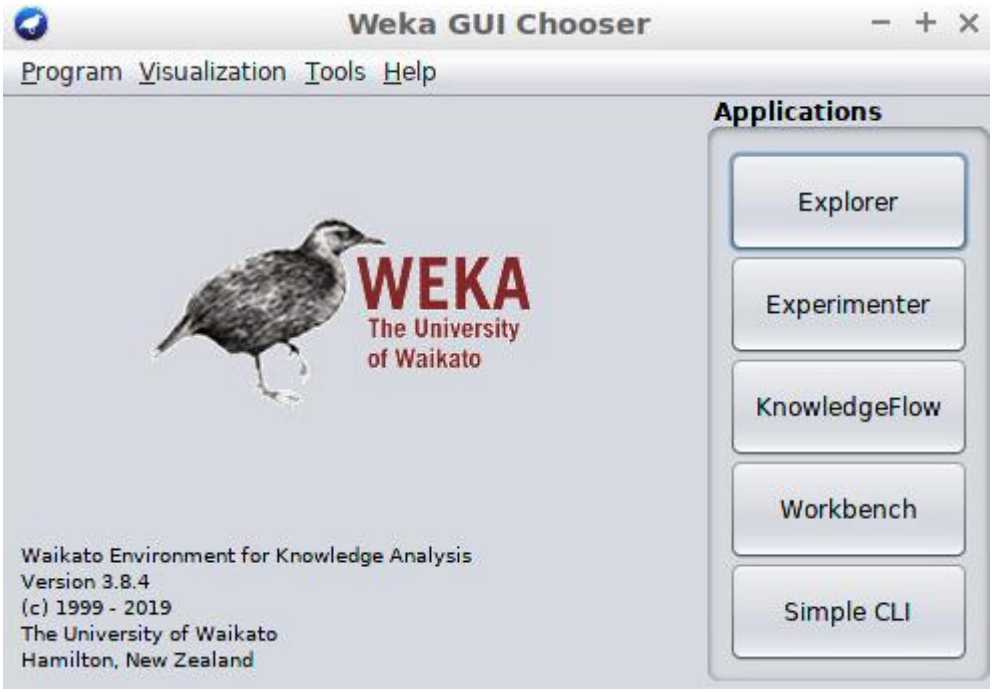
Step 4. Using Java, launch the Weka software.

```
java -jar weka.jar
```

Step 5. Once Weka has been loaded, a notification containing Weka related information will be displayed. Select the *OK* button to continue to the *Weka GUI Chooser panel*.



The *Weka GUI Chooser* panel will look similar to the following image.



Step 6. For this lab, we will be using the *Explorer* application. Click the *Explorer* button to launch the application.

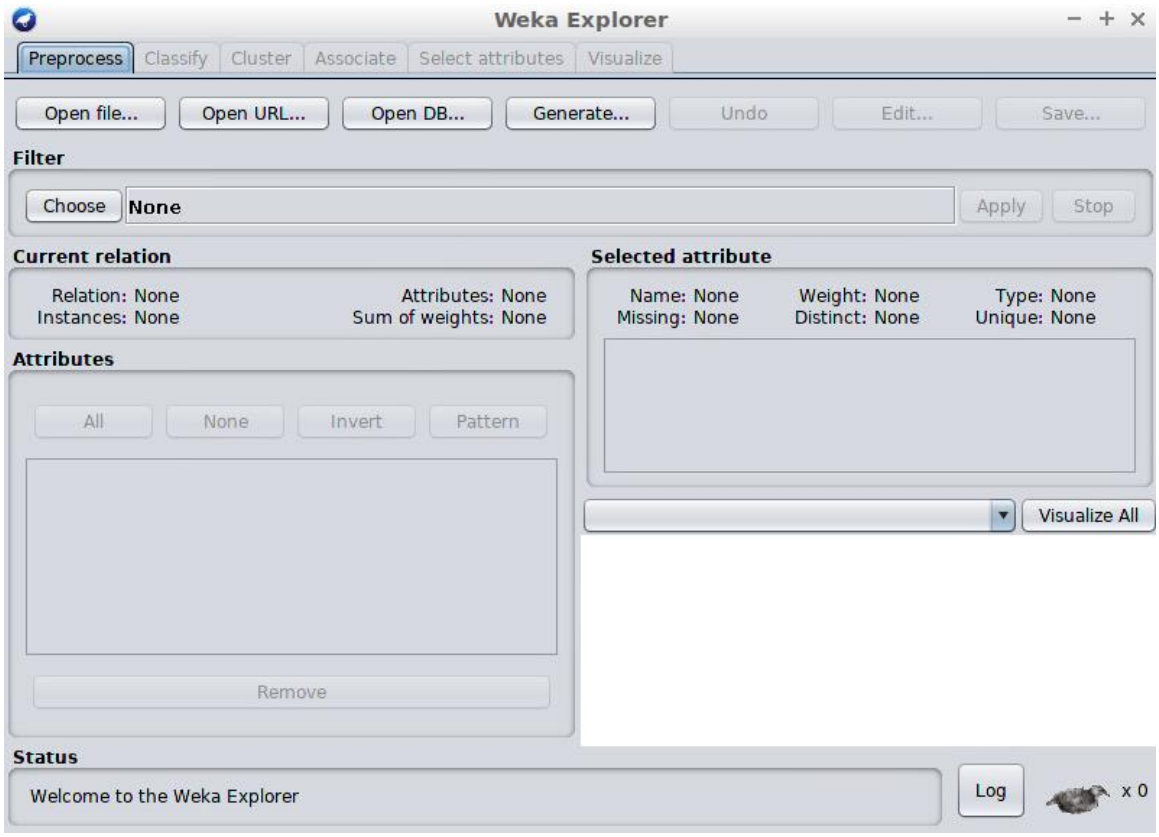
Weka has been successfully launched and we can proceed to the next section.

2 Importing a dataset into Weka

Once the *Explorer* application opens, a new GUI window will be displayed. Initially, this window has all options greyed out, indicating that we have not yet opened or loaded a dataset.

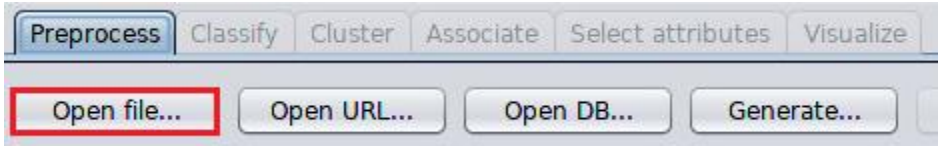
The *Explorer* panel contains a *Menu Bar* located at the top of the GUI window. There is a total of 6 additional panels, which contain related information necessary to train, test and visualize classifiers developed while using Weka. By default, the *Preprocess* panel will be selected.

The *Preprocess* panel is used to import a training dataset to be used for training a machine learning classifier. Features can be removed, randomized or appended within this panel.



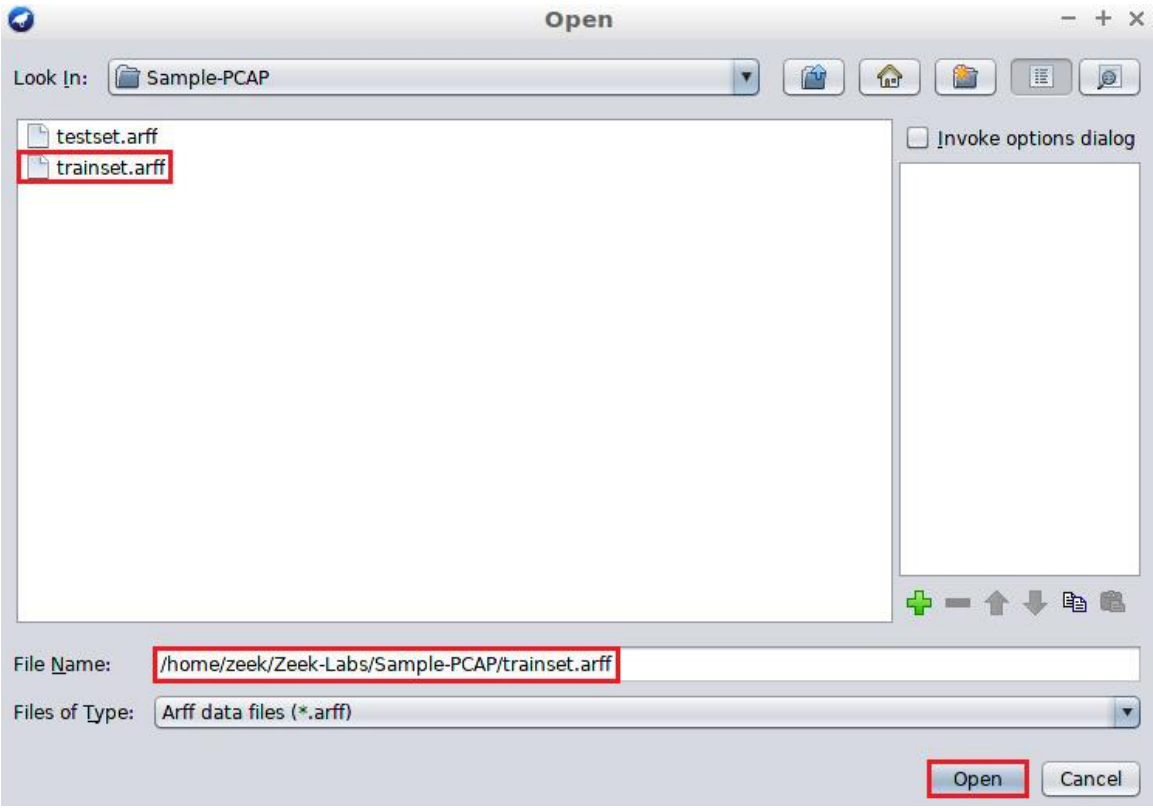
2.1 Loading the training dataset

Step 1. On the top left of the *Preprocess* window the *Open file* button can be found. Click the *Open file* button to load the training dataset.



Step 2. Enter the path to the *trainset.arff* file. Alternatively, use the GUI to navigate to the lab workspace directory to select the file. Use the *Open* button to load the *trainset.arff* file into Weka.

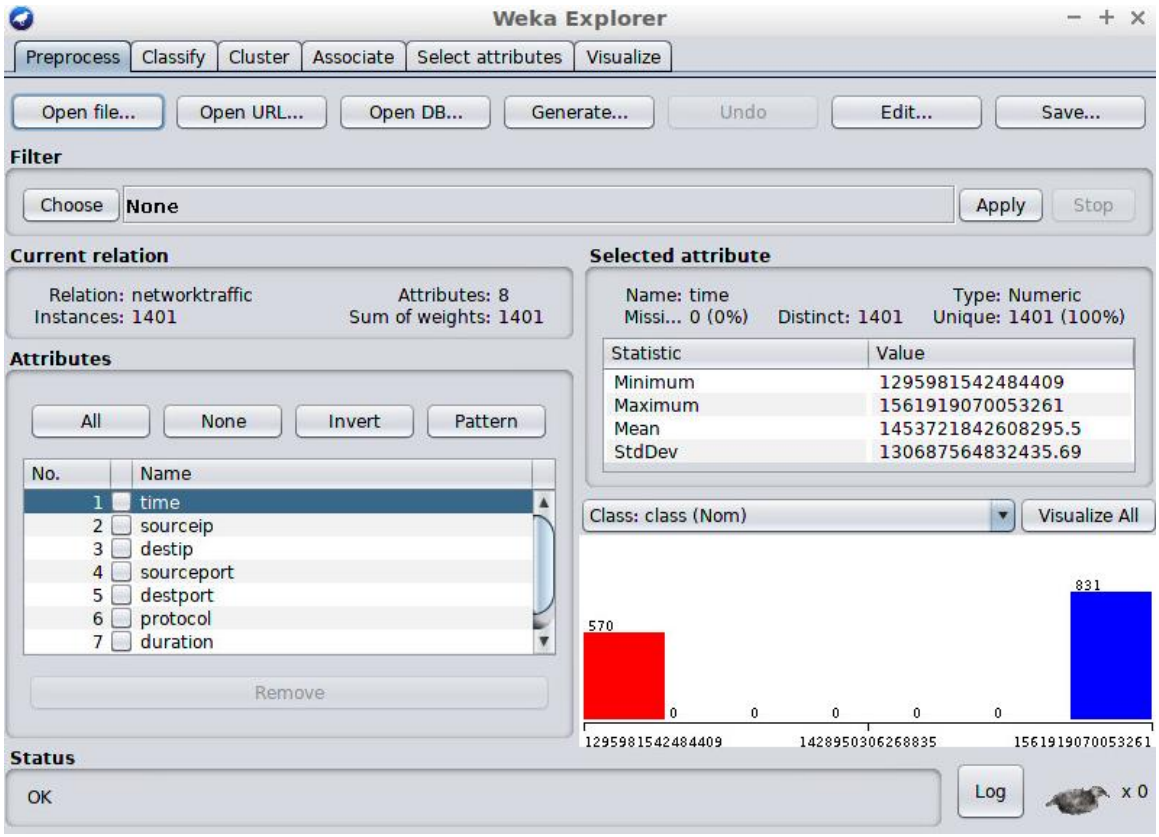
```
/home/zeek/Zeek-Labs/Sample-PCAP/trainset.arff
```



After click the *Open* button, the *Preprocess* panel will be updated to contain the *trainset.arff* file statistics.

Each section header has been highlighted with a red box. We can see that the *Current relation, Attributes* and *Selected attribute* sections have been updated to contain *trainset.arff* file data.

Step 3. Within the *Attributes* section, click the *class* feature to change the active attribute.



By selecting the *class* feature within the *Attributes* section, the *Explorer* panel will be updated to display the active feature.

Within the *Current relation* section, our dataset's name, *networktraffic*, is displayed. Additionally, it is shown that the dataset contains 1401 unique data objects (instances).

Within the *Selected attribute* section, the *class* labels added to the dataset in the previous lab are counted. The *trainset.arff* dataset contains 831 data objects labeled with a 1, belonging to the malicious class, while 570 data objects are labeled with a 0, belonging to the benign class.

At this point, *trainset.arff* dataset has been successfully loaded into Weka and we can begin filtering the data before training a machine learning classifier.

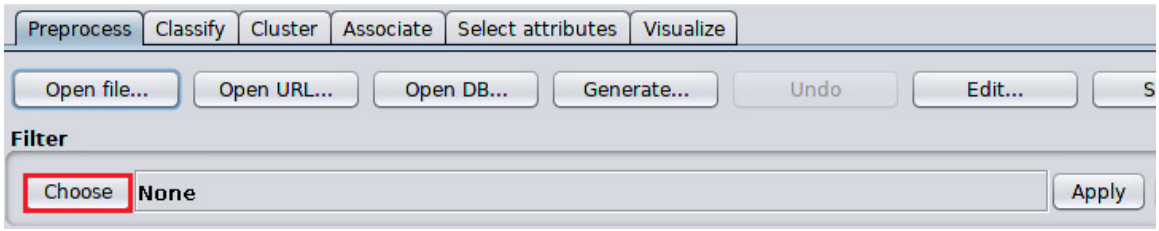
2.2 Filtering the training dataset

The majority of machine learning classifiers are unable to handle string attributes. For network analysts, source and destination IP addresses are valuable features that are often necessary for traffic analysis. However, these IP addresses are unable to be stored as string values when training a classifier.

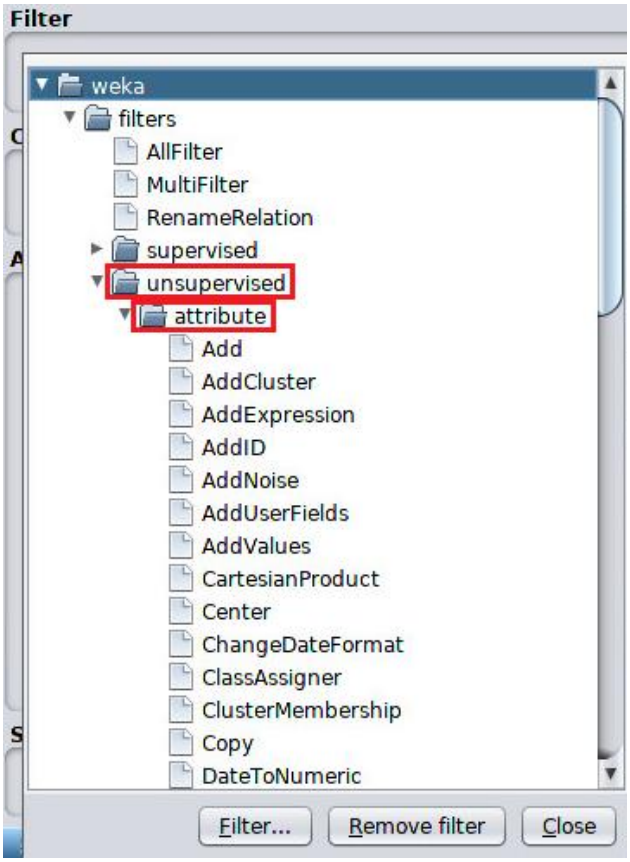
There is a number of ways to address this issue. If a network analyst were to know all of the unique IP addresses, when generating their ARFF dataset, they can create the nominal values similar to how we created the nominal protocol values.

Because Internet-scale traffic contains a very large number of unique IP addresses, the aforementioned process may not be feasible. Therefore, in the previous lab, we converted our source and destination IP addresses into numerical values. In this section, we will be using all unique iterations of the numerical values to generate a nominal list. By reformatting the IP addresses into numeric values using Terminal utilities, the Weka software will be able to select all unique IP addresses and convert them into a nominal feature set.

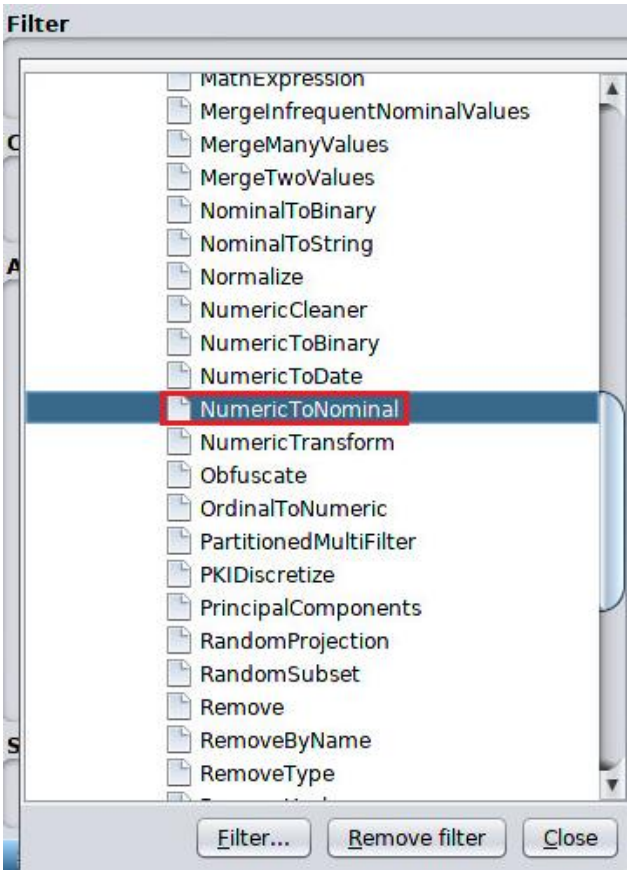
Step 1. Within the *Preprocess* tab, under the *Filter* section, click the *Choose* button.



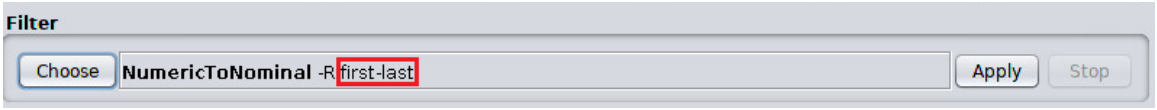
Step 2. Under the *unsupervised* option, select the *attribute* option to display a list of attribute-based filters.



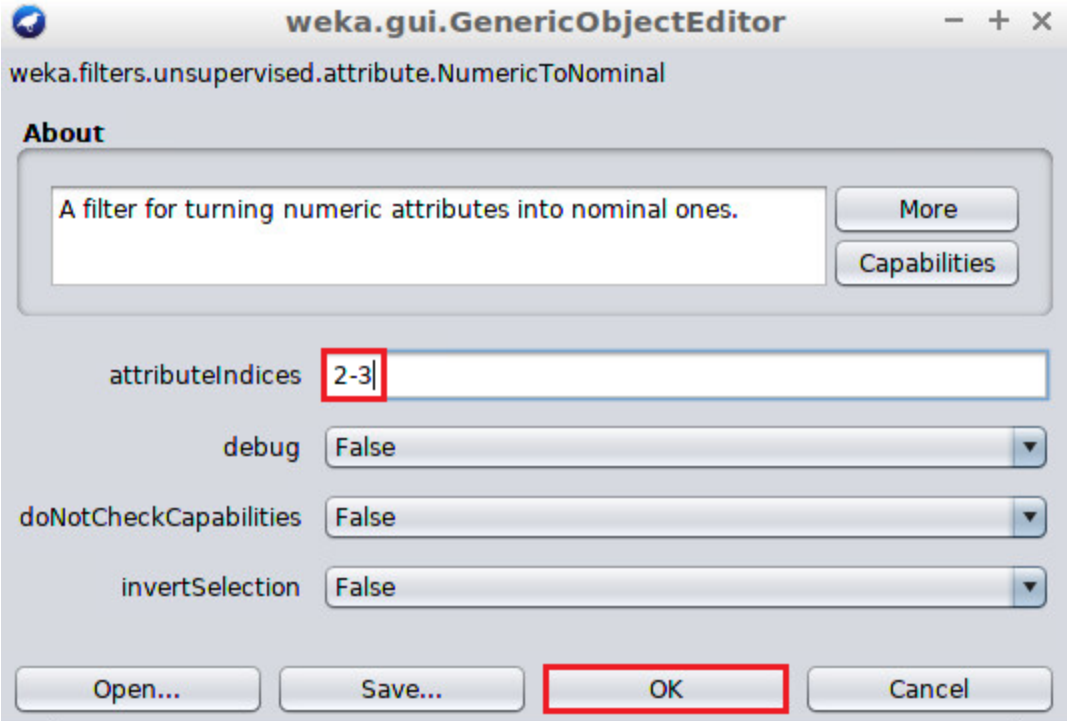
Step 3. Scroll to the *NumericToNominal* filter and double click to select it.



Step 4. Within the *Filter* section, click the *first-last* description to modify the filter.



Step 5. Update the *Indexes of the Attributes* to be filtered. Click the *Apply* button to edit the indexes.



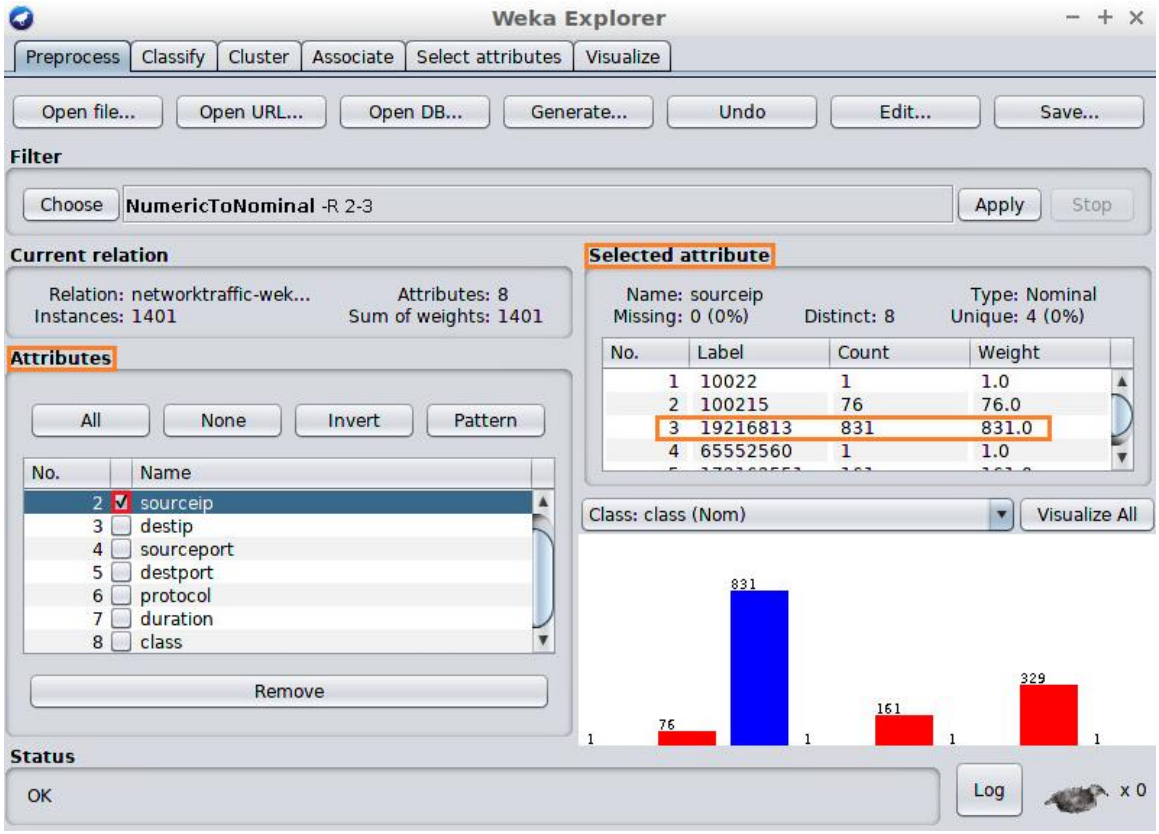
Step 6. On the right side of the *Filter* section, click the *Apply* button to apply the modified filter.

The source and destination IP addresses will now be converted to the Nominal feature type.

Step 7. Within the *Attributes* section, click the *sourceip* feature to change the active attribute.

By selecting the *sourceip* feature within the *Attributes* section, the *Explorer* panel will be updated to display the active feature.

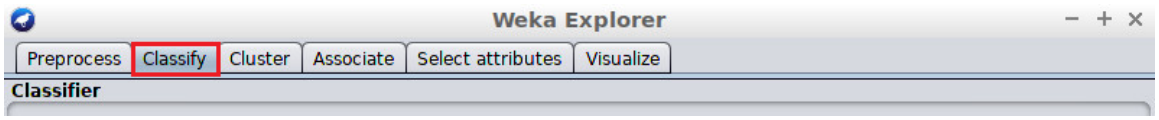
Within the *Selected attribute* section, the *sourceip* feature will now display the Nominal data objects. In the following image, the highlighted *sourceip* related to the scanning machine's IP address (192.168.1.3), displays 831 unique instances being recorded.



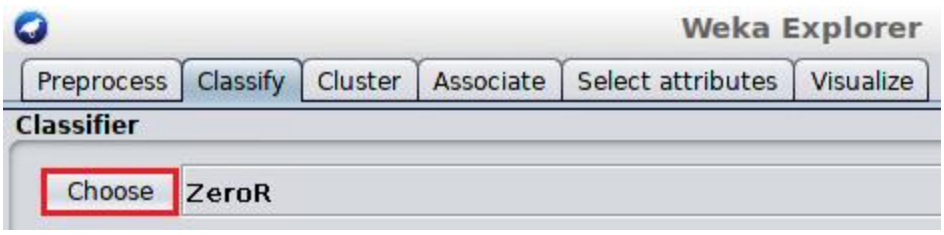
Additionally, the *Selected attribute* section will be updated to show new statistics for each feature. The updated *Selected attribute* section is displayed in the previous image.

2.3 Training a decision table classifier

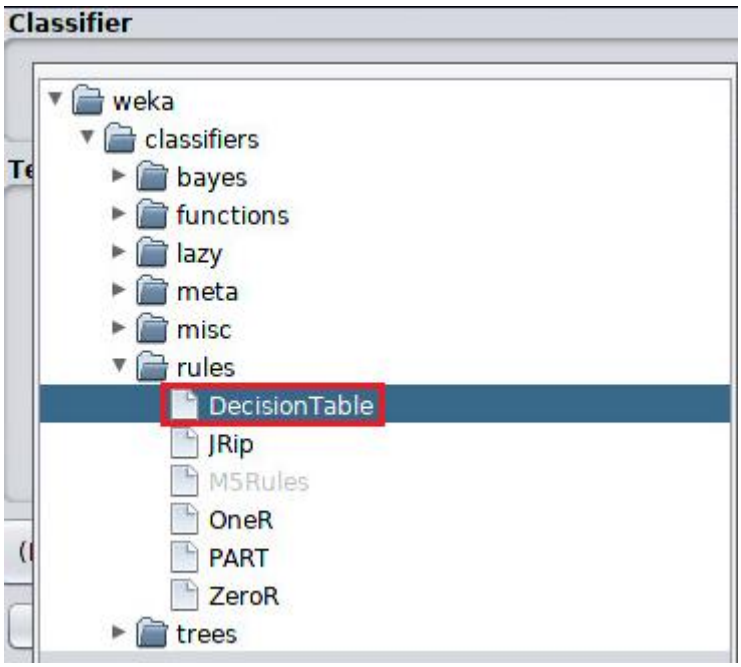
Step 1. Within the *Explorer* panel, click the *Classify* tab located at the top of the *Explorer* panel to switch to the *Classify* panel.



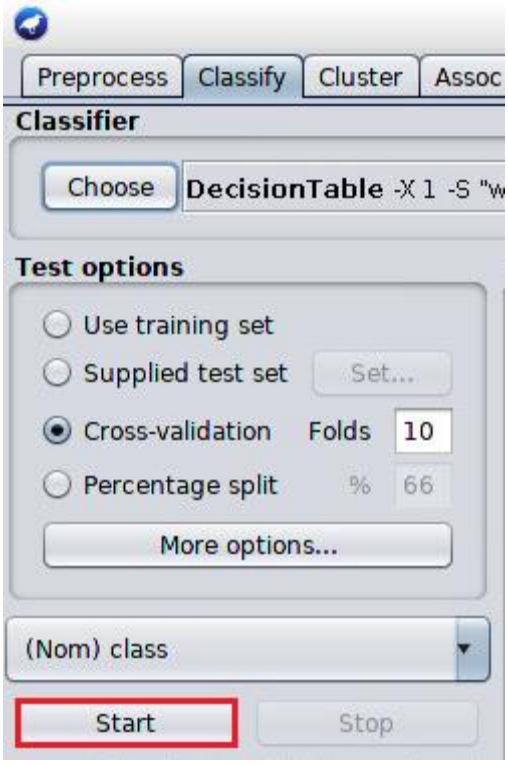
Step 2. Once the *Classify* panel has loaded, click the *Choose* button within the *Classifier* section to select which machine learning classifier we are developing.



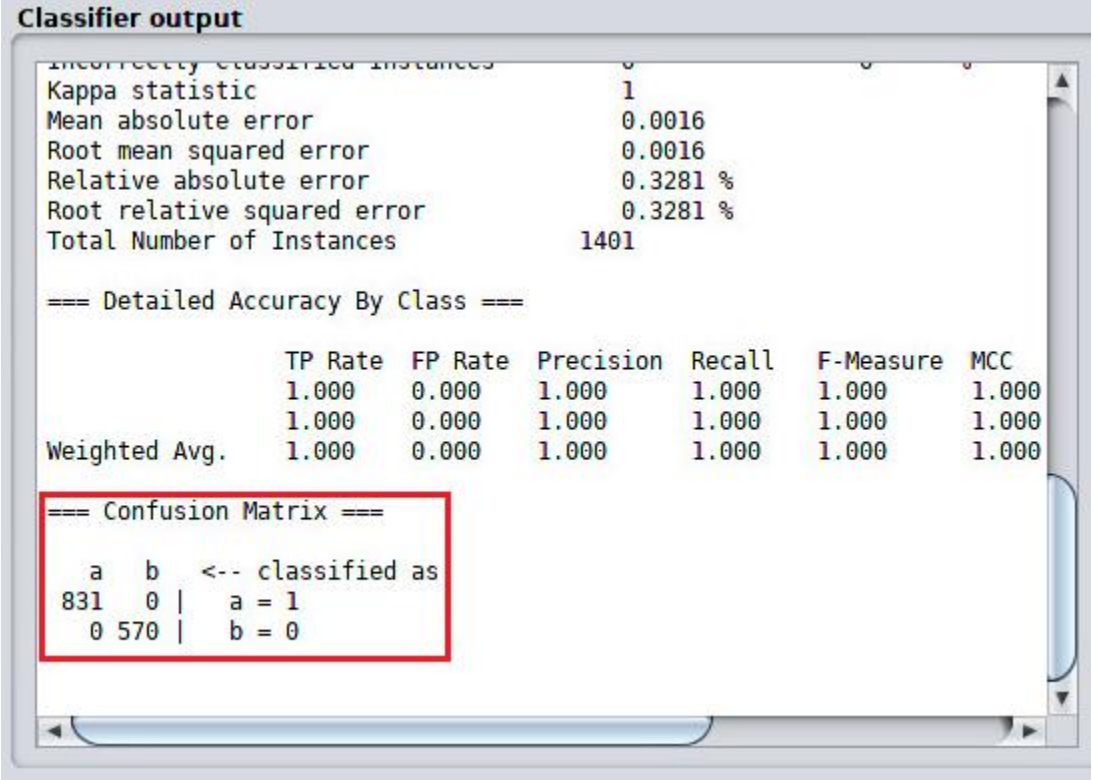
Step 3. Under the *rules* collection, double-click with your mouse to select the *DecisionTable* classifier.



Step 4. Under the *Test options* section, click the *Start* button to begin training the classifier. Notice the *Classifier* section has been updated to display the *DecisionTable* classifier.



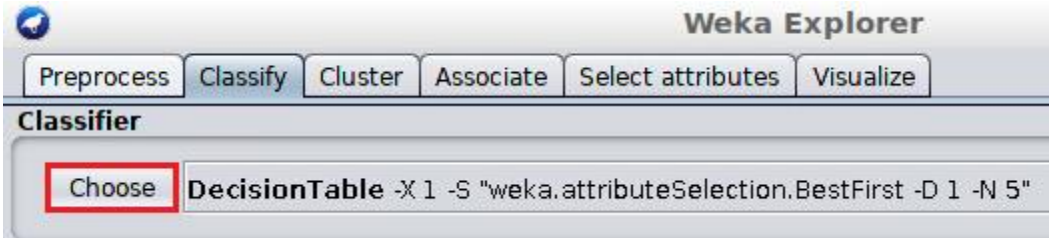
Step 5. See the *Decision Table* classifier's results.



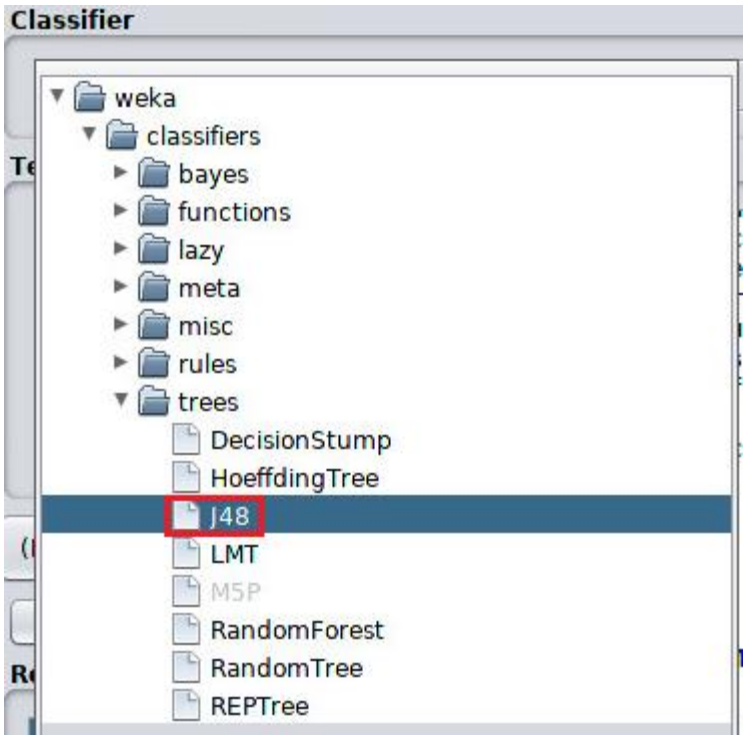
Within the *Result list* section we can see our new *Decision Table* that has been trained with the *transet.arff* dataset. Within the *Classifier output* section, we can see the prediction results for the *Decision Table* classifier. The *Confusion Matrix* depicts that the classifier had a 100% accuracy when predicting labels after being trained.

2.4 Training a decision tree classifier

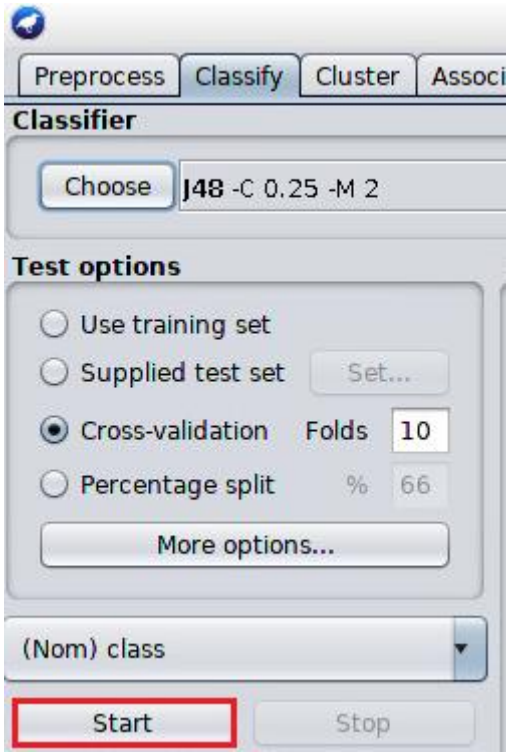
Step 1. Click the *Choose* button within the *Classifier* section to select which machine learning classifier we are developing.



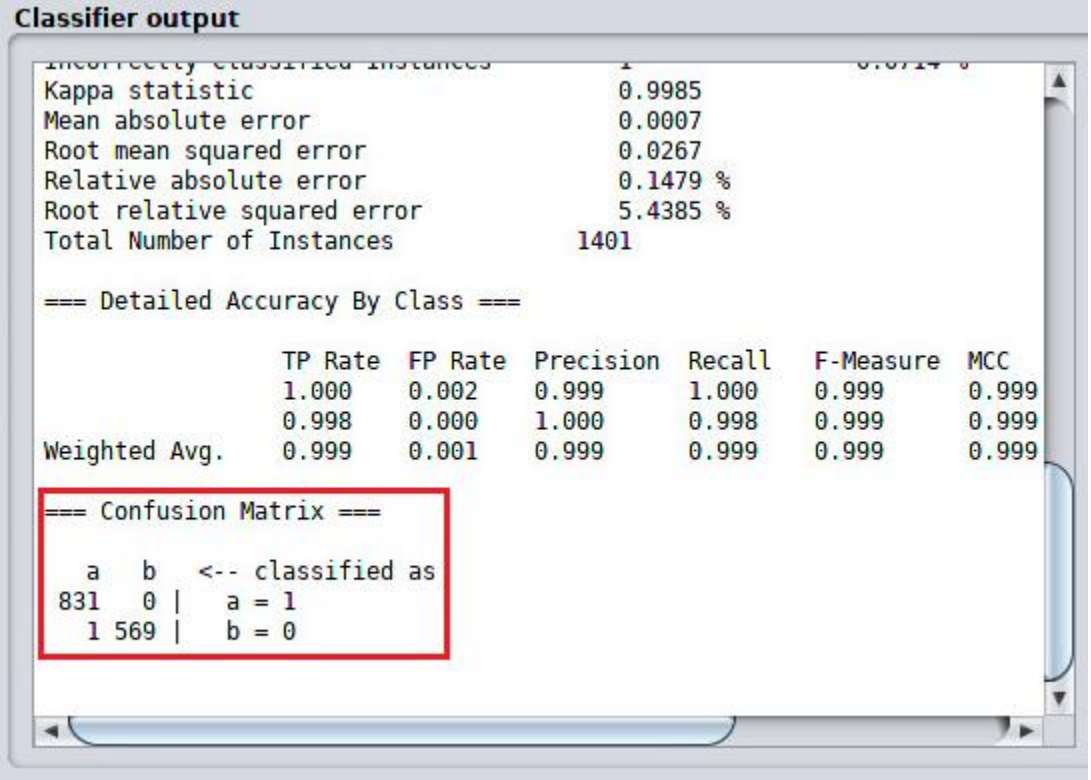
Step 2. Under the *trees* collection, double-click with your mouse to select the *J48* decision tree classifier.



Step 3. Under the *Test options* section, click the *Start* button to begin training the classifier. Notice the *Classifier* section has been updated to display the *J48* classifier.



Step 4. See the *J48 Decision Tree* classifier's results.

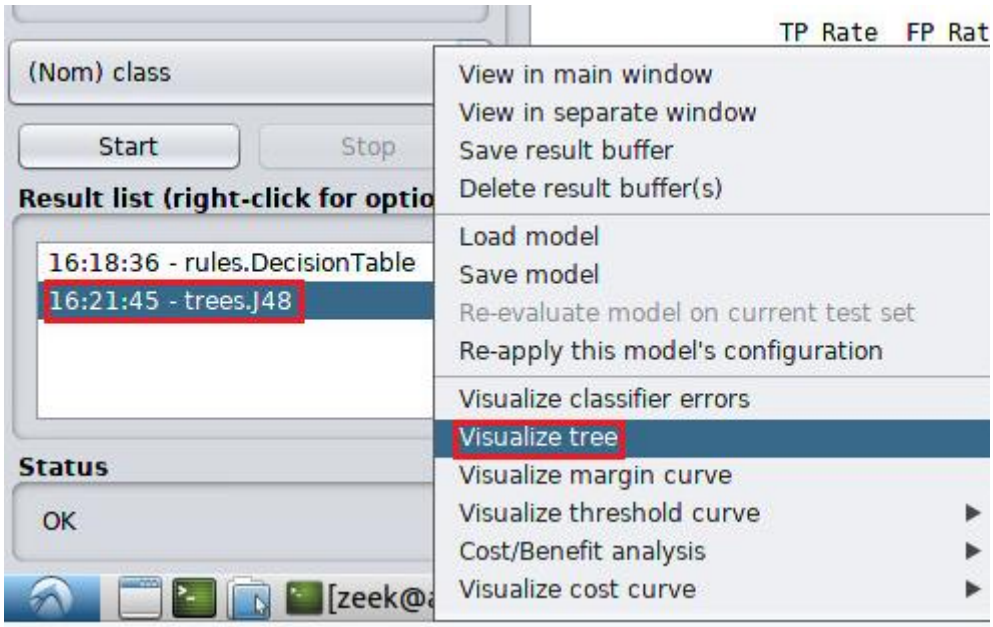


Within the *Result list* section we can see our new *J48 Decision Tree* that has been trained with the *transet.arff* dataset. Within the *Classifier output* section, we can view the prediction results for the *Decision Tree* classifier. The *Confusion Matrix* depicts that the classifier did not have a 100% accuracy when predicting labels after being trained and misclassified a single malicious data packet as benign.

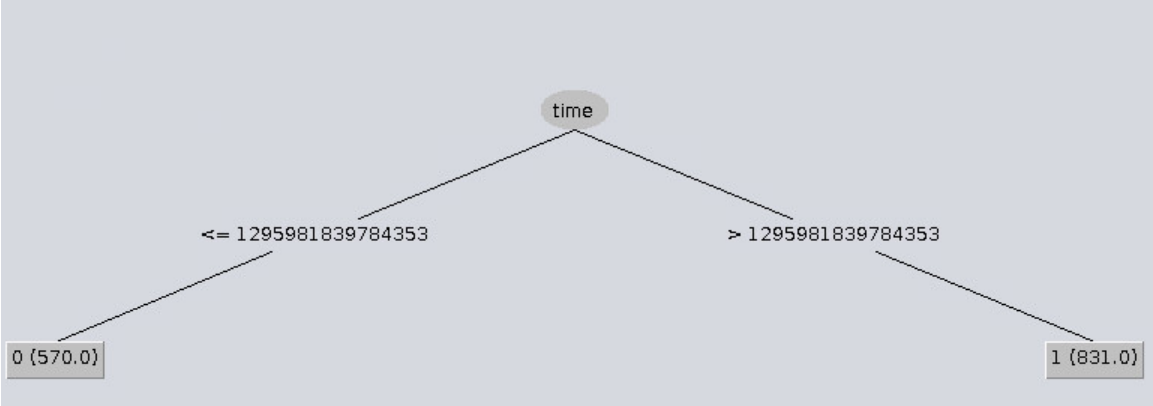
2.4.1 Updating the decision tree classifier

Because our *J48 Decision Tree* has made an error in predicted a label, we can attempt to remove or add additional features to increase the classifier's accuracy.

Step 1. Right click the *J48 Decision Tree* under the *Result list* section to display more options. Click to *Visualize* the *J48 Decision Tree.*



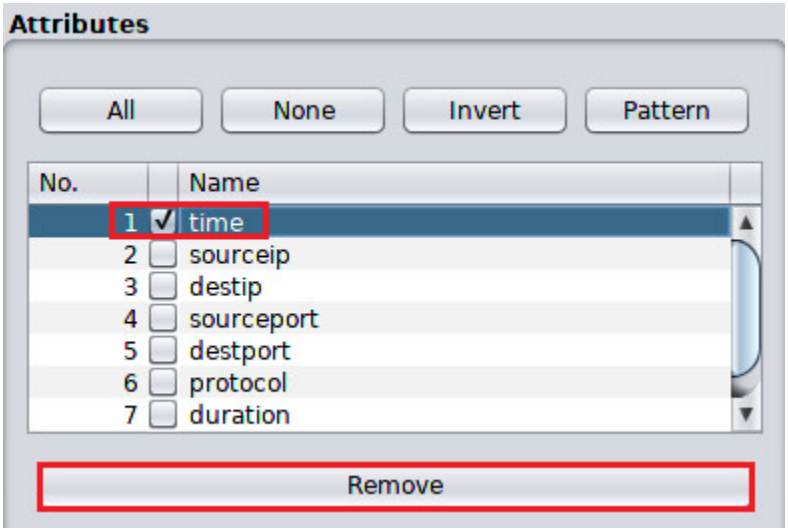
Step 2. View the *Visualized J48 Decision Tree*.



We can see the *time* feature column was the only decision node within the tree. For the purposes of this lab, the datasets were collected at varying times; therefore, the decision tree had an over reliance on the *time* feature to determine when the malicious and benign events took place.

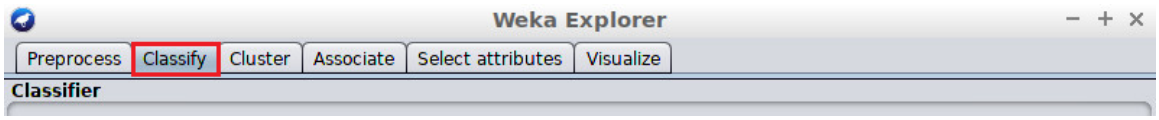
Step 3. Click the top right ☒ button to close the Tree Visualizer window then, within the *Explorer* panel, click the *Preprocess* tab located at the top of the *Explorer* panel to switch to the *Preprocess* panel.

Step 4. Once the *Preprocess* tab has loaded, click the *time* feature within the *Attributes* section and select the *Remove* button.

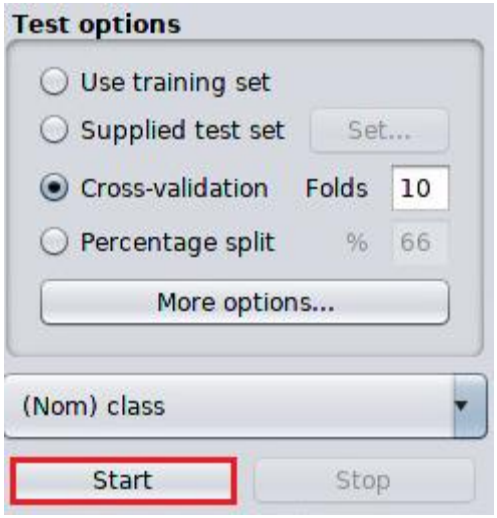


With the *time* feature removed, we can retrain our decision tree to view the new accuracy.

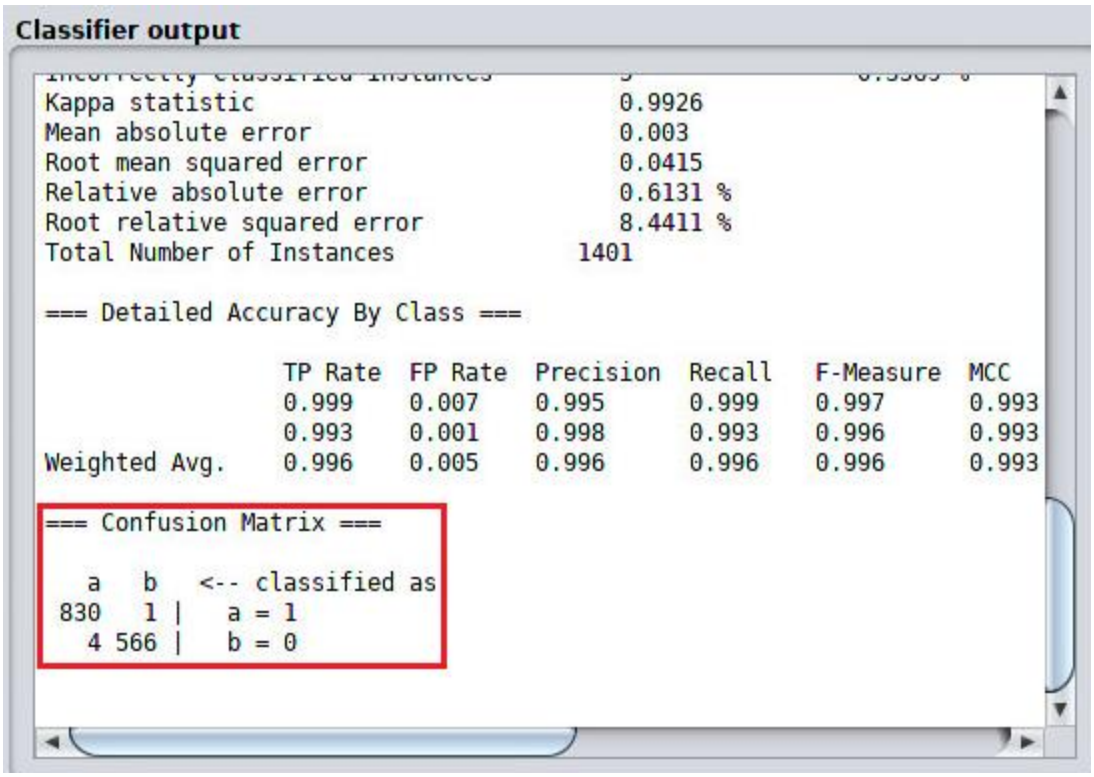
Step 5. Within the *Explorer* panel, click the *Classify* tab located at the top of the *Explorer* panel to switch to the *Classify* panel.



Step 6. The *J48 Decision Tree* should still be selected. Under the *Test options* section, click the *Start* button to begin training the classifier. Notice the *Classifier* section has been updated to display the new *J48* classifier.



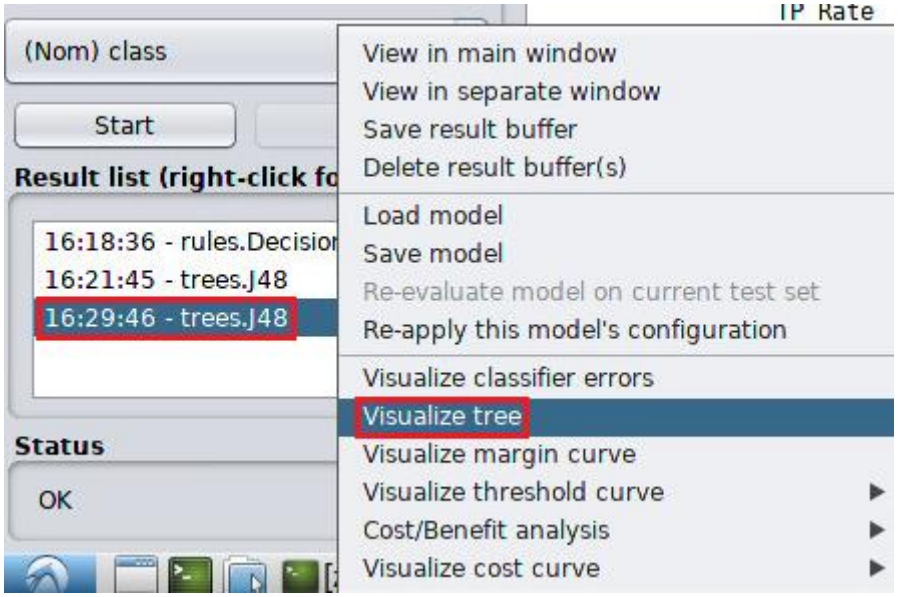
Step 7. See the *J48 Decision Tree* classifier's results.



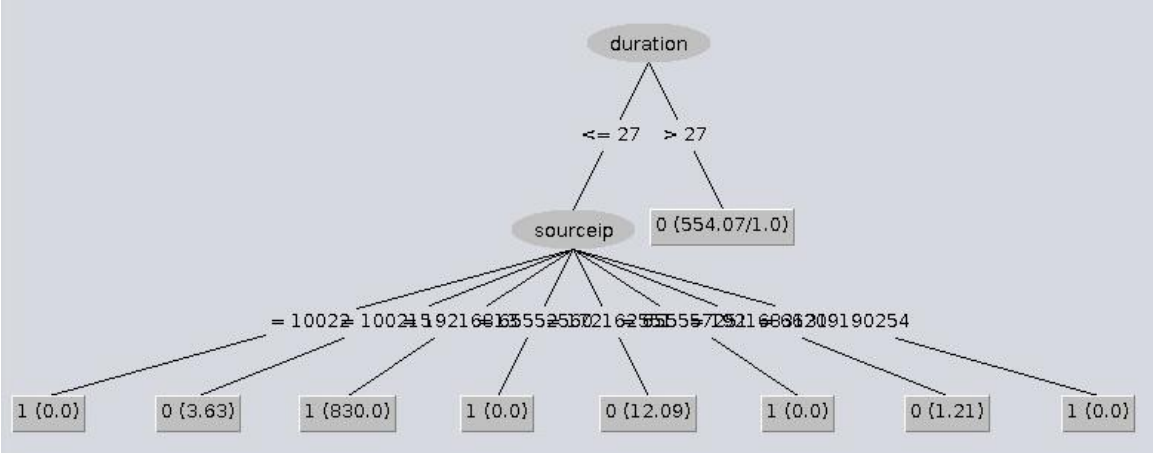
Within the *Result list* section we can see our new *J48 Decision Tree* that has been trained with the *transet.arff* dataset. Within the *Classifier output* section, we can view the prediction results for the *Decision Tree* classifier. The *Confusion Matrix* depicts that the classifier actually had a worse accuracy than the previously trained *J48 Decision Tree*.

In this example, we highlight the importance of choosing the best fit features when training a classifier. In a real-time network environment, it may take multiple tests before discovering which features are necessary for classifying a specific anomaly.

Step 8. Right click the newest *J48 Decision Tree* under the *Result list* section to display more options. Click to *Visualize* the *J48 Decision Tree.*

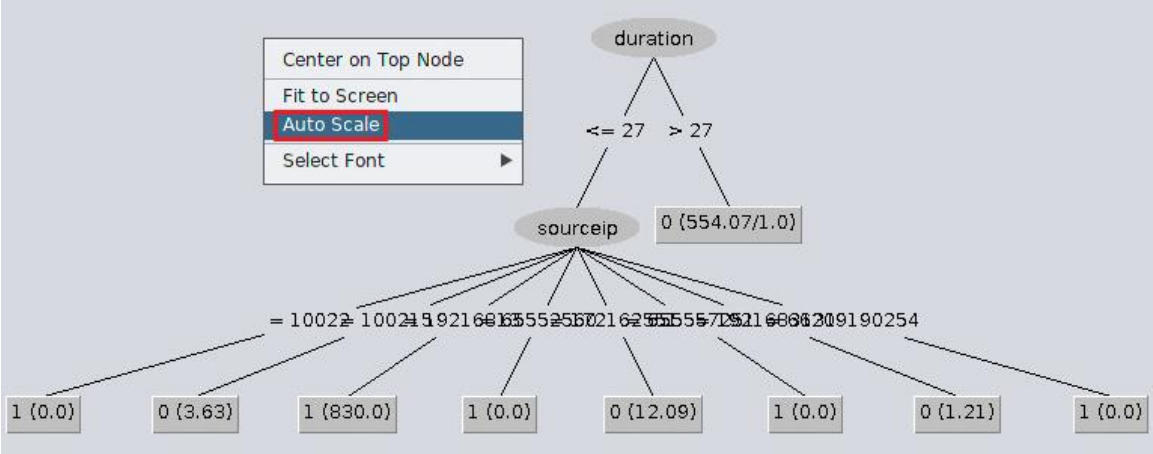


Step 9. View the *Visualized J48 Decision Tree*.

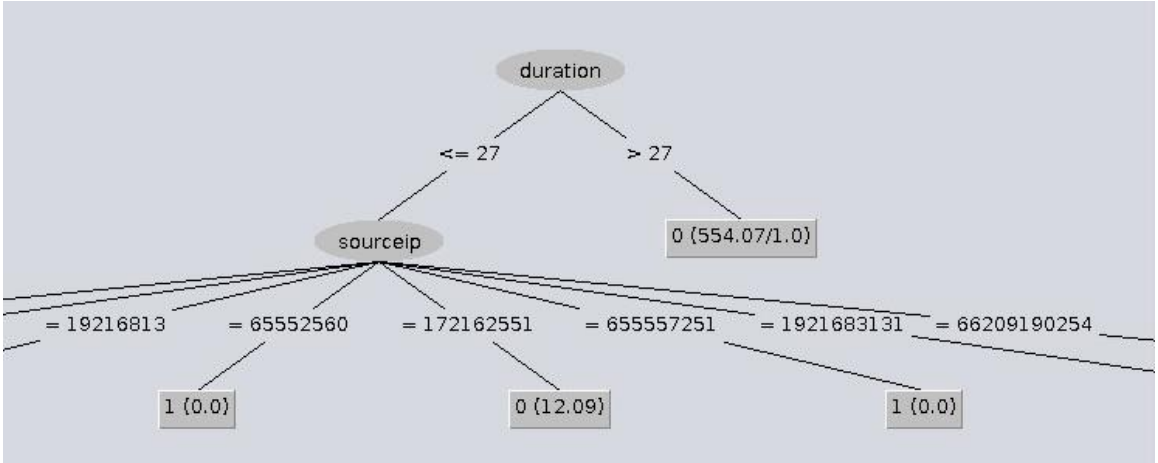


Because the *Decision Tree* has a larger number of nodes, we are unable to see some of the decision thresholds. The following steps will explain how to scale the *Visualized* tree.

Step 10. Right click on the *Visualized J48 Decision Tree* and select the *Auto Scale* option.



Step 11. Click the top right ☒ button to close the Tree Visualizer window then, open the *Visualized J48 Decision Tree*.



Here we can see the new *J48 Decision Tree* has multiple layers and decision nodes. The *duration* feature has replaced the *time* feature as the root node, and the *sourceip* feature is used to further classify the dataset. However, because this tree has reduced accuracy, we will be continuing the lab by using the *Decision Table* created initially.

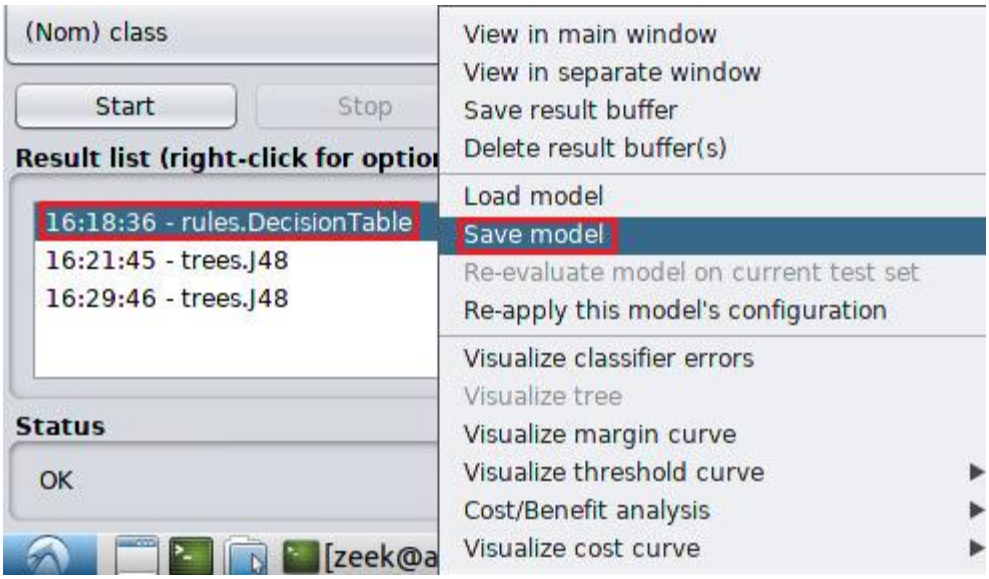
3 Reviewing the classifier's predictions on a test dataset

Now that we have determined that the *Decision Table* was a more accurate classifier, we can begin testing the classifier's accuracy using the test dataset.

3.1 Saving the decision table

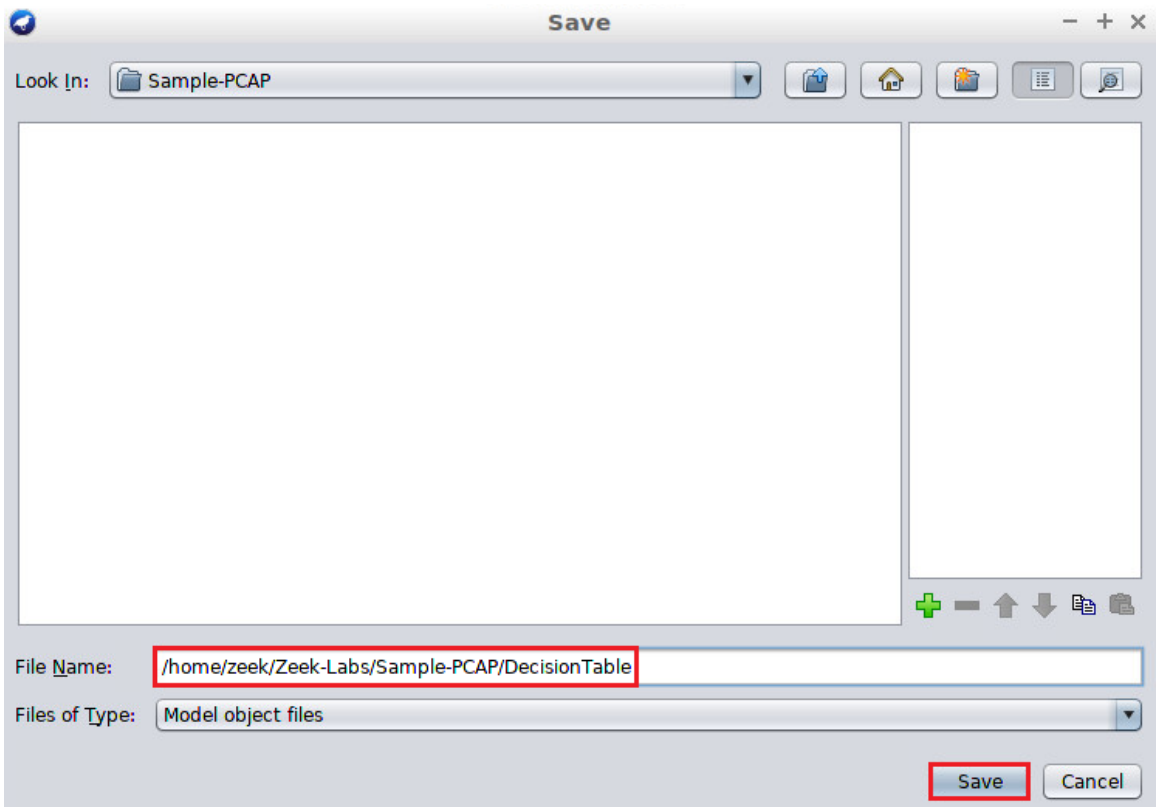
It is possible to save a trained classifier to be reused in future instances of testing and classification. This section will introduce how to save a trained classifier.

Step 1. Under the *Result list* section, right click on the *Decision Table* and select *Save model*.



Step 2. Navigate to the Lab workspace directory and save the *Decision Table*. Alternatively, use the GUI to navigate to the lab workspace directory to select the file. Use the *Save* button to save the new *DecisionTable* file into Weka.

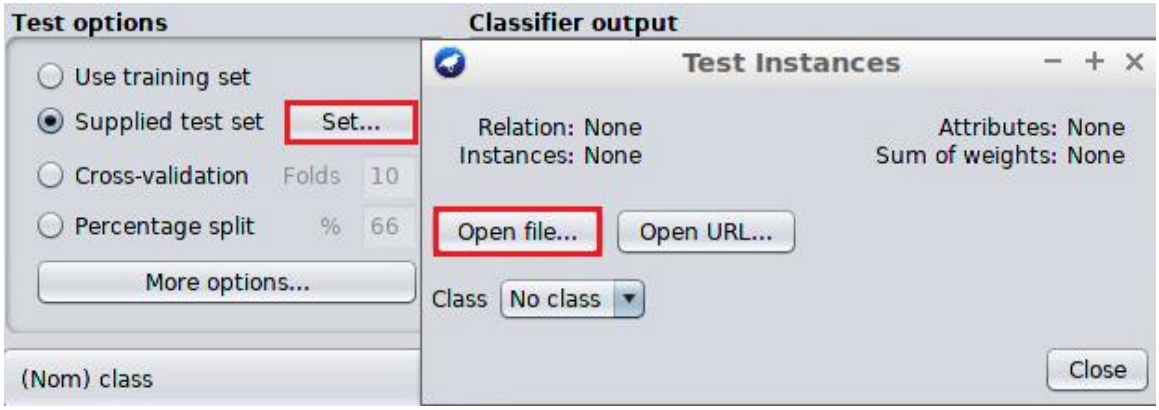
```
/home/zeek/Zeek-Labs/Sample-PCAP/DecisionTable
```



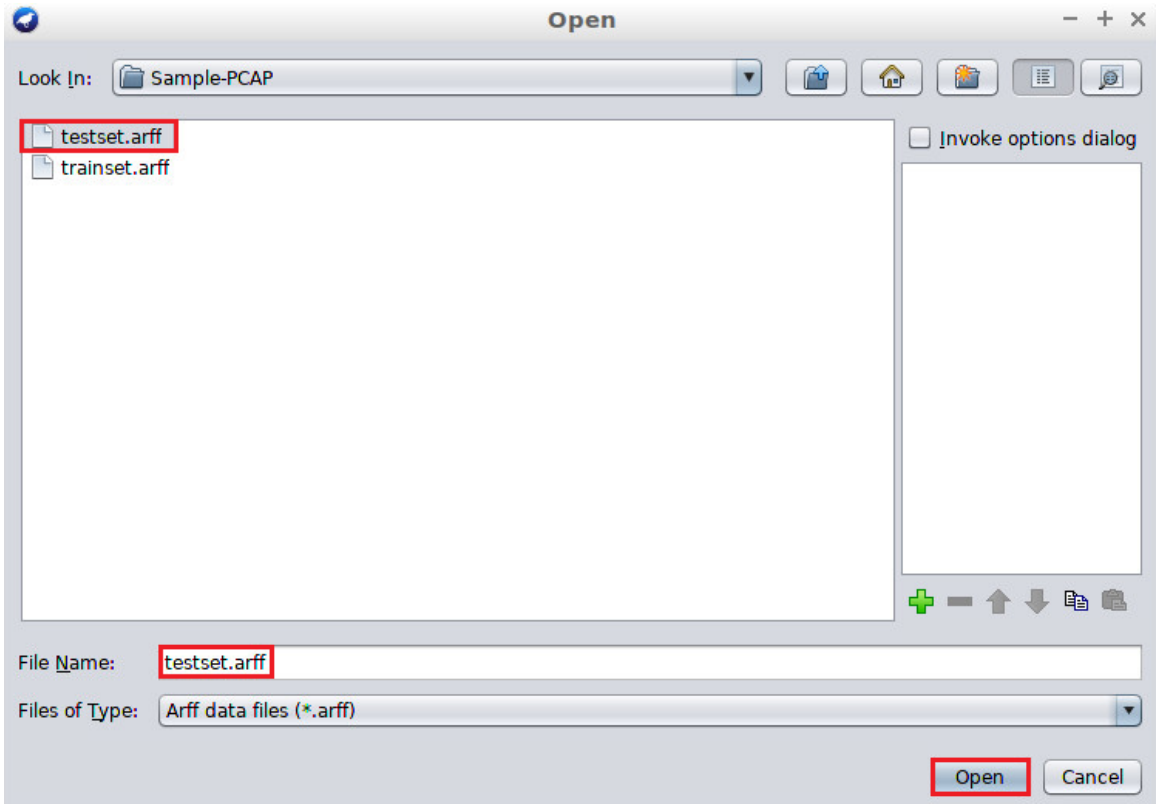
Once saved, we can proceed to testing the classifier's accuracy on predicting labels for the test dataset.

3.2 Using the classifier to predict labels for the test dataset

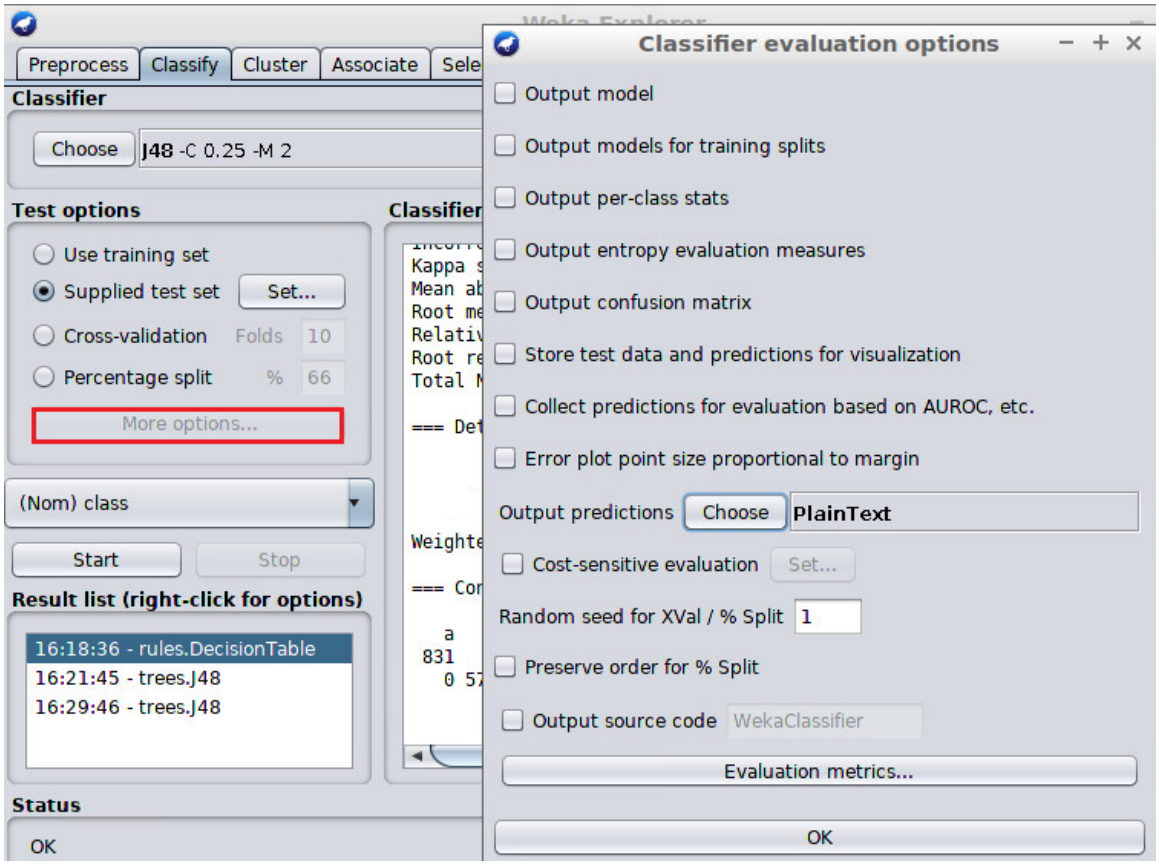
Step 1. Under the Test options section, select the Supplied test set then, select *Set* button to load the select the *Set* button to load the test dataset. Within the *Test Instances* window, click the *Open file* button.



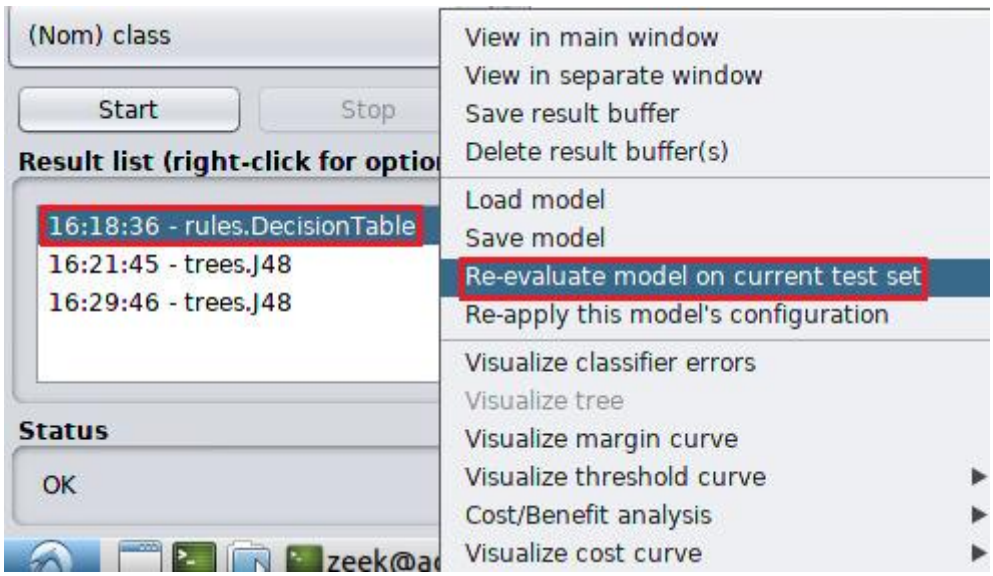
Step 2. Select the *testset.arff* file and click the *Open* button to load the test dataset then, click the Close button.



Step 3. Under the *Test options* section, select the *More options* button to configure the classifier to match the following image then, click on *OK*.

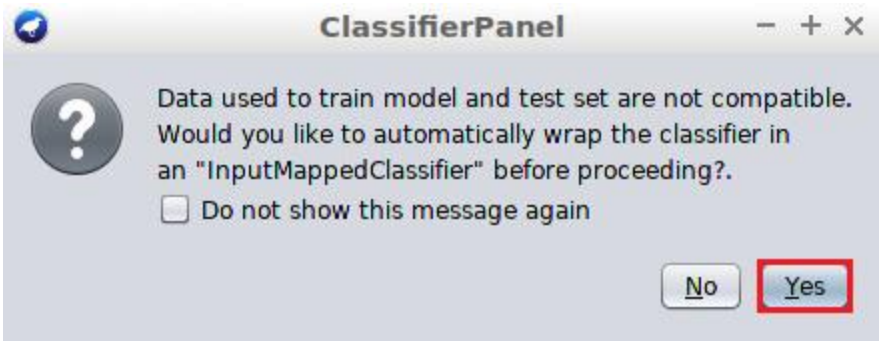


Step 4. Under the *Result list* section, right click on the *Decision Table* and select *Re-evaluate model on current test set*.



Step 5. After filtering the *sourceip* and *destip* features into Nominal attributes, the *testset.arff* file will not be properly formatted. Weka will need to update the *testset.arff*

dataset to be used by the classifier. Select the *Yes* button on the *ClassifierPanel* pop-up panel.

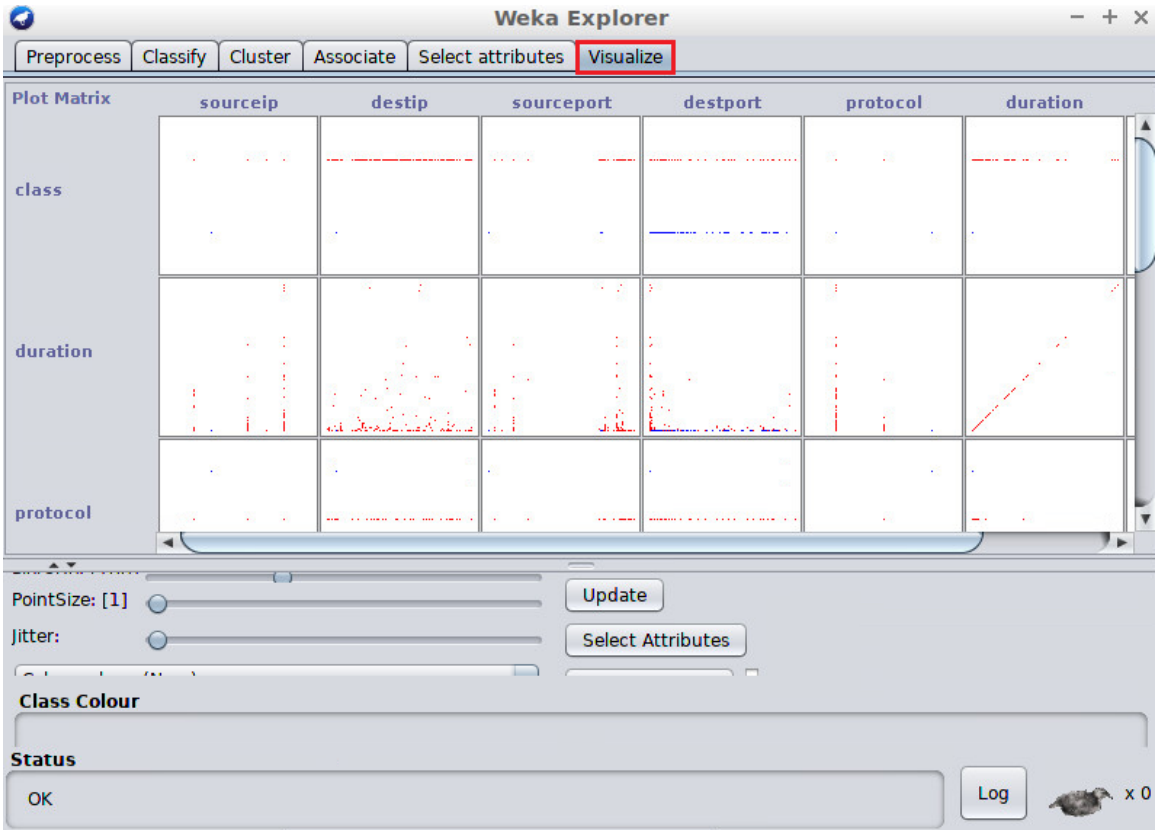


The classifier will generate new predictions, which can be viewed by saving the resulting *.arff* file.

3.3 Viewing the predicted labels for the testdataset

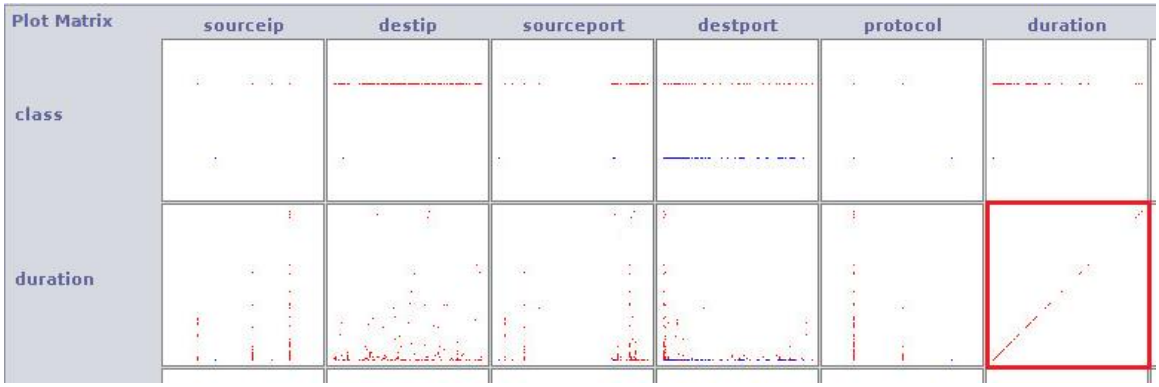
To save the resulting *.arff* file,

Step 1. Within the *Explorer* panel, click the *Visualize* tab located at the top of the *Explorer* panel to switch to the *Visualize* panel .

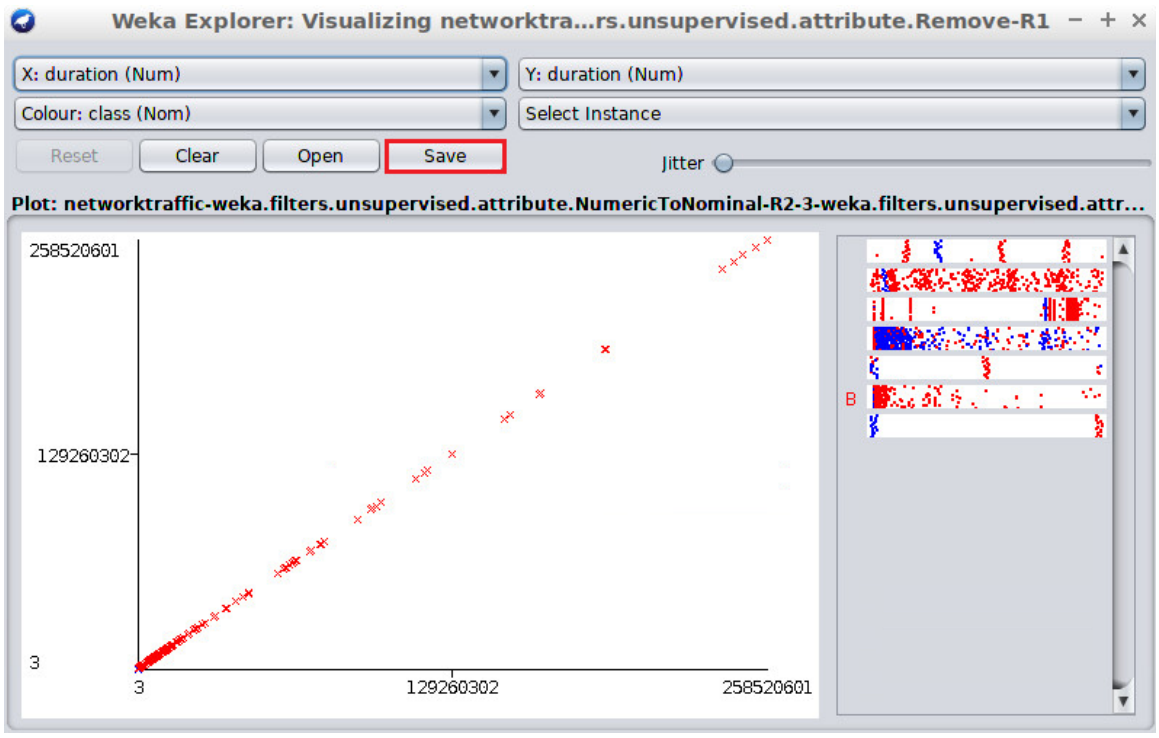


Displayed will be resulting graphs from attribute correlations solved by the machine learning classifier.

Step 2. Select the *duration x duration* graph, found in the sixth column (*duration)* and second row (*duration)*.

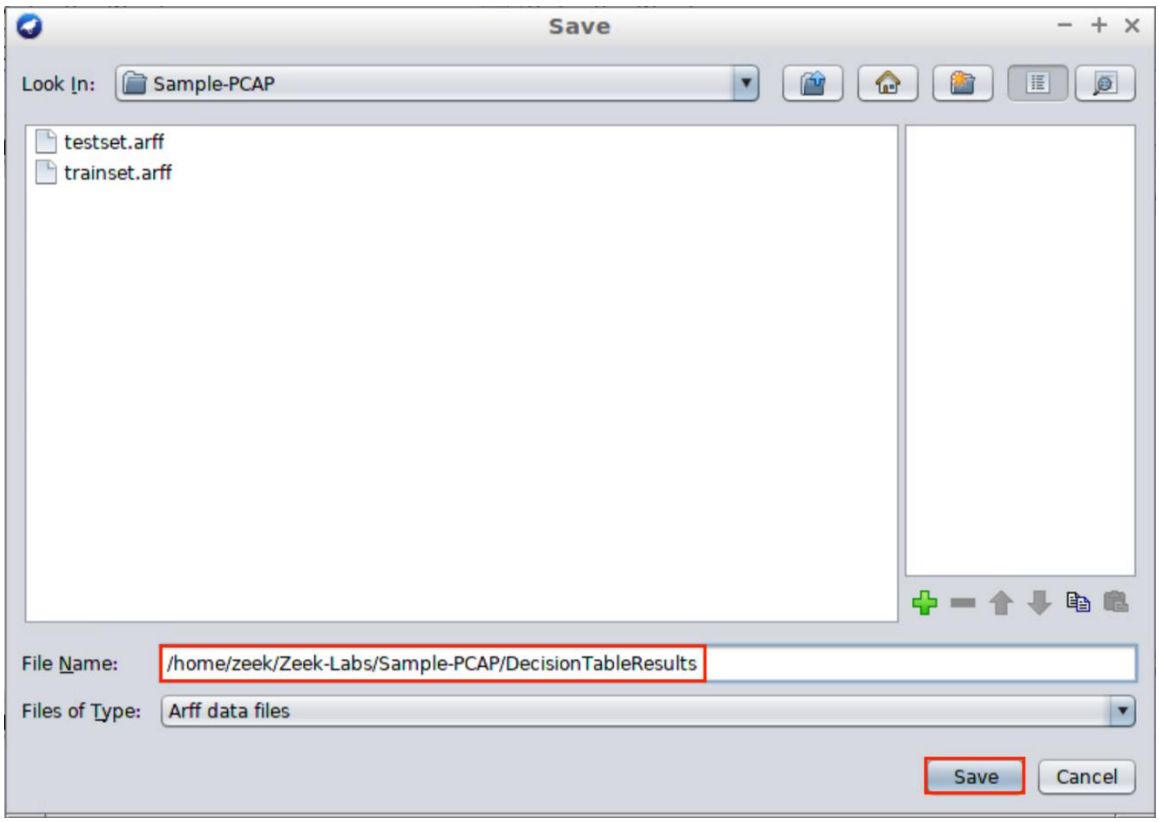


Step 3. Click the *Save* button.



Step 4. Navigate to the Lab workspace directory and save the *DecisionTableResults*. Alternatively, use the GUI to navigate to the lab workspace directory to select the file. Use the *Save* button to save the new *DecisionTableResults* file into Weka.

```
/home/zeek/Zeek-Labs/Sample-PCAP/DecisionTableResults
```



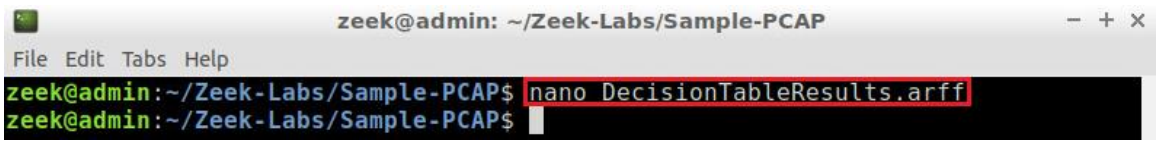
Step 5. Close all of the Weka tabs with the orange *x* on the top right corner of each panel.

Step 6. Return to the Terminal and navigate to the lab workspace directory.

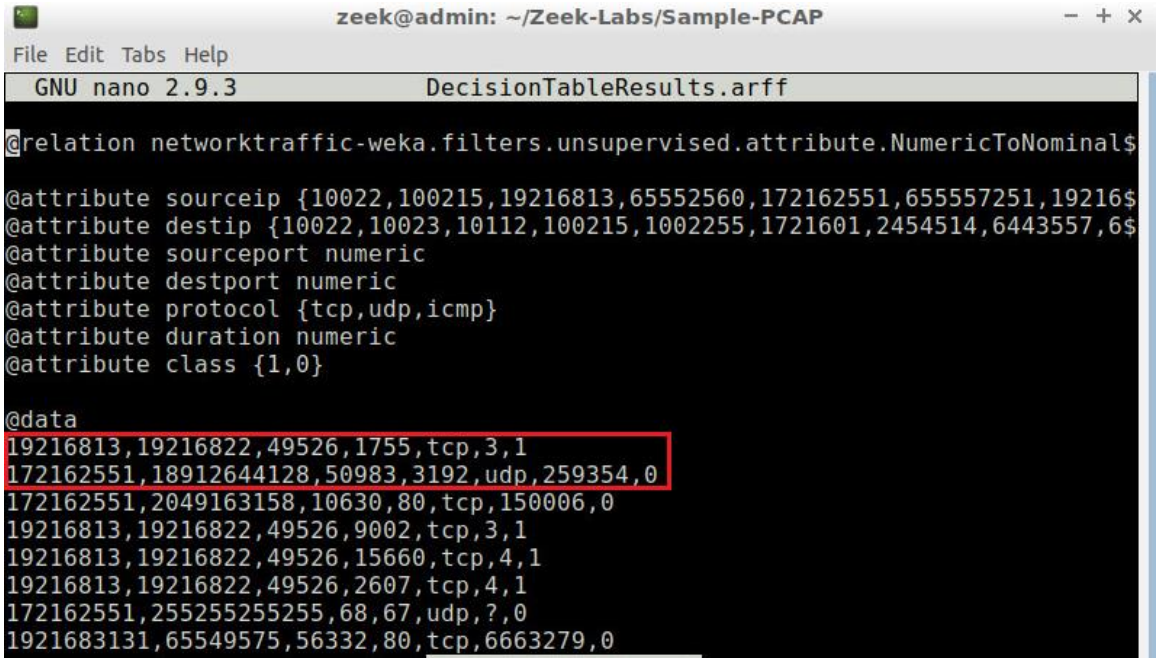
```
cd ~/Zeek-Labs/Sample-PCAP/
```

Step 7. Using a text editor, view the *DecisionTableResults.arff* file.

```
nano DecisionTableResults.arff
```



The file will be opened, and each data object will contain a new classification label.



Traffic found in the first row of data was labeled with a 1, as malicious traffic. Traffic found in the second row of data was labeled with a 0, as benign traffic.

Concluding this lab, we have introduced the capabilities of implementing a machine learning classifier to detect specific anomalies or events. Multiple classifiers can be used for training network security classifiers, and the features within each training dataset can have a profound impact on the classifier's accuracy. By removing, modifying or adding new features you can test the accuracy of a classifier. In this lab, we generated a *Decision Table* that was capable of labeling malicious and benign traffic.

References

1. "Attribute-relation file format", The university of waikato, [Online], Available: https://www.cs.waikato.ac.nz/~ml/weka/arff.html