ECEN758 – Assignment 6

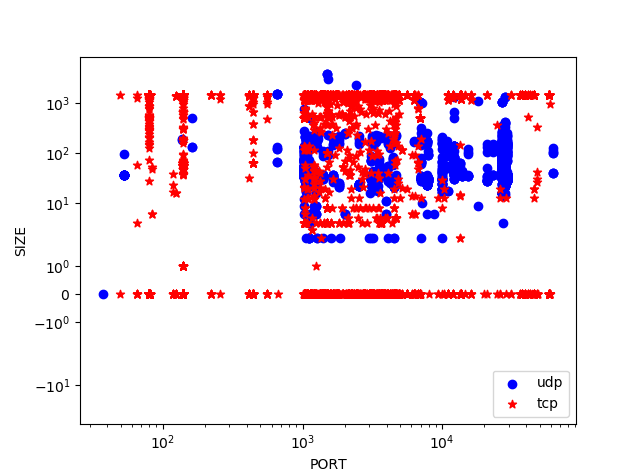
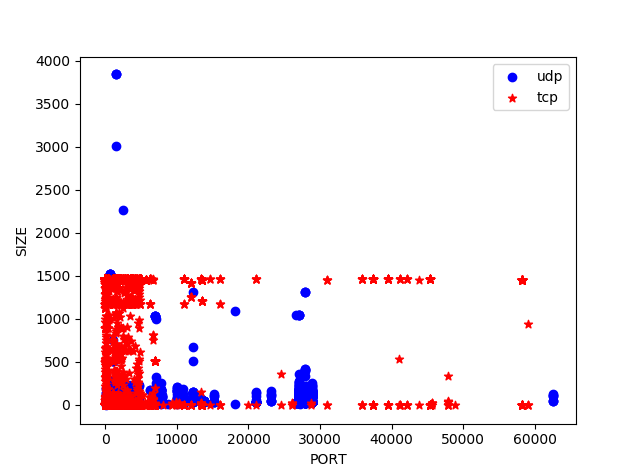
Rachel Hsingtze Yeh

hsingtze1007@tamu.edu

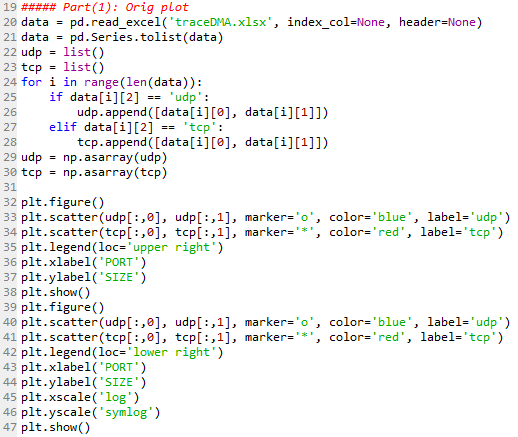
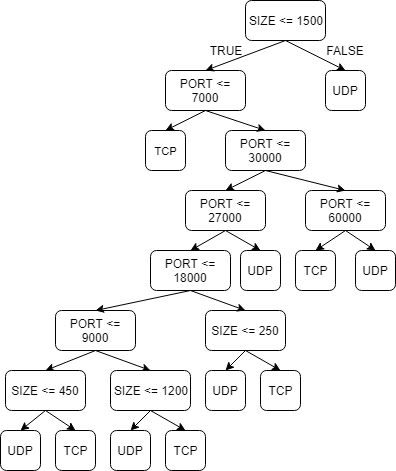
**Decision Trees**

1. **Scatter plot of PORT vs. SIZE.**

The scatter plot of Port vs Size is showing in two figures below with linear axis scaling and log axis scaling respectively. The red star is class of TCP and the blue dot is the class of UDP.



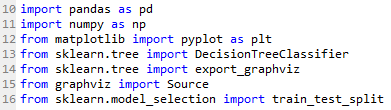
From the fig1 with linear scale, I can split points and build my decision tree using 10 split points as:



1. **Select a package to use for decision tree analysis.**

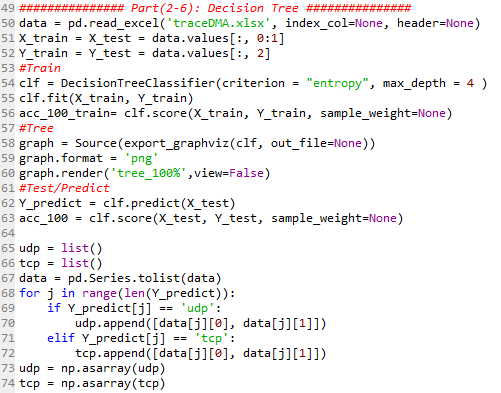
I’m using python 3.6 to compute my tree. The classifier, prediction accuracy, and graphical representation can be generated as:

* **Decision Tree**: can be imported as “*from sklearn.tree import DecisionTreeClassifier”*, and use with various parameters *criterion* as entropy, and *max\_depth* to adjust the number of split points.
* **Predict**: using the method under DecisionTreeClassifier as *predict(X)* where X is the check input.
* **Prediction Accuracy**: can be calculated by the method under DecisionTreeClassifier as *score(X, y, sample\_weight=None)* where X is the test samples and y is the true labels for X.
* **Graphical Representation**: using the *export\_graphciz* to export the graph with input parameters of our own decision tree.

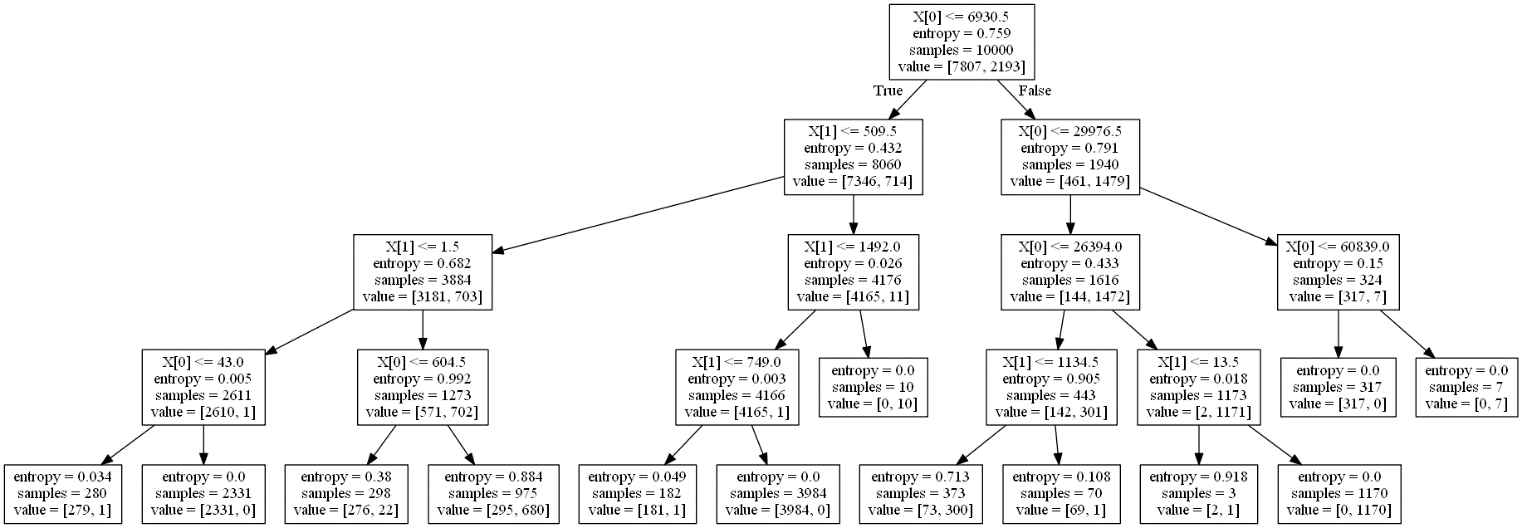


1. **Create a decision tree from predicting Class from PORT and SIZE.**

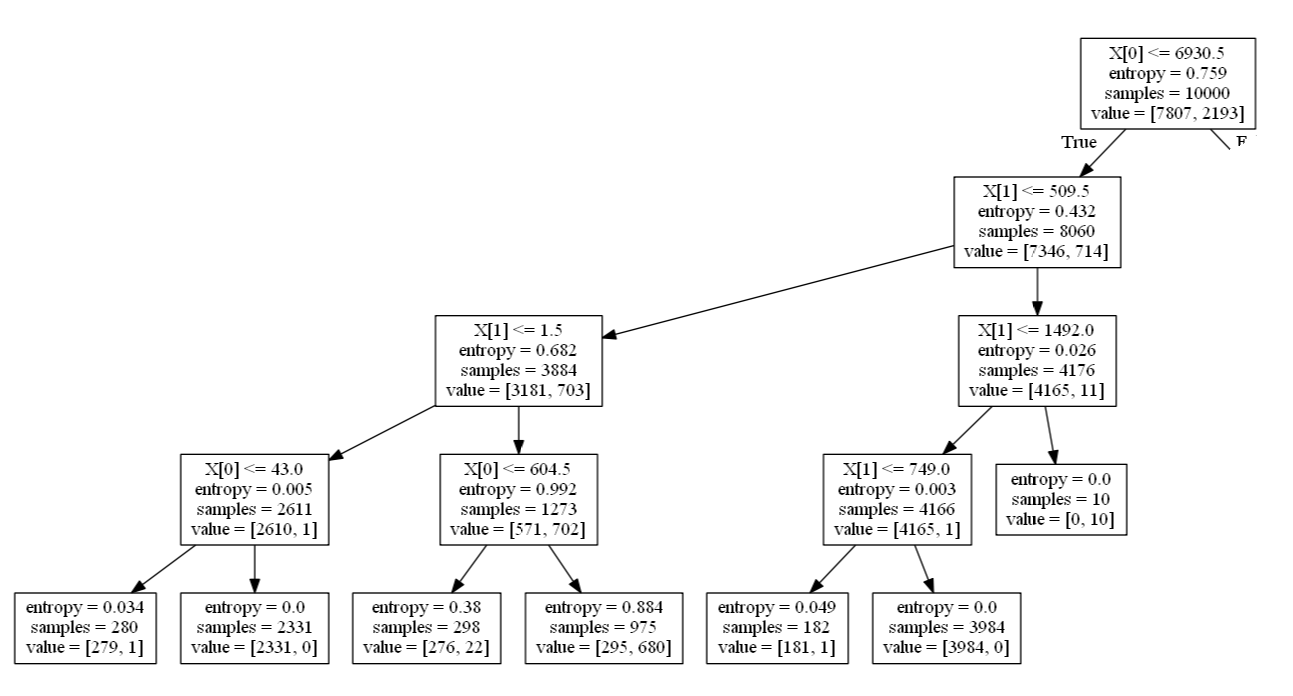
* Using the above functions in python, putting the whole data into the decision tree, the tree can be computed and generated as code below.
* As originally, I created the decision tree without any limitation where the maximum depth is no limit, and the decision tree will be large and looks like overfitting. Then I change the max\_depth = 4 where it gives me about 12 splitting points and looks much more reasonable for me to compare.
* The code is as below.



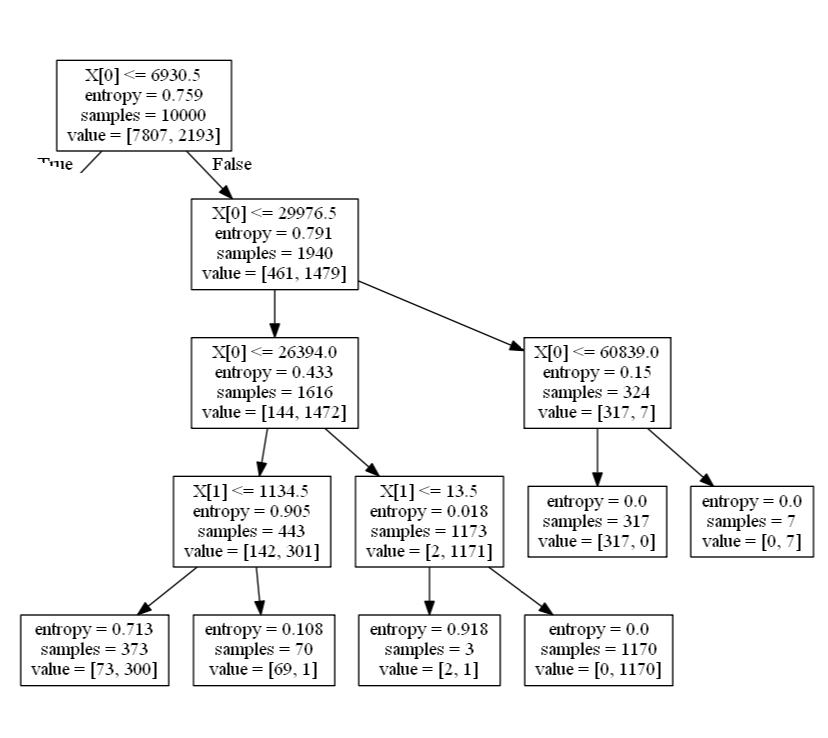
1. **Create a graph of the decision tree.**



***Figure: The Decision Tree graph where takes whole data and using entropy to compute with depth of 4.***



***Figure: Detail tree graph with left hand side.***



***Figure: Detail tree graph with right hand side.***

1. **Comment the similarities or differences between this decision tree and your choices.**

Compare with my section 1 and the computational decision tree, the table below shows the difference and can be compare as:

* **Number of points:** my tree has only 10 points but the computational one has 12 points.
* **Depth of the tree:** my tree has depth of 6 but the computational has only 4 layers.
* **Splitting points:** my splitting points is using whole values where I take approximate and round the value; the computational has more accurate value up to some decimal values where it depends on the distribution.
* **Small point values:** my tree doesn’t have splitting point in the area of high distribution (e.g. Port < 7000) since I cannot determine the which class has more points based on large data; but the computational tree can have splitting point in this area (e.g. Port < 43) which is more accuracy.

Conclude above, the computational tree can have much more accuracy since it uses less depth of tree to get more points, and it can find the better splitting point in the high distribution area for large data and the values for the points can be accurate as decimal point based on the distribution.

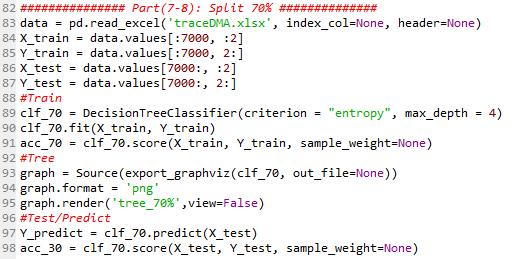
|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Section 1 | Decision Tree |
|  | Port | 7000 | 6930.5 |
| Right | Port | 30000 | 29976.5 |
|  | Port | 27000 | 26394.0 |
|  | Port | 60000 | 60839.0 |
|  | Size | 1200 | 1134.5 |
|  | Size |  | 13.5 |
| Left | Size | 450 | 509.5 |
|  | Size |  | 1.5 |
|  | Size | 1500 | 1492.0 |
|  | Port |  | 43 |
|  | Port |  | 604.5 |
|  | Size |  | 749 |
| \* | Port | 18000 |  |
|  | Port | 9000 |  |
|  | Size | 250 |  |

1. **Compute the prediction accuracy for the full dataset.**

* The prediction accuracy is **96.06%** for both training and testing with depth = 4.
* The prediction accuracy is **99.97%** for both training and testing with no limitation of depth.
* The code can be found above in section 3.

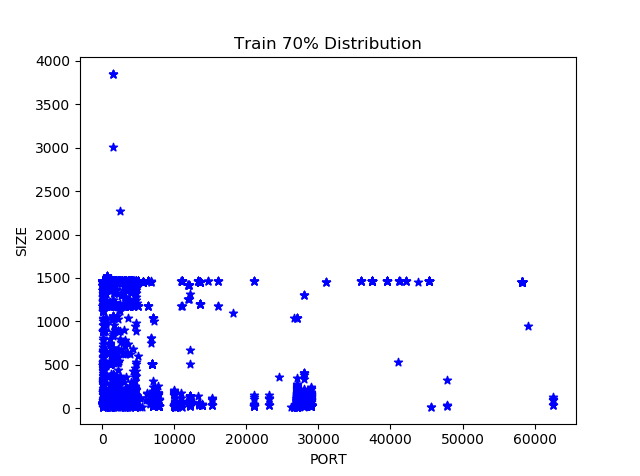
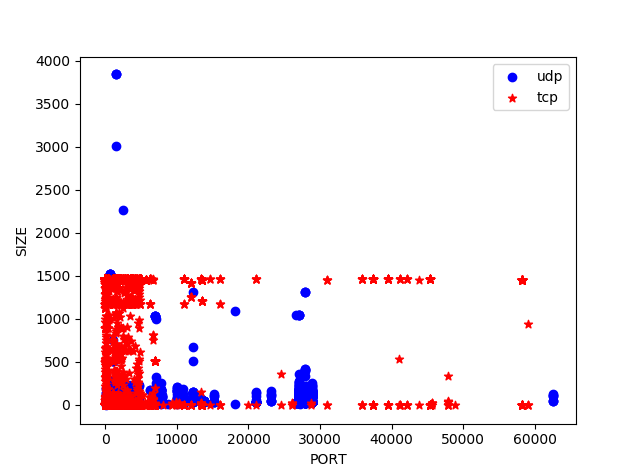
1. **Split the data into two parts: first 7000 with training and last 3000 for prediction. Compute the prediction accuracy.**

* The training accuracy is **97.64%** but the prediction accuracy is **40.40%** with depth = 4.
* The training accuracy is **99.97%** but the prediction accuracy is **63.47%** with no limitation of depth.
* The code is as below.



1. **Comment on two values of prediction accuracy.**

* The testing accuracy is quite low compare with the training accuracy and the full-data accuracy (from section 6). And the reason is because the separating of the data, where the first 7000 data distribution is much more difference than last 3000 data.
* From the figures below, left side is the whole data and right is the distribution of train data with first 70%, and we can see that our train 70% missing lots of TCP class at the bottom distributed between port of 10000 – 50000, and this caused the low accuracy of our data tree because the improper choosing the training data set.
* So if we choose the train\_test\_split method in our computational it will randomly choose the 70% of our whole data based on the distribution, which will bring us the better accuracy.



1. **Determine a better split of the data (70% training. 30% prediction). Compute accuracy.**

* The training accuracy is **96.11%** and the prediction accuracy is **95.9%** with depth = 4; and the training accuracy is **99.98%** and prediction accuracy is **99.2%** with no limitation of depth.
* I change the selecting method for training and testing data, use the train\_test\_split method where it can randomly choose the training data more evenly based on the distribution.
* From below figure, we can clearly see that it chooses the training data based on the distribution and the bottom TCP points are also be chosen.
* The code can be found below.

