NUS CS-3235: Computer Security November 14, 2021

Assignment 3 Report

Lecturer: Reza Shokri *Student: Wang Chenyu A0245095B*

1 Overview

Although the traffic data are encrypted, we can still find some "patterns" of the traffic data and then it is able for us to "distinguish" then. By "summarising" the traffic data and then using machine learning and apply "Classification Algorithm", urls can be classified properly.

2 Implementation Mechanism

In general, we need to figure out two important things:

1. What classification algorithms shall we use?
2. How can we "abstract summary from each url file?"

The first question gets multiple answers: Logistic Regression, k-Nearest Neighbors, Decision Trees, Support Vector Machine, Naive Bayes... However, in order to find the best one, we need to know what the algorithm needs to perform:

1. Many features as input.
2. Multiple (35) label classes as output.

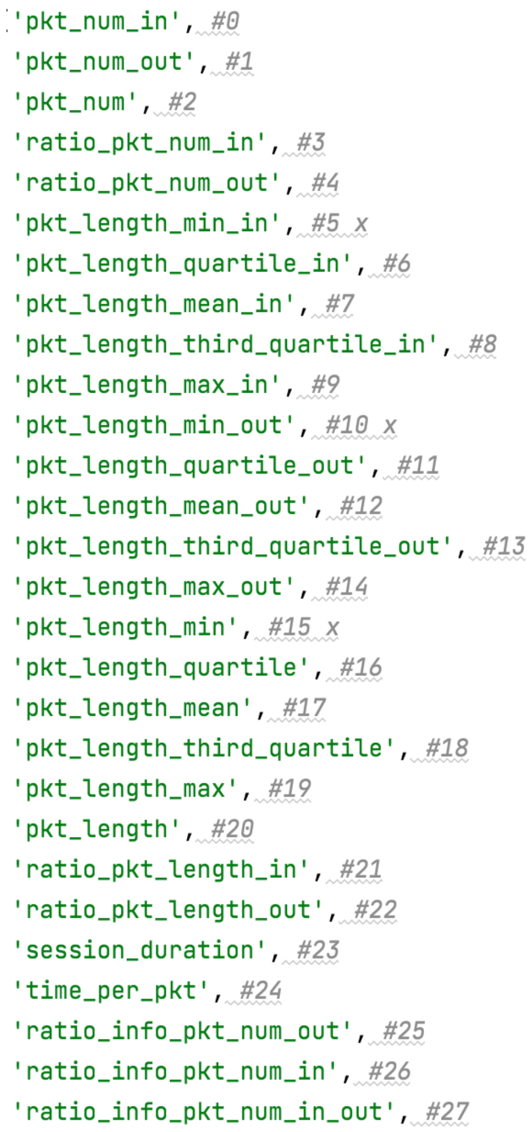
So our remaining choices are SVM, KNN and Naive Bayes because they can deal with classifications with multiple output labels well. Then after reading some papers about network analysis, I planned to try KNN and SVM. It turned out that

KNN(90%) works better with SVM(40%), I think the reason is that: if you have k different features, then SVM will try to set up hyper-planes as "boundaries" to divide the data and when you put a point inside to predict, the result is which "Section" the point lands to. It is indeed very good in some cases, however because the inputs are all 'network traffic' features, they are quite similar and do not have such a clear "boundary" to differ them, so SVM may not work well.

Different with SVM, KNN will let the k nearest neighbours of the test point to 'Vote' in order to determine the predict value, in that case we no need a "clear boundary", if most of the neighbours are URL 5, then you are URL 5. Therefore it gets a better performance here because "The unknown url no need to be different from other urls (have a clear boundry), but only need to have many urls from the same website around it".

Then it comes with the "data processing" part. At first I think that as training set, we only get 8 profiles * 35 url files, if we treat each of the url file as one single training data entry, it is not enough. Then my first approach is to divide each profile into several chunks, then by summarising each chunk we can get one training data entry. However, the performance comes out to be very bad. Then I realised that by doing so, I lost many "potential features", like the session's duration, number of packets during the session and so on. So I end up using each url file to generate only one training data entry, but I create a very detailed "summary" of it which have many many features: Instead of only mean, 25 quartile and 75 quartile can also reflect the characteristics well, and I focus on 3 statistic fields: pkt number statistic, pkt length statistic, time statistic. I also add 3 statistics for 0 byte packets because they can be regarded as 'Information packets', like ACK, which will also reflect the website's characteristic. I then created 27 features and by feeding the KNN classifier with it, I can obtain a 90%+ average accurate rate.

3 Appendix



```
'pkt_num_in', #0
'pkt_num_out', #1
'pkt_num', #2
'ratio_pkt_num_in', #3
'ratio_pkt_num_out', #4
'pkt_length_min_in', #5 x
'pkt_length_quartile_in', #6
'pkt_length_mean_in', #7
'pkt_length_third_quartile_in', #8
'pkt_length_max_in', #9
'pkt_length_min_out', #10 x
'pkt_length_quartile_out', #11
'pkt_length_mean_out', #12
'pkt_length_third_quartile_out', #13
'pkt_length_max_out', #14
'pkt_length_min', #15 x
'pkt_length_quartile', #16
'pkt_length_mean', #17
'pkt_length_third_quartile', #18
'pkt_length_max', #19
'pkt_length', #20
'ratio_pkt_length_in', #21
'ratio_pkt_length_out', #22
'session_duration', #23
'time_per_pkt', #24
'ratio_info_pkt_num_out', #25
'ratio_info_pkt_num_in', #26
'ratio_info_pkt_num_in_out', #27
```

Figure 1: Features of network traffic data that are being used to train

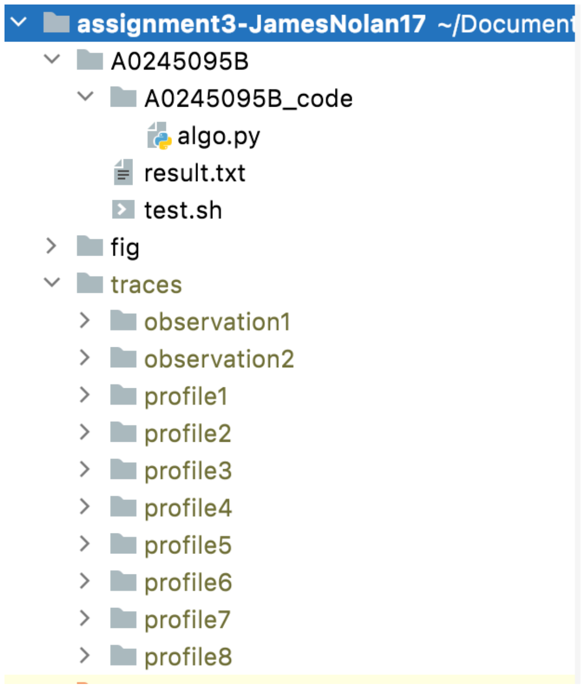
Execution example:

1. put the observation1 and observation2 folder into traces folder.

2. cd to directory A0245095B.

3. run ./test.sh ../traces/observation1 ../traces/observation2 (It is okay to change '../traces/observationX' to absolute path)

4. get result.txt in A0245095B folder.



```
assignment3-JamesNolan17 ~/Document
  A0245095B
    A0245095B_code
      algo.py
    result.txt
    test.sh
  fig
  traces
    observation1
    observation2
    profile1
    profile2
    profile3
    profile4
    profile5
    profile6
    profile7
    profile8
```

Figure 2: Code Structure

Sitation:

Iterative-Tuning Support Vector Machine For Network Traffic Classification

Machine Learning for Traffic Analysis: A Review