CS4035 Cyber Data Analytics: Assignment 3
Group 9

Georgios Dimitropoulos: 4727657
Emmanouil Manousogiannis: 4727517

June 22, 2018

Sampling Task

For this task we used the unidirectional netflows from scenario 5 of the CTU-13 **Dataset** (Malware capture 46 [1]).We selected one infected IP as host and calculated the 10 most frequent IPs this host interacted with.The subset of the dataset with the host interactions consisted of 2129 rows and the resulted IPs with their corresponding frequencies can be seen in the table below.
Our next step was to try to find an approximation of the most frequent IPs that interact with the host by implementing *min-wise sampling* algorithm.To do this, we initially kept a subset of the whole dataset where the examined host was present. Then we assigned every row of this dataset with a randonly generated number between 0 and 1. The dataset was then sorted based on this value in descending order. The only step we had to do then, was to select an integer value k, where $k < length(dataset)$ and select only the first k values of the dataset as a sample. We demonstrate the results for indicative values of k below.

IP	Occuracies	Frequency
212.117.171.138	588	0.27
46.4.36.120	550	0.26
147.32.80.9	92	0.043
64.4.2.109	48	0.022
205.188.186.137	42	0.019
74.3.164.224	36	0.016
65.55.72.7	34	0.0159
74.3.164.223	33	0.0155
65.55.40.23	32	0.015
209.85.148.147	30	0.014

Table 1: Most Frequent IP

k:500, IP	Occuracies	Frequency
212.117.171.138	126	0.252
46.4.36.120	124	0.248
147.32.80.9	19	0.038
64.4.2.109	14	0.028
65.55.40.23	13	0.026
65.55.72.7	11	0.022
74.3.164.224	8	0.016
74.3.164.223	8	0.016
64.4.56.23	7	0.014
74.3.164.222	7	0.014

Table 2: min-wise sampling results with k=500

As can be seen from the results of the tables above, if we consider a significant percentage of our dataset as sample,meaning that we have a large enough k, we can approximate the most frequent IPs accurately.The frequencies obtained with the min-wise sampling method are quite close to the real ones, while the 10 most frequent ip's are the same in 8 out of 10 cases.In our implementation, you can find the results of min-wise algorithm for several values of k. From the results of those experiments, we realized that the larger the k is, the fewer the approximation error becomes, however it reaches a limit at some point near 1/3 of the whole dataset. In addition, while experimenting with other scenarios of the CTU-13 , we concluded that the results and the approximation accuracy is also dependent on the size and type of the dataframe we are testing.

Sketching Task

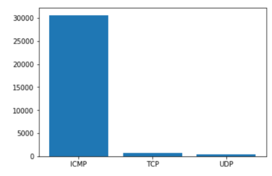
In this task we implemented the Count-Min sketch algorithm [2] through using the unidirectional Netflows from scenario 5 of the CTU-13 dataset (Malware capture 46). We chose one infected IP as host and calculated the 10 most frequent IPs this host interacted with. We investigated a lot of different values for the parameters of height and width for the Count-Min sketch matrix. The parameters w and d can be chosen by setting $w = \lceil \frac{e}{\epsilon} \rceil$ and $d = \lceil \frac{ln1}{\delta} \rceil$, where the error in answering a query is within an additive factor of ϵ with probability $1 - \delta$ and e is Euler's number. Thus, we implemented this algorithm with different values of ϵ and δ which lead to different values of height and width. Hence, we tried to cross-validate our results and to tune our parameters in order to estimate the true frequency with a high probability and to draw useful conclusions in comparison with the Min Wise sampling strategy.
Our results for the various values of width and height parameters can be depicted in Table 3. Based on these results, we can depict that if we have a high value for the width parameter, then the estimated frequency is identical to the true frequency no matter what the value of the height parameter is. Thus our implementation achieves a zero error. Furthermore, we can see that in case that the width parameter becomes smaller and the height parameter becomes large enough, then we have a small error rate. More specifically, we can see that the ordering of the IPs remains almost the same and some estimated frequencies have a few minor differences in comparison to the true frequencies. However, in case that the width parameter and the height parameter become smaller in the same time, then the performance of our algorithm derogates and the majority of the addresses are not even possible to be found.
As far as the comparison between the Count-Min sketch algorithm and the Min Wise sampling strategy is concerned, we could say that in general, thorough a careful tuning of their parameters we can achieve similar results with both methods. However, if we optimize the parameters in the Count-Min sketch algorithm we can have a better estimation in comparison to the Min Wise sampling strategy. The main reason behind this, is that the if we increase the width parameter, then the theoretical bound guarantees that the error will be very low and thus the Count-Min sketch algorithm is more accurate.
Furthemore, taking into account the structure of the Count-Min sketch algorithm and the fact that uses hash functions, we could say that the Count-Min sketch algorithm has very cheap space requirements in terms of memory and it is faster in comparison to the Min Wise sampling strategy.

IP Address	**True Frequency**	**Estimated Frequency Height=9 ,Width=27183**	**Estimated Frequency Height=2 ,Width=27183**	**Estimated Frequency Height=9 ,Width=27**	**Estimated Frequency Height=2,Width=27**
212.117.171.138	588	588	588	604	604
46.4.36.120	550	550	550	562	572
147.32.80.9	92	92	92	99	99
64.4.2.109	48	48	48	62	0
205.188.186.137	42	42	42	47	0
74.3.164.224	36	36	36	56	0
65.55.72.7	34	34	34	41	0
74.3.164.223	33	33	33	0	0
65.55.40.23	32	32	32	41	0
209.85.148.147	30	30	30	49	82

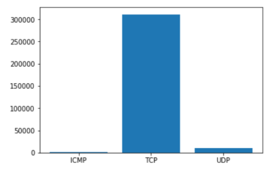
Table 3: Count-Min sketch results for various values of width and height

Botnet Flow Data Discretization Task

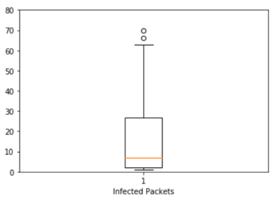
In this task we considered scenario 10 from the CTU-13 datasets as requested.After loading the data, we removed all flows labeled as "Background".The resulted dataset consisted of many different features per flow, such as Potocol,Duration,Packet, Bytes etc.Since we had to select two of them and apply discretization we had to investigate which two of them would be discriminative enough to model the behavior of an infected host.After some experiments we concluded in **Protocol and Packet** features as they presented a quite indicative behavior for modeling infected hosts. This can be seen in the plots provided below,where one of the infected host features are plotted compared to other normal hosts. As we can observe, our infected host mostly uses ICMP protocol instead of the common TCP in normal cases.Similarly, the infected packets have a quite larger variance (without taking into account the outliers) and higher mean value, than the corresponding normal ones.



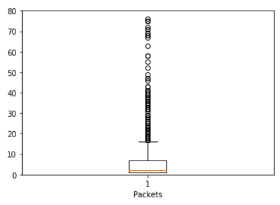
Infected Protocol



Normal Protocol

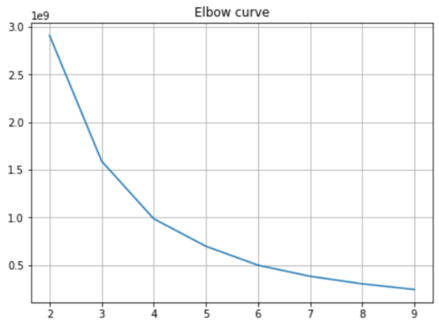


InfectedPackets



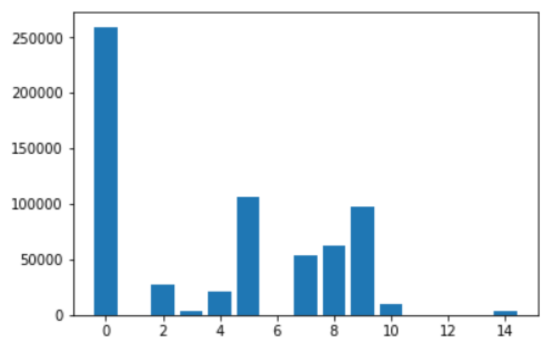
Normal Packets

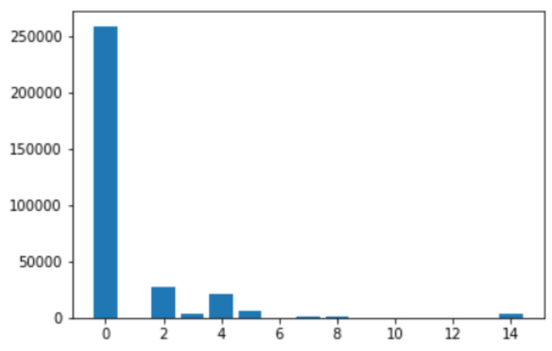
For the discretization of those two features we were based on the method followed in [5].The protocol feature, which only takes 3 values was easy to handle as it is discrete by nature. In order to discretize 'Packet' feature however, we had to create clusters. In order to determine the number of clusters, we used ELBOW algorithm. The results indicated, as can be seen in the plot below, that the optimal number of clusters is between 4 and 6. We selected 5 so that we are on the safe side.



ELBOW algorithm cluster indication

Now that we determined the number of clusters that we need to use, we sorted our dataframe in ascending order and followed the steps to set the cluster limits as in [5].Since we have 5 clusters we have to determine the 20th , 40th ,60th and 80th percentile. If one value is below the 20th percentile,it will be assigned to cluster 0, if it is between 20th and 40th to cluster 1 etc.The ordinal rank of each percentile is computed with the following formula: $r(p) = [\frac{p}{100} XN]$, where p is the percentile (i.e 20,40 etc) and N is the collection size.
After completing this procedure, every flow of our data will be assigned to one cluster based on its "Packet" value and one cluster based on its "Protocol" value.The next thing we had to do, was to combine those two values of each flow in one single discrete value.To do that, we followed the steps of the mapping algorithm as explained in the cited paper.
After completing the discretization task, we could observe that the two features we selected to discretize provided us with a quite interesting indication in terms of finding infected hosts, when they are combined.The plot below, shows the final discrete value representing each flow,in case of dealing with infected hosts and in case of normal hosts.


Normal scores


Infected Scores

Botnet Profiling Task

In this section we selected an N-Gram probabilistic model and tried to learn a model for one of the infected hosts of the dataset, during a specific time window. Then we tried to match this profile with the other hosts of our data in order to detect those that could be infected.Our method was based on [3].

Initially we converted our "StartTime" feature to DateTime object so that we could apply sliding windows. Then,after some trials, we selected a sliding window of 90 milliseconds and tried to observe one of the infected hosts. We collected all the flows related to that host inside this time window and kept a record of the final discrete values that represented each of the flows (state list), as calculated in the previous question.

The next step, was to compute all possible N-grams that were created from the state list we mentioned above and their corresponding frequencies. We used different values of N, but we concluded that the most suitable was N=2, as it could perform well in smaller state lists too.So after calculating all the created bigrams,we selected the 10 most frequent of them to represent the 'profile' of the examined infected host. We followed exactly the same procedure for one normal host,meaning it was not infected and created its profile.

Having created two profiles, one for an infected host and one for a normal host, we now had to test how those models could match with other unknown host models. To do this, we created an ngram model for every single remaining host of our dataset and then tried to see which of the known profiles, the infected or the normal one, matched better with it. The matching measure was performed as in the cited research paper [3]

If the examined host matched better with the infected host model,it was classified as infected. Otherwise it was classified as normal.The results of this method were quite impressive as can be seen below.Please note that the infected host we used for modelling was "147.32.84.208" and the normal one was "147.32.84.170"

TP	FP	TN	FN	precision	recall
9	1	4	0	0.9	1

Bonus: Flow Classification Task

Data Exploration

In this task we constructed a classifier for detecting anomalous behavior in individual Netflows. Our Netflow dataset contains background, benign and botnet packets. Initially, in order to implement this task we used scenario 5 of the CTU-13 dataset (Malware capture 46). As a first step we performed a data cleaning. Thus, we treat this problem as a binary classification problem between benign and botnet cases and for this reason we ignored all the background Netflows as they do not provide any meaningful information to our classifier. After that we visualized the distribution of the instances in each of the two aforementioned classes in order to have an intuition and a better understanding of our data as shown in Figure 1. It can easily be observed from this figure that the distribution is skewed and imbalanced. Furthermore, we observed that there are both categorical and numerical variables and the one-hot encoding was adopted for the categorical variables.

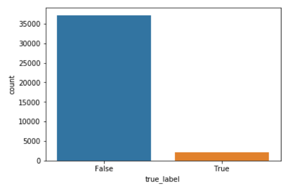


Figure 1: Distribution of the instances in each class. True represents "botnet" and False represents "benign"

Building of the Classifier

For this classification task, firstly we used the data cleaning of the previous step and after that we selected the same features as in the Botnet Profiling Task in order to make a fair comparison between the two methods. Thus, we made a feature selection and we chose the numerical feature "Packet" and the categorical feature "Protocol" in which the one-hot encoding was adopted. It is worth mentioning here that we also experimented with other features such as the "Duration", and the "Bytes". However, in the evaluation there were only minor differences and we conclude that the later features were not so discriminative in comparison to the former ones. Moreover, downsampling of the majority class (Benign) was applied in order to deal with the imbalanced issue. Finally, we used an ensemble classifier, namely the Random Forest [6],[7],[8] for our implementation and our results were evaluated through cross validation.

Evaluation of the classifier

In this question we evaluated our model in packet level as in [4] and in the host level as in [5].

Packet Level

The results of our evaluation metrics for packet level where we follow the procedure in [5] can be depicted in Table 4. We can see here that our classifier achieves a very good performance in terms of all metrics and is able to detect the majority of TP while producing only a few FP.

TP	FP	TN	FN	Accuracy	Precision	Recall	F-Measure
277	78	7341	159	96.77%	66.51%	78.03%	69.54%

Table 4: Packet level evaluation

Host Level

For host level evaluation, after studying [4] we experimented with two different approaches. In the first one we detect whether all Netflows from a host have been detected as benign or botnet. In case that one record of a host is botnet, then we say that the host is detected as a botnet. In the second approach, we used a threshold and we classify a host as a botnet in case that multiple records (cumulative sum is larger than the threshold) of a host are botnets. The results of our evaluation metrics for host level for the two approaches can be depicted in Tables 5 and 6. We can see here that in the first approach we achieve a very good performance in terms of recall (identify all the TP) but in the other metrics the performance is not good at all. While, our second approach achieves a sound performance in terms of the recall and accuracy as it is able to detect all the TP while producing almost no FP. Finally we could say that for detecting botnets we would prefer using a sequential model instead of a classifier based on our metrics.

TP	FP	TN	FN	Accuracy	Precision	Recall	F-Measure
1	68	84	0	55.56%	1.45%	100%	2.86%

Table 5: Host level evaluation: Approach 1

TP	FP	TN	FN	Accuracy	Precision	Recall	F-Measure
1	4	148	0	97.39%	20%	100%	33.3%

Table 6: Host level evaluation: Approach 2

References

[1] *https://www.stratosphereips.org/datasets-ctu13/*

[2] Graham Cormode , S. Muthukrishnan. *An improved data stream summary: the count-min sketch and its applications, Journal of Algorithms, v.55 n.1, p.58-75, April 2005.*

[3] Abou-Assaleh T, Cercone N, Keselj V, et al. *Detection of New Malicious Code Using N-grams Signatures.2004: 193-196.*

[4] Garcia, Sebastian, et al. *"An empirical comparison of botnet detection methods." computers and security 45 (2014): 100-123.*

[5] Pellegrino, Gaetano, et al. *"Learning Behavioral Fingerprints From Netflows Using Timed Automata.*

[6] L. Breiman. *Random forest, Machine Learning 45,2001, 5–32.*

[7] Siddhartha Bhattacharyya , Sanjeev Jha , Kurian Tharakunnel and J. Christopher Westland. *Data mining for credit card fraud:A comparative study. In Decision Support Systems. v.50 n.3, p.602-613, February, 2011*

[8] C. Whitrow, D.J. Hand, P. Juszczak, D. Westona and N.M. Adams. *Transaction aggregation as a strategy for credit card fraud detection. In Data Mining and Knowledge Discovery 18 (1) (2009) 30–55*