Elsa Rodriguez Plaza
GT CS 6035: Introduction to Information Security

Project 2.5:

Malware Analysis and Machine Learning

Summer 2020

Task 1 – Cluster and Classify

I was able to accomplish both goals in this task by using the **malheir** configuration that I uploaded to Canvas. In terms of goal 1, I got a score of 72.8 in the testing phase and in terms of goal 2, I was able to classify all malware using my trained model. Here's the output:

```
debian@debian:~/Desktop/avml$ malheur -c config.mlw -o classify.txt -vv classify subjects/;
cat classify.txt
generic = {
	input_format	= "text";
	event_delim	= ";";
	state_dir	= "./malheur_state";
	output_file	= "classify.txt";
};

features = {
	ngram_len	= 1;
	vect_embed	= "cnt";
	mist_level	= 1;
	hash_seed1	= 514084890;
	hash_seed2	= 1978915395;
};

prototypes = {
	max_dist	= 0.60000;
	max_num		= 0;
};

classify = {
	max_dist	= 1.00000;
};

cluster = {
	link_mode	= "average";
	min_dist	= 0.20000;
	reject_num	= 0;
```

```
       shared_ngrams   = 0.00000;
};

Extracting features from 'subjects/'.
  [##################################################] 100.0%  total 00m 00s
  Done. 5 feature vectors using 0.01Mb extracted.
Loading feature vectors from './malheur_state/prototypes.zfa'.
Classifying feature vectors to 176 prototypes.
  [##################################################] 100.0%  total 00m 00s
  Done. Classified 5 feature vectors to 176 prototypes.
Saving 0 feature vectors to './malheur_state/rejected.zfa'.
Exporting classification to 'classify.txt'.
# MALHEUR (0.6.0) - Automatic Analysis of Malware Behavior
# Copyright (c) 2009-2015 Konrad Rieck (konrad@mlsec.org)
# University of Goettingen, Berlin Institute of Technology
# ---
# Classification for subjects/
# Precision of classification: 100.0 %
# Recall of classification: 40.0 %
# F-measure of classification: 57.1 %
# ---
# <report> <label> <prototype> <distance>
malware4 C000-0126 a99231deb1cf944a2a3a3d435338569bbc7702f3a25c484416502a990ebb7801.ramnit
0.341956
malware2 C000-0057 2f494a201de170a3e215c48b3b1cc4b74546df23cdf3e7510d30e85a7678589e.speedbit
0.840913
malware3 C000-0011 5395f965d9414044279f33ef0a8211e28a14784dcd67de7149dae7dd98e693ef.mydoom
0.589434
malware1 C000-0134 303012889f814f18c23be0b49cc48df3b1fa2bac591be5ab43f14eee5885f088.jeefo
0.367176
malware5 C000-0126 a99231deb1cf944a2a3a3d435338569bbc7702f3a25c484416502a990ebb7801.ramnit
0.687392
```

Task 2: Explain your results

Answer the following questions about malheur and machine learning in general:

1. Explain the meaning of precision, recall, and f-score in the context of the model you have just created using malheur.
    a. In the case of the model that I trained for this task, the **precision** expresses the accuracy of the malware classes that I obtained in each of the phases (training, testing, classification). The **recall** reflects the classes's distribution over the different clusters of classified malware that I passed into the training and testing

phases. Finally, the **f-score** represents the overall performance of the model in terms of the **precision** and the **recall** and how well classified the malwares were. [1]

2. What are 'prototypes' with regards to malheur? Your answer must include an explanation about what malheur uses them for, what is the strategy malheur uses to find them, and their relationship to malheur clustering and classification.
    a. The prototypes are elements in the selected report of program behaviors. As detailed in [2], a small subset of program behaviors is chosen, based on how representative the subset is compared to the complete dataset. The prototypes are identified by trying to minimize the distance between the reports and its nearest prototype. [2]. During the dataset's clustering, the clustering is initially determined by the prototypes and then those prototypes contained in the selected clusters are stored as internal state and subsequently used during classification and incremental analysis.
3. What is the strategy malheur uses to calculate distances between samples and why it is important to have API call names (our features) in the order calls were actually made in the malware execution? What would happen if this order is lost?
    a. Distances are computed by using a matrix that represents the reports' proximity. [2] The order in which the calls were made matters, because certain malwares exhibit the same call API call order that would cause a harmful action, this allows better identify a a program is a malware that exhibits a similar behavior than those contained in the training set. If the order had been lost, then that would have caused decreased **precision**, **recall**, and **f-score** during the training, testing and classification phases.
4. Explain your results in Task 1 by providing the following:
   Write a short paragraph explaining the meaning of the following malheur configuration parameter, how you have achieved your proposed value for it, and how that value relates to your results:

i. features = { ngram_len = XXX;}: I set this parameter to 1, because not all malware classified exhibit a sequential behavior. [1]

ii. features = {vect_embed = "XXX";} I set this configuration to “cnt”. Since, each malware has a number of features (API calls), it is suitable to quantify the number of features each of those have and assign that number to each dimension. A binary quantification would not have given me an accurate representation of each of the dimensions and it would cause all 5 malwares to be rejected during the classification.

iii. prototypes = { max_dist = XXX;} . I set this number to 0.6. I tried setting this configuration to a value different than 0, because I didn't want all reports to be considered prototypes [2]. I also didn't want the distance to be too high, because the malwares analyzed exhibited similar behaviors, since the prototypes were somewhat close.

iv. prototypes = { max_num = XXX;} I set this value to 0, because I didn't want to limit the number of prototypes. According to [2], this parameter is set to reduce prototype identification computational overhead, but in my case I was analyzing a small dataset.

v. classify = {max_dist = XXX;} I left this parameter unchanged (value of 1), since I wanted to make sure that my model would classify all reports within distance of 1 to cluster and the rest of the reports rejected. The higher this number is the poorest the classification would be, because it would include more reports that are far in distance. In my case, all reports and prototypes were close in distance. [2]

vi. cluster = {link_mode = “XXX”;} I set this configuration to average to make sure that distance between clusters would

be the average of the distance between members across different clusters. [2][3]

vii. cluster = {min_dist = XXX;} I set this parameter to 0.2, because the cluster needed to be close to each other, but not to the point that all clusters became a single cluster.

viii. cluster = {reject_num = XXX;} I set this to 0, because I didn't want any clusters to be rejected. I figured that in my case due to the length of the data and the distance that I set in the other parameters, I would have gotten many clusters rejected if I had not set this value to 0.

Works Cited

1. Rieck, k ,Trinius, P ,Willems, C , and Thorsten, H, "Automatic Analysis of Malware Behavior using Machine Learning". April, 2011. Accessed: July 22, 2020. [Online] Available: http://www.mlsec.org/malheur/docs/malheur-jcs.pdf
2. "Malheur. Automatic Analysis of Malware Behavior". April, 2011. Accessed: July 22, 2020. [Online] Available: http://www.mlsec.org/malheur/manual.html
3. "Average Linkage". March, 2015. Accessed: July 22, 2020. [Online] Available: https://www.ebi.ac.uk/training/online/glossary/average-linkage#:~:text=Average%20Linkage%20is%20a%20type,member%20of%20the%20other%20cluster.