Malware Classification

Zeyuan Xu

1 Introduction

Malware, short for malicious software, includes a class of hostile or intrusive software. The taxonomy of malware classes have been versatile and broad, and antivirus software have been successfully detecting the malware for decades. The traditional ways to detect malware are either signature based or behavior based. Signature based detection compares newly scanned software with the stored patterns of known malware, called "signature" of malware. If high similarity score is found, a malware sample is discovered. Since the method captures the software sample, it is also called static method. Behavior based method, also called 'anomaly based' method, tries to capture the anomalies in the runtime environment of the software, if abnormal behavior is observed, such as malicious network traffic, the software would be quarantined. The method is also called dynamic method. In a real world scenario, dynamic methods generate lots of false positives, and static methods are generally preferred for antivirus. However, the traditional methods are not enough for the newly emerged classes of malware.

The newly emerged malware are called "polymorphic" and "metamorphic" malware, able to reprogram themselves in different execution environments. The names indicate that traditional static analysis would not work, since the new malware no longer has a fixed signature attached to it. The nature of the problem fits to solve as a classification problem, since the predictive nature of machine learning algorithms can effectively classify polymorphic malware based on certain attributes observed in training set. Therefore, the motivation for the project is to use machine learning to classify known classes of polymorphic malware.

2 Data Preprocessing

The data was taken from security blogger Marco Ramilli's website, where he parsed a number of known malware samples into JSON files, with each JSON file representing a single malware piece. The original malware samples were encoded in the format specified in MIST format, which encodes the hex dumped assembly code of malware into more compact strings (Tirnius, et al, 2009).

Data preprocessing include to process all the JSON files included, and to construct pandas dataframe from all the JSON files, with columns for attributes of each malware sample. Every JSON file has a specific malware name attached to it, so a column of label is added to the dataframe, which was later changed from categorical scale to numeric scale. Since the malware names are too specific, certain labels are grouped together. For example, "Trojan-Destover-Sony-1201172" and "Trojan.Dropper.Gen-1201172" are all labled as "Trojan" and later as class label 3.

3 Feature Selection

The initial dataframe gives 137 features, and there are lots of NaN values in the dataframe. Therefore, feature selection is required, and certain statistical measures are applied. The first metric used is the number of non-NaNs for a feature: Among the 137 features, 9 features have more than 2000 non-NaN values, which makes them more proper candidates for training set, since more training data implies better classification result.

The second metric used is the count for each distinct and unique values for a single attribute. If the distribution of a feature is too sparse, it is omitted, since this indicates higher entropy and less information used to classify. This metric results in 3 final candidates for attributes: `pe_sec_character, pe_sec_entropy, pe_sec_name`. The final 3 attributes and the label were in the end merged into a single pandas dataframe for classification task.

4 Model Selection

Since the class is prelabeled, a supervised learning algorithm is a plausible candidate for model. Since the problem is essentially multi-class, the candidate models chosen are Logistic Regression (LR), Random Forest (RF), and Multi-layer Perceptron (MLP), all have the capability to do multiclass classification.

4.1 Model Assumptions

Logistic Regression does not require a linear relationship between variables, and the errors need not be normally distributed. Since little can be assumed about the training set, it is a good benchmark for testing. However, there are still some assumptions made: first, logistic regression requires observations to be independent of each other. Second, independent variables should not be too higly correlated, and finally, it assumes linearity of log odds. These assumptions are fine for the dataset, since malware samples are not from a single source, but from different hosts on different Operating systems.

Random Forest: since random forest model makes best split and performs best when the samples are independent, which is the case here. Other than that, random forest makes no assumption about the distribution of data, so it is a great classifier for the task at hand.

Multilayer Perceptron, or Feedforward Neural Network: since neural network is a universal function fitting model, no assumption is needed to made here about the dataset. Therefore, neural network is a good model to test on the dataset.

4.2 Feasibility and Comparison

The assumptions are reasonable from the three different models, and logistic regression has the most assumptions. Since all the models are capable of doing multiclass classification tasks, solving the model is simple. In the end, each attribute is set as the training data and the label as the target value, and the performance of models on each attributes should be compared to see which models give the best performance.

Other possible choices include Support Vector Machine, K-nearest neighbor, Decision tree. Unsupervised learning algorithms such as K-means, Gaussian Mixture, and DBSCAN can be considered as well. However, since the dataset is prelabeled, and supervised learning algorithm generally performs better, they are omitted. Decision tree is a rudimentary version of Random Forest, so it is not considered. SVM could be a plausible candidate, but multiclass SVM is a concept I have not fully grasped. So in the end, the models used are LR, RF, and MLP.

5 Algorithm

5.1 Algorithm for data preprocessing

5.2 Algorithm for classifiers

6 Result

For feature 1 `pe_sec_character`, the summary for model performance, using 0.33 train-test split on the dataset, is:

Name	pr(MLP)	rc(MLP)	f1(MLP)	pr(RF)	rc(RF)	f1(RF)	pr(LR)	rc(LR)	f1(LR)
Crypto	0.85	0.97	0.90	0.84	1.00	0.91	0.84	0.99	0.91
Locker	0.65	0.36	0.46	0.97	0.20	0.33	0.61	0.20	0.30
Ransomware	0	0	0	1.00	0.17	0.29	0	0	0
Trojan	0	0	0	1.00	0.67	0.80	0	0	0
VolatileCedar	0	0	0	1.00	0.62	0.76	0	0	0
WMIGhost	0	0	0	0	0	0	0	0	0

The attribute performs great for Crypto and bad for Locker, MLP and LR did bad on teh other features, mainly because the existence of NAs in the dataset. But Random Forest gives better performance compared to other two classifiers.

For feature 2, `pe_sec_entropy`, the summary for model performance is shown in the table:

Name	pr(MLP)	rc(MLP)	f1(MLP)	pr(RF)	rc(RF)	f1(RF)	pr(LR)	rc(LR)	f1(LR)
Crypto	1.00	0.87	0.93	1.00	1.00	1.00	1.00	0.99	0.99
Locker	0.54	0.99	0.70	0.99	0.99	0.99	0.78	0.99	0.88
Ransomware	0	0	0	1.00	0.67	0.80	0	0	0
Trojan	0	0	0	1.00	1.00	1.00	0	0	0
VolatileCedar	0	0	0	1.00	1.00	1.00	0	0	0
WMIGhost	0	0	0	0	0	0	0	0	0

The problem remains the same for classes other than Crypto and Locker, due to the lack of sample numbers in the training sets. In the Random forest example, overfitting is observed due to the same issue.

For feature 3, `pe_sec_name`, the summary is shown in the table:

Name	pr(MLP)	rc(MLP)	f1(MLP)	pr(RF)	rc(RF)	f1(RF)	pr(LR)	rc(LR)	f1(LR)
Crypto	0.84	0.98	0.90	0.84	1.00	0.91	0.84	0.99	0.91
Locker	0.55	0.20	0.29	0.97	0.20	0.33	0.61	0.20	0.30
Ransomware	0	0	0	1.00	0.17	0.29	0	0	0
Trojan	0	0	0	1.00	0.67	0.80	0	0	0
VolatileCedar	0	0	0	1.00	0.62	0.76	0	0	0
WMIGhost	0	0	0	0	0	0	0	0	0

The results are similar to the previous, with Random Forest overfitting the other categories, which lack sample size, and WMIGhost, which only has 9 samples in the entire attributes, have zero score across all combinations.

7 Furtherwork

The further work is to collect more malware samples for better training results. The Microsoft Big2015 dataset is worth a try given enough computing power. The dataset has around 1TB data of hex dumped malware samples, and a team has come up with a way to turn the problem into image classification problem (Kaggle 2015), which could be both fun and productive to classify with a Convolutional Neural Network.

8 reference

1. Philip Tirnius, Carsten Willems, et al. *A Malware Instruction Set for Behavior-Based Analysis*, 2009

2. Kaggle, *Microsoft Malware Winner's Interview: 1st place, "NO to overfitting!"*, 2015 http://blog.kaggle.com/2015/05/26/microsoft-malware-winners-interview-1st-place-no-to-overfitting

3. Marco Ramilli, *Malware Training Sets: A machine learning dataset for everyone*, 2016 https://marcoramilli.blogspot.com/2016/12/malware-training-sets-machine-learning.html