Anomaly Detection Challenges - Challenge IV

Hamza Tahir (03670002) and Muhammad Hamza Usmani (03669506)

Technical University of Munich

1 Introduction

This brief report serves as a purpose to present and explain the methodologies applied to tackle the fourth challenge in the Practical: Anomaly Detection Challenges. Section 2 discusses the challenge task and the data set for the machine learning/anomaly detection task. Section 3 explains the approaches adopted for the task, and Section 4 summarizes the results.

2 The Challenge

Machine Learning Task The machine learning task for this challenge is to determine if a PEinfo sample is benign or malicious.

Table 1. Decision Classes

Class	Representation
Benign	0
Malicious	1

The Data set The dataset consists of PEinfo files. There are 9,743 records in the training set. The training dataset consists of 4,753 benign records and 5,000 malicious rows. The machine learning task for this challenge is to classify a record as benign or malicious.

3 Methodology

This section explains the data analysis and the machine learning process to build the model for classifying the given samples as attack or normal record.

3.1 Feature Extraction

Our approach was to first manually analyze around 40 samples of malicious and benign files to diffrentiate and identify relevant discriminatory features between them. The following features were extracted for the machine learning process:

1. Number of Exports
2. Number of Imports
3. DLLs
4. PE Sections
5. PE hash
6. Debug

Number of Exports: This feature represents count of the number of functions exported in a PEinfo sample.

Reasoning: In our manual analysis, we noted that malicious files had a lot less exports (usually even 0 exports) than benign files.

Number of Imports This feature represents count of the number of functions imported in a PEinfo sample.

Reasoning: In our manual analysis, we noted that malicious files had less imports than benign files.

DLLs This feature represents count of different DLLs in a PEinfo sample. Only DLLs of training samples are used for this feature. The feature is a sparse matrix, represented as a bag of words like representation. There are 10XX columns used to represent count (in case of presence), and 0 (in case of absence) of each DLL in a data point.

Reasoning: In our manual analysis, we noted that some malicious files had certain identifying DLL's like "COMCTL32.dll" which were not present in the benign files. Therefore, a bag of words representation seemed appropriate to capture this information.

PE Sections This feature represents data-point averages in case of presence, and 0 in case of absence of:

1. Size
2. Virtual Size
3. Entropy

within the PE section of a sample.

Reasoning: In our manual analysis, we noted that some malicious files had higher average values of these values than benign ones.

PE Hash This feature presence or absence of PE Hash section in a data point.

Reasoning: In our manual analysis, we noted that some malicious files had this section missing and most benign files had it present.

Debug This feature represents presence or absence of debug section in a data point.

Reasoning: In our manual analysis, we noted that some malicious files had this section missing and most benign files had it present.

The features extracted have the following class wise arrangement:

Table 2. Representation Accross Classes

Feature	Benign Samples	Malicious Samples
Total Samples	**4753**	**5000**
Number of Exports (avg)	38.51	2.79
Number of Imports (avg)	107.287	134.42
Size (avg)	67472.62	313666.43
Virtual Size (avg)	68098.42	169342.02
Entropy (avg)	3.756	4.36
PE Hash (avg)	4543	4967
Debug (count)	4754	588

3.2 Feature Scaling

The presented features have different scales, this required to normalize all features on one scale, so that the different ranges and scales of features do not contribute to relative weights of those features. To normalize, the following method was used:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

3.3 Re-sampling

The training data set is higher number of malicious data points in comparison to benign records, thus the data was re-sampled. The re-sampled data set had all (4753) benign records from the data set, and 4753 malicious records that were randomly chosen from the training data set.

3.4 Machine Learning Techniques

Following supervised and unsupervised machine learning techniques were used to classify given samples as "benign" (represented as 0) or "Malicious"(represented as 1):

1. Random Forest (Supervised, meta heuristic technique)
2. Extra Trees Classifier (Supervised)
3. Multilayer Perceptron (Supervised with logistic, relu & tanh kernels)
4. Support Vector Machines (Supervised with kernels sigmoid, linear & poly)
5. KNN (Supervised with K tuned between 2 and 100)
6. DBScan (Unsupervised with little tuning)
7. Adaboost (Supervised, meta heuristic technique)

The chosen techniques were tuned, however because random forests & extra trees classifiers yielded promising results, other classifiers were abandoned.

3.5 Optimizing - Developing a Robust Classifier

To build a robust classifier, that is relatively general and is not restricted only to the given training set following techniques were used:

Ten-fold Cross Validation: Ten-fold cross validation was used to overcome the problem of over-fitting, and to build a model to that will generalize to an independent dataset, Hawkins et. al. The ten-fold cross validation scores were used to optimize hyper parameters(see [1]).

Optimizing for Hyperparameters: Most of the chosen classification techniques require to tune and choose hyperparameters, because good results were yielded via random forests & extra trees classifiers, the authors chose to tune the these techniques. For optimizing all hyperparameter configurations were tested. The classifiers were evaluated on the ten fold cross validation scores and on overlapping and partition set based crossvalidation approaches.& .

4 Results

Results of the challenge are summarized in this section. We present results based upon training accuracies only. The results presented are achieved after feature scaling and random re-sampling and ten-fold cross validation. The optimal value of number of estimaters for random forests is found to be:
Results of the challenge are summarized in this section. We present results based upon training accuracies only. The results presented are achieved after feature scaling and random re-sampling and ten-fold cross validation. The optimal value of number of estimaters for random forests is found to be: The optimal value of number of estimaters for extra trees is found to be: Results of the challenge are

summarized in this section. We present results based upon training accuracies only. The results presented are achieved after feature scaling and random resampling and ten-fold cross validation. The optimal value of number of estimaters for random forests is found to be: The optimal value of number of estimaters for extra trees is found to be:

Table 3. Best Average Cross-Validation Accuracies

Technique	Cross Validation Accuracy	Testing Accuracy(Public Score)
Random Forest	91.347	97.978
Extra Trees	96.33	97.542
Multi Layer Perceptron(1 layer)	92.75	93.95
Multi Layer Perceptron(3 layers)	93.67	94.4
Multi Layer Perceptron(4 layers)	93.66	94.53
Adaboost	96.221	95.293
DBScan	73.224	Did Not Submit
KNN	87.284	82.798

References

1. Hawkins D. , Basak S. , and Denise M. Assessing Model Fit by Cross-Validation J. Chem. Inf. Comput. Sci., 2003, 43 (2), pp 579586 (2003)