Project Report : Detecting new Malware families with Natural Language Features and Concept Drift Measurement

Vinay Bhapkar
New York University
vvb231@nyu.edu

1 Abstract

Malware classification remains a long-standing problem in cybersecurity. Machine learning approaches for detecting and classifying samples have proven successful by using different techniques, like observing natural language features or visualizing malware as image representations. They remain, however, susceptible to the rapid capacity for malware to evolve in the wild and adapt to new detection methods, and the statistical properties of malware change in what is known as concept drift. In this project, we attempt to implement natural language-based classifiers that are able to measure this evolution of malware, and compare two different approaches for quantifying it. For this purpose, we employ a subset of a dataset called MALREC which consists of Weighted Term Frequency Inverse Document Frequency (TF-IDF) scores for natural language features observed during the malware sample's execution.

1.1 MALREC data

MALREC is a malware sandbox system created by Severi. G. et al, which uses PANDAS record and replay capabilities to collect whole-system traces of malware executions. With this system, they were able to compile a dataset of the same name comprised of 66,301 recordings in various representations. We will use a subset of this dataset, consisting of Weighted TF-IDF scores for natural language features recorded during the malware's execution, which evaluates the importance of a word in relation to a document collection.

1.2 Datasets available–description of data available with malrec dataset

The MALREC Weighted TF-IDF scores derived dataset consists of 66,301 JSON format documents,

Figure 1: Weighted TF-IDF samples structure.

in which each word observed during the sample's execution is assigned a TF-IDF score based on the relative importance to the document collection of a word present in the sample. The full set of samples contains approximately 460,000 such features. The full dataset is comprised of more than 1300 different families of malware, but many of these families are widely underrepresented, so we excluded these to avoid class-imbalance. We also subsampled all those families that had more than 1,000 samples. Specifically, we set aside all malware families that had fewer than 100 samples, and subsampled those exceeding 1,000 samples up to 1,000 samples per family. The resulting dataset is made up of 29,466 samples and different 75 families. We then proceeded to remove the 10 least represented families, sampled 70 percent of each class into a training set, 15 percent into a cross-validation set, and 15 percent into a test set. This test set was augmented with the 3 malware families we just removed, in order to introduce never before seen families with which to test our classifier's ability to measure the concept drift of these samples in relation to the 8 classes it was trained with.

2 Methods used

For the purpose of malware classification, we implemented two well known classifiers that work very well with natural language features. One of them is the Multinomial Naive-Bayes (NB) classifier with Term Frequency features, and the other is the Support Vector Machine (SVM) classifier with a linear ker-

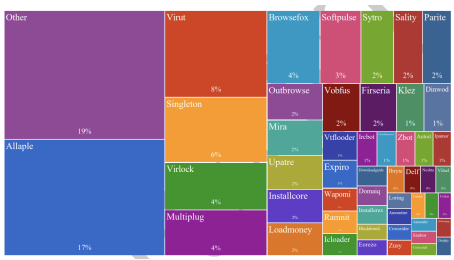

Figure 2: Weighted TF-IDF samples structure.

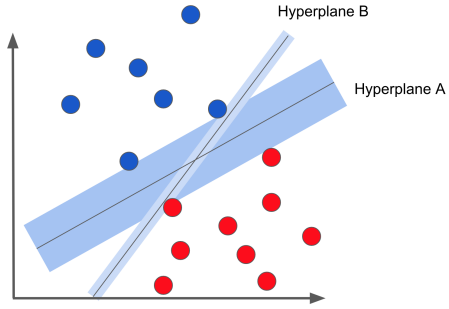

Figure 3: SVM example with multiple hyperplanes.

nel. For the NB classifier, we selected the top 5,000 features in the dataset using the Information Gain (IG) metric. For the SVM classifier, we employed two dimensionality-reduction approaches. The first one is once again utilizing the IG metric, whereas the second method we employed was performing Principal Component Analysis (PCA) using 1024 components. This results in three baseline classifiers: NB with top 5,000 IG features, SVM with top 5,000 IG features, and SVM with PCA-1024. For the purpose of measuring concept drift, we first define what concept drift is. When we train machine learning classifiers, over time, the statistical properties of the variables we are trying to predict change in unforeseen ways. This change is what we call concept drift. In the context of malware analysis, concept drift may be due to two phenomena. The first scenario is the evolution of malware within a family over time. The second scenario, is running into new malware families in the wild. This second scenario is the one we choose to focus on, as we believe our ability to properly classify malware is highly dependent on recognizing new malware families well before their presence becomes a problem for us. We employ two methods to try to detect concept drift. The first one consists on computing the geometric centroid of all of the training samples belonging to one class, and computing the distance of each sample to the centroid of their class. We use this measurement to define a threshold based on the statistical properties of the class (mean, variance, etc.). The second method, is only useful when we use SVM algorithms, and consists in computing for each class the distance of each of the training samples to their respective hyperplane. We use this measurement as well to define a threshold based on the statistical properties of the class (mean, variance, etc.). If we observe a new sample, and our classifier outputs a label for this sample, we compute the distance of this sample either to the class centroid, or to the hyperplane, and if it is outside of the boundaries of our defined threshold, we flag this sample so that we know that our classifier's prediction is untrustworthy. (i)SVM

(ii) Rocchio Classification

In machine learning, a nearest centroid classifier or nearest prototype classifier is a classification model that assigns to observations the label of the class of training samples whose mean (centroid) is closest to the observation.We have used this method to assign cluster of known family to test object and classify it as concept drift object if its Euclidian distance from centroid crosses certain threshold.

When applied to text classification using tf*idf vectors to represent documents, the nearest centroid classifier is known as the Rocchio classifier because of its similarity to the Rocchio algorithm for relevance feedback

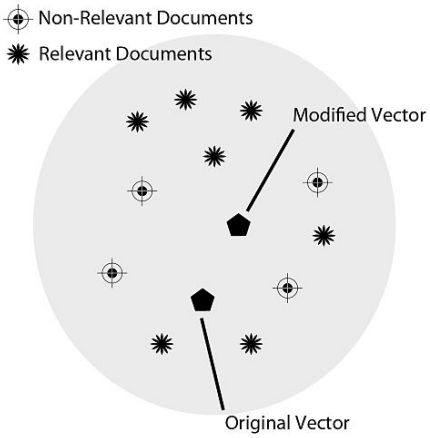

Figure 4: Nearest centroid or Rocchio classifier.

3 Concept Drift Detection

In order to detect concept drift and adapt our classifier to it, we will implement the following process:

1. Compute a distance measure (Nearest centroid or distance to hyperplane) for every training

sample belonging to a specific class

2. Set a drift threshold per class from the statistical properties of the class members in cross-validation fashion (mean, mean one standard deviation, etc)

3. For each new test sample:

   - Obtain classifier prediction
   - Compute distance measure
   - If the sample is outside of the threshold, flag the sample and save in a bucket

4. Relabel only the flagged samples with the appropriate label, and re-train the classifier

In the case of the nearest centroid distance measure, being outside of the threshold means being farther away from the centroid than the samples belonging to that class. In the case of hyperplane distance, being outside of the threshold means being closer to the hyperplane than the samples belonging to that class.

4 Results for Methods tested

We have trained and tested data with two sizes of datasets: (i) dataset-1: 8 train classes, 8 same validation classes and 11 test classes(3 unknown) (ii)dataset-2: 25 train classes, 25 same validation classes and 35 test classes(10 unknown).

We have used first dataset with less number of classes for verifying algorithms and analysis. Results for 25 classes are presented in Evaluation section of this document.

Dataset-1:

- Training data - 2277 malware samples and 337610 features
- Crossval data - 489 malware samples and 337610 features
- Test data - 568 malware samples and 337610 features

Confusion matrices,precision and recall score for validation data, test data before and after training with concept drift objects for dataset-1 is as below:

4.1 Naive-Bayes with Information Gain metric

Figure 5 shows confusion matrices for cross-validation data, test data and test data after retraining with

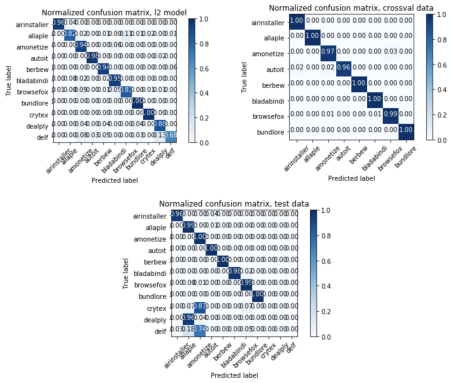

Figure 5: cm for validation data(top left),cm for test data(top right),cm for data trained with concept drift objects(bottom).

concept drift objects. Precision is improved from 73% to 88% after training with drifted samples.

- Precision, recall on crossval data - 0.9291 ,0.9080
- Precision, recall on test data - 0.7352 , 0.7817
- Concept drift aware model: Precision, recall on test data (Threshold: mean + 1*std) - 0.8892 ,0.8697

4.2 Linear SVM with IG metric and SVM hyperplane distance model

Figure-7 shows confusion matrices for cross validation data, test data and for test data after training with detected concept drift objects. Compared to Naive-bayes(fi SVM-Hyperplane distance method gives better result when one standard deviation is used as threshold. Top 5000 Average values of Information gain of each word across sample of each class is used for training SVM. This averaging technique resulted in good classification accuracy and detection of concept drift objects. Figure 6 shows flowchart for this method.

- Precision, recall on crossval data - 0.9900 ,0.9898
- Precision, recall on test data - 0.7746, 0.8574

Concept drift aware model:

- Precision, recall on test data (Threshold: mean + 1*std) - 0.9796 ,0.9789

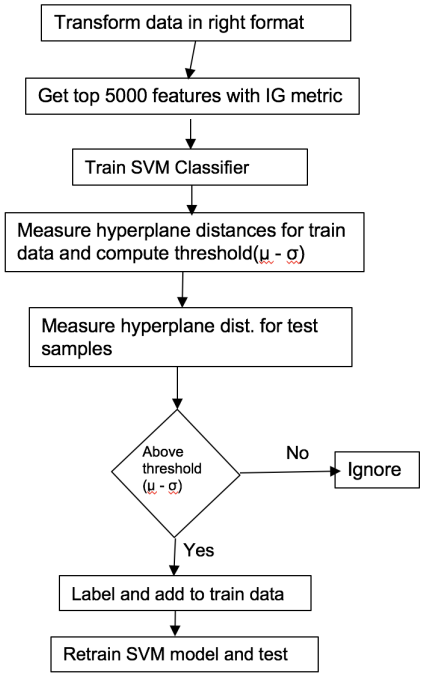

Figure 6: Flowchart for SVM with IG and Hyperplane distance threshold

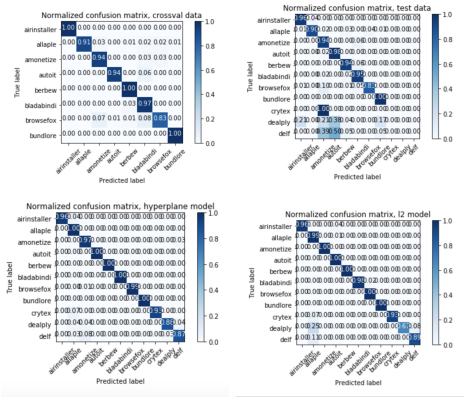

Figure 7: cm for validation data(top left),cm for test data(top right),cm for data trained with concept drift objects(bottom).

4.3 Linear SVM with IG metric and Nearest centroid distance model

Figure-7(bottom left) shows confusion matrices for cross validation data, test data and for test data after training with detected concept drift objects. Compared to Naive-bayes(fi SVM-Hyperplane distance method gives better result when one standard deviation is used as threshold. Top 5000 Average values of Information gain of each word across sample of each class is used for training SVM.This technique resulted in good classification accuracy and detection of concept drift objects. Figure 8 shows flowchart for this method.

- Precision, recall on crossval data - 0.9900 ,0.9898
- Precision, recall on test data - 0.7746, 0.8574

Concept Drift aware model:

- Precision, recall on test data (Threshold: mean + 1*std) - 0.9730, 0.9718

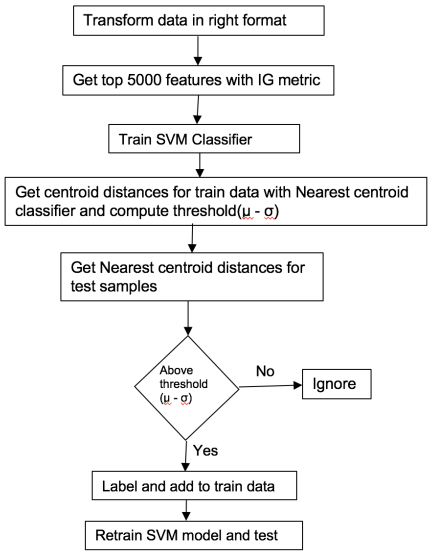

Figure 8: Flowchart- SVM,IG, and Nearest Centroid.

4.4 Linear SVM with PCA and SVM hyperplane distance model

Dimensionality reduction of data done with Principal component analysis. SVM trained with this dimensionally reduced data. With the principal component analysis method we selected 1024 top principal components. Training SVM with this reduced data improved computation and accuracy both. Figure 9 shows flowchart for this method. Figure 10(bottom right) shows confusion matrix for predictions after retraining with drifted samples.

- Precision, recall on crossval data - 0.9734 ,0.9714
- Precision, recall on test data - 0.7636, 0.8504

Concept Drift aware model:

- Precision, recall on test data (Threshold: mean + 1*std) - 0.9845 ,0.9842

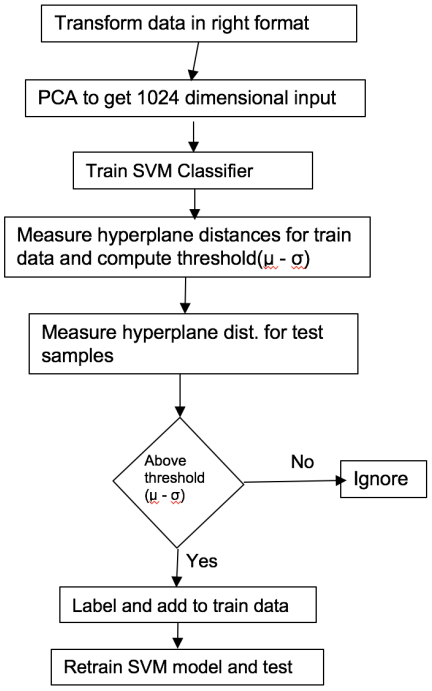

Figure 9: Flowchart- SVM,PCA, and Hyperplane distance threshold

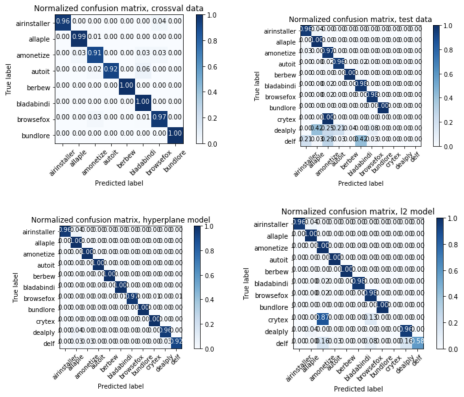

Figure 10: cm for validation data(top left),cm for test data(top right),cm for data trained with concept drift objects(bottom).

4.5 Linear SVM with PCA and Nearest Centroid distance model

In this method nearest centroid Euclidean as distance is used for concept drift detection. Figure 11 shows the flowchart.

- Precision, recall on cross val data - 0.9023 ,0.8925
- Precision, recall on test data - 0.0958, 0.1239

Concept drift aware model:

- Precision, recall on test data (Threshold: mean + 1*std) - 0.9274 ,0.9349

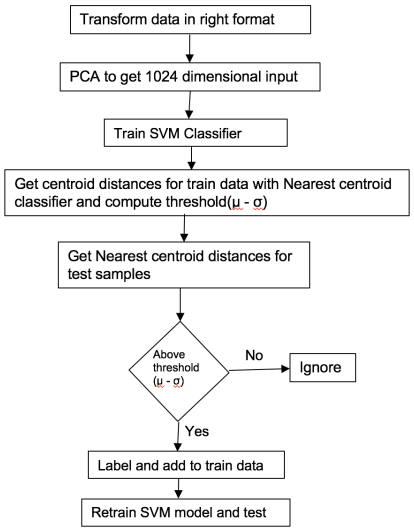

Figure 11: Flowchart- SVM,PCA, and Nearest Centroid.

5 Evaluation

Figure 12 shows precision and recall values obtained for validation and test dataset for - Naive Bayes classifier with 5000IG features, Linear SVM with top 5000 features(Information gain metric) and Linear SVM with 1024 principal components on dataset-1.

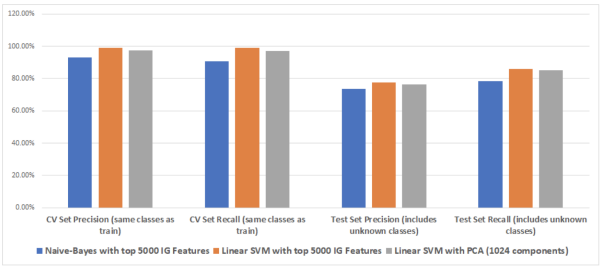

Figure 12: Histogram-1.

Figure 9 and Figure 10 shows results for dataset-1 (8 train classes , 11 test classes(3unknown)) which we used for developing models. Ideally number of old family detected as concept drift object should be zero. It can be observed from Histogram-1 and Histogram-2 that, SVM hyperplane based concept drift detector

with threshold - (mean minus one std) gives better results compared to other classifiers and thresholds. In this case all models performed well.

As we increased number of classes and percentage of unknown classes in model performance deteriorates. Results for dataset-2 or 25 Train classes and 35 test classes(10 unknown) are presented below. The data shows that model is not robust to increasing number of classes and increase percentage of unknown classes.

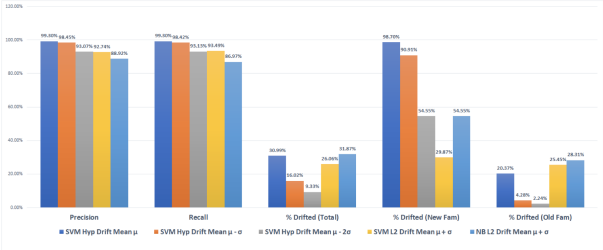

Figure 13: Histogram-2.

5.1 Naive-Bayes with Information Gain metric

Figure 14 shows confusion matrices of cross val data, test and and test data after retraining.

- Precision, recall on crossval data -0.9023 ,0.8925
- Precision, recall on test data - 0.1149 , 0.1190

Concept drift aware model:

- Precision, recall on test data (Threshold: mean + 1*std) - 0.5921 ,0.4465

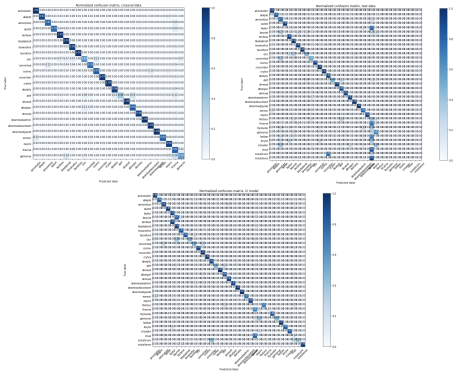

Figure 14: cm for validation data(top left),cm for test data(top right),cm for data trained with concept drift objects(bottom).

5.2 Linear SVM with IG metric and SVM hyperplane distance model

figure 15 shows confusion matrices for this method.

- Precision, recall on crossval data - 0.9645 ,0.9644
- Precision, recall on test data - 0.0797, 0.1248

Concept drift aware model:

- Precision, recall (Threshold: mean + 1*std) - 0.7108 ,0.5941

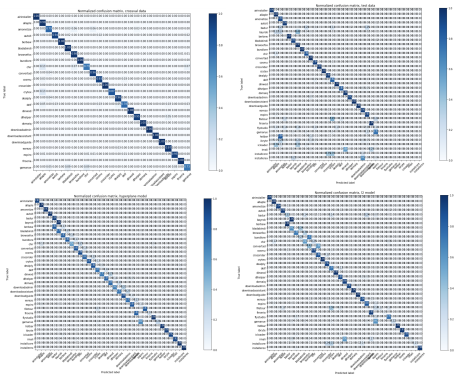

Figure 15: cm for validation data(top left),cm for test data(top right),cm for data trained with concept drift objects(bottom).

5.3 Linear SVM with IG metric and Nearest centroid distance model

figure 15 shows confusion matrices for this method.

- Precision, recall on crossval data - 0.9645 ,0.9644
- Precision, recall on test data - 0.0797, 0.1248

Concept Drift aware model:

- Precision, recall on test data (Threshold: mean + 1*std) - 0.6420, 0.4228

5.4 Linear SVM with PCA and SVM hyperplane distance model

figure 16 shows confusion matrices for this method.

- Precision, recall on crossval data - 0.9548 ,0.9442
- Precision, recall on test data - 0.0958, 0.1239

Concept Drift aware model:

- Precision, recall on test data (Threshold: mean + 1*std) - 0.5963 ,0.4940

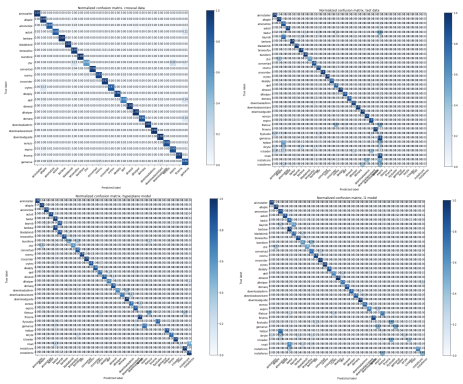

Figure 16: cm for validation data(top left),cm for test data(top right),cm for data trained with concept drift objects(bottom).

5.5 Linear SVM with PCA and Nearest Centroid distance model

figure 16 shows confusion matrices for this method.

- Precision, recall on crossval data - 0.9548 ,0.9442
- Precision, recall on test data - 0.0958, 0.1239

Concept drift aware model:

- Precision, recall on test data (Threshold: mean + 1*std) - 0.5333 ,0.3653

6 Future Work

Making the model more robust and computationally efficient.

7 Code

Our full repository can be found here: https://github.com/vvb231/ML-Cybersecurity-Project.git

8 Related work and references

- G. Severi, T. Leek, B. Dolan-Gavitt, MALREC: Compact Full Trace Malware Recording for Retrospective Deep analysis.
- R. Jordaney,K. Sharad, S. Dash,Z. Wang,D.Papini, Trancend:Detecting Concept Drift in Malware Classification Models.