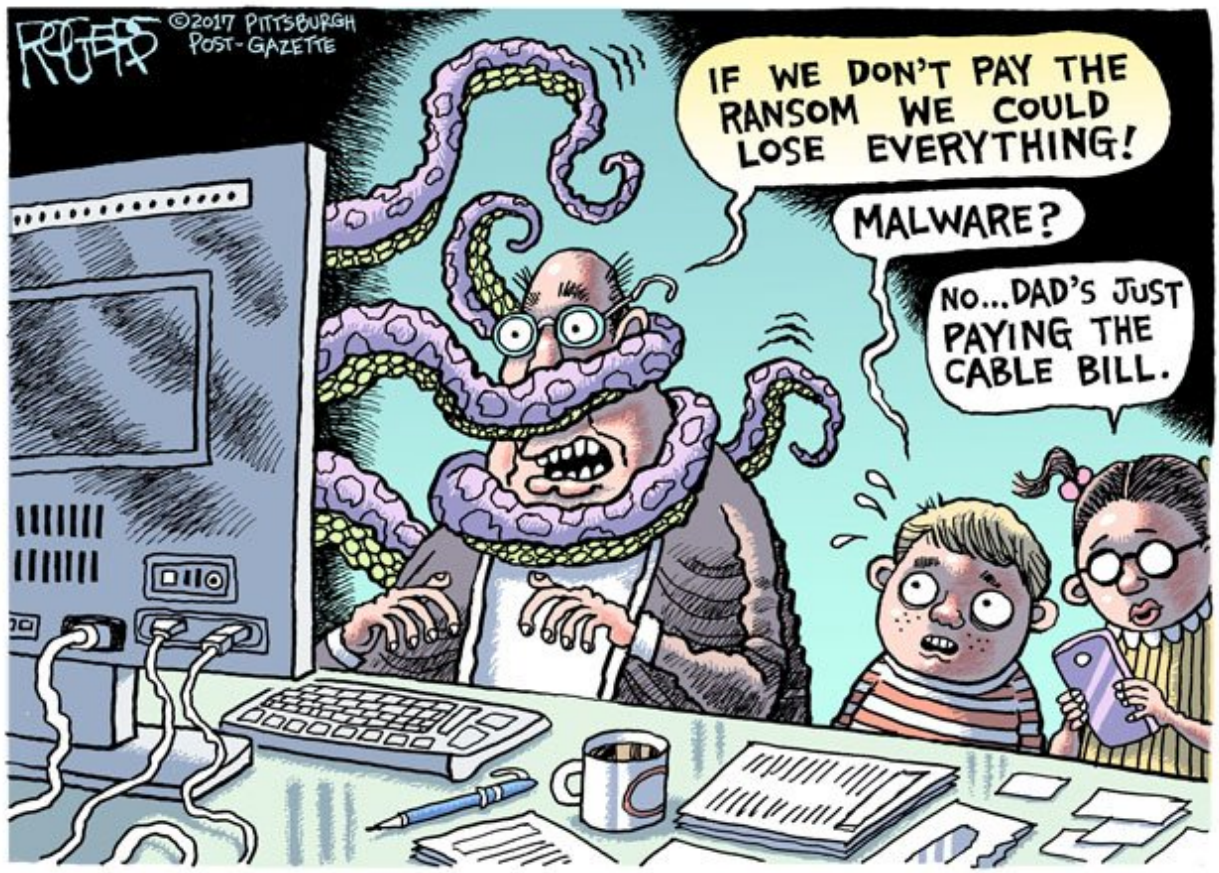
Machine Learning for IT Security

Detecting Malware

Anthony Balducci

2023-12-01



Overview

The first computer virus, called Elk Cloner, was discovered on a Mac in 1982 (interestingly enough it was written by a 15 year old as a 'joke'. This was followed by the first PC-based malware, known as Brain, in 1986. During the late 1980s and early 1990s, the most malicious programs were simple boot sector and file infectors spread via floppy disk. As computer network adoption and expansion continued, malware distribution became easier, and volume increased.

By the mid-1990s, businesses became increasingly affected, due in large part to macro viruses, meaning propagation had become network-driven. Distribution was further accelerated by an increase in internet use

and the adoption of Web 2.0 technologies. By the late 1990s, viruses had begun impacting home users, with email propagation ramping up.

Further, an increase in the use of exploit kits led to an explosion of malware delivered online during the 2000s. Since then, the number of malware attacks has grown exponentially.

In 2022, it was reported that 75% of organizations experienced malware activity that spread from one employee to another. Ransomware attacks frequently lead to disruption of business. Almost 70% of organizations may have understaffed cybersecurity teams. So, from its humble beginnings as a curiosity in the 1980s, malware has evolved into a significant threat to individuals and organizations alike.

Further, ever increasingly Machine Learning and Deep Learning have been coming in *support* of computer defense. A classic example of this is in the use of the Naive Bayes classifier for defending against spam[1]. However where strong foundations are built there remain always those looking to tear them down. I read just yesterday in the news about how Google has upgraded their spam filtering system for the first time in 10 years.[2].

While Bayesian methods generally work quite well still, apparently spammers are increasingly using 'adversarial attacks' to get through the firewall. One of these methods is the extensive use of Unicode characters, instead of the standard Roman Typeset such as they vaguely resemble a recognizable word, but otherwise would not be picked up by a standard computerized dictionary given that there could be endless possible variants.

Seeing as I also have a professional certificate in C and Linux from Dartmouth college I definitely felt this to be an interesting area of study. To be honest I struggled a bit in so far as selecting a final project– There are a lot of really interesting data sets out there, but they have been heavily cherry picked.

Thus I settled on Parthajit Borah and Dhruba K. Bhattacharyya's recent (8/14/2023) TUANDROMD malware data set via the UC Irvine Machine Learning Repository (see link in footnote)[3].

The dataset is, in its original form, 242 x 4464 with 241 features symbolizing different states of activation that occur (or not) in the presence of what they term 'goodware' in the Label category (malware_data[,242]).

Unfortunately, upfront I must humbly apologize slightly– While they do have an accompanying paper with the data it is behind a paywall. I did try to reach out to them via ResearchGate to request a review copy, but unfortunately never heard back from them. My health has not been so well the past few years and I have been unemployed since the pandemic so money has been tight. It is my sincere hope that by finally completing this program it will provide me with more opportunties in a new career in a remote role. Thus, with out this I am unable to give a ton of info on exactly *what* the results of my analysis mean– And even actually, to my surprise it worked out much better (and was much easier) than I had anticipated.

All that aside, I feel it is not all that unusual these days for a Data Scientist to be hired for a project in a field for which the data they know all the tools and methods, but less so in that field they are consulting for, what all the end results of their analysis are.

However from my results here it does show machine learning to be a very promising addition to the IT security field. We simply have too many '0-Days' as it is.

Methods/Analysis

So, as mentioned, I expected this project to be much more challenging than it turned out to be. In fact I would say the brunt of the work turned out to be mostly data cleansing / wrangling.

As of course as I am still learning, I first attempted to proceed with some standard regressions of the form:

[1]https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering
[2]https://arstechnica.com/gadgets/2023/12/gmails-ai-powered-spam-detection-is-its-biggest-security-upgrade-in-years/
[3]https://archive.ics.uci.edu/dataset/855/tuandromd+(tezpur+university+android+malware+dataset)

```
fit <- train(lm(Label ~ .))
```

However my results kept coming out incoherent or not registering any end values. That is when I first realized, given that (almost– get to that in a second) all the features were binary factors of zero or one pointing to a 'Label'– Either 'goodware' or 'malware', this was fundamentally a classification and not a regression problem.

Researching this matter further, at least as far as base R goes, for classification the following methods are available:[4]

1. Logistic Regression
2. Support Vector Machines (SVM)
3. K-Nearest Neighbor (KNN)
4. Naive Bayes
5. Decision Trees

Originally when I started this project I was kind of interested in trying out one of the few neural net packages for R. It was interesting to me that Prof. Irizarry chose to bring up the MNIST set with KNN. I have written some single hidden layer (i.e. simple) neural nets (only forward propagation) in Python and Javascript, and I must admit, for certain tasks they do work *really well* with very little preprocessing needed.

However, here this would have been an overkill. So I selected SVM to start with which I hadn't worked with before, but I was still getting various errors in having the analysis go through.

Thus, with exploration I found the following problems and included are my fixes to them:

```
# During my first run at analysis I was made aware of the fact that certain columns
# only have a single factor attribute, meaning they both prevent further analysis
# and otherwise add no 'information' to the equation-- thus these columns are stripped
# and removed.

malware_data<- malware_data[, sapply(malware_data, function(col) length(unique(col))) > 1]
#malware_test_set <- malware_test_set[, sapply(malware_test_set, function(col) length(unique(col))) > 1

sum(is.na(malware_data))

# Further a run over the data shows we are potentially dealing with a number of blank
# entries or 'NA's in the data set. From this examination we are looking at 241 columns
# with potential candidates:

names(which(colSums(is.na(malware_data)) > 0))

# Interestingly, creating a frequency table() each of the 241 columns with an NA has
# only one entry. Perhaps it is a particular row that is affected ?

table(names(which(rowSums(is.na(malware_data)) > 0)))

# Okay, so luckily it appears only two rows are affected: 1 and 2534. Let's take a look.

malware_data[1,]
malware_data[2534,]

# Hmm, it is not immediately obvious why these two rows are so affected, but out of
```

[4] https://www.javatpoint.com/r-classification

```
# nrow(malware_data) I feel it is safe to just remove them rather than insert some
# potentially confusing offset value that affects the data or the total number of factors

malware_data <- na.omit(malware_data)

# Now double check:

sum(is.na(malware_data)) # sum is '0'-- no more NA's present
```

Once those issues were resolved I was now ready to run my analysis.

Results

My one (unexpected) point of contention here is that my first run results were so good with SVM, it almost didn't make sense to run another model– Though I realize the requirements asked us to.

My Confusion Matrix from SVM alone:

```
Prediction goodware malware
  goodware       87       5
  malware         3     352

               Accuracy : 0.9821
                 95% CI : (0.965, 0.9922)
    No Information Rate : 0.7987
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9448

 Mcnemar's Test P-Value : 0.7237

            Sensitivity : 0.9667
            Specificity : 0.9860
         Pos Pred Value : 0.9457
         Neg Pred Value : 0.9915
             Prevalence : 0.2013
         Detection Rate : 0.1946
   Detection Prevalence : 0.2058
      Balanced Accuracy : 0.9763

       'Positive' Class : goodware
```

Notably an unexpectedly *extremely* high accuracy rate, along with excellent specificity *and* sensitivity. I had selected 90/10 for my train/test set split.

Results for the further, second KNN model were similar, though as I had expected after the first result with SVM, could only possibly get worse:

```
Confusion Matrix and Statistics

          Reference
Prediction goodware malware
  goodware       82       2
```

```
 malware         8     355

              Accuracy : 0.9776
                95% CI : (0.9592, 0.9892)
   No Information Rate : 0.7987
   P-Value [Acc > NIR] : <2e-16

                 Kappa : 0.9287

Mcnemar's Test P-Value : 0.1138

           Sensitivity : 0.9111
           Specificity : 0.9944
        Pos Pred Value : 0.9762
        Neg Pred Value : 0.9780
            Prevalence : 0.2013
        Detection Rate : 0.1834
  Detection Prevalence : 0.1879
     Balanced Accuracy : 0.9528

      'Positive' Class : goodware
```

Conclusion

The immediate results from my survey/analysis look quite promising for the future role of Machine Learning and Data Science. I must admit this undertaking has peaked my interest in this area for future research. However, as mentioned when I selected the dataset, I also didn't realize it would be less work than I expected.

In the end though (at least thus far), it is people, not computers that write viruses and malware, is it not ? Perhaps they'll learn to protect us from 'them' after all.