Machine Learning and Anomaly Detection

Azka Javaid

December 16, 2016

Abstract

Machine Learning has various applications in cybersecurity where it can be used to extract anomalies, potentially indicative of a cyberattack, from unstructured data available from surface, dark and deep web. Anomalies can be detected through natural language processing (NLP), sentiment analysis, and facial and speech recognition. Additionally cognitive computing, artificial intelligence and deep learning can be combined with machine learning to automate the process of anomaly detection and detect attacks in dynamic conditions like that of the Advanced Persistent Attacks (APT). IBM Security, for example, is planning to use Watson's ability to process thousands of journal articles as a means of tracking and detecting attack signatures. This study explores one such application of machine learning to predict whether an HTTP request is normal or anomalous using the 2010 HTTP CSIC (Spanish Research National Council) dataset.

Problem

Detect whether an incoming HTTP request is normal or anomalous based on features like the HTTP request method type, the length of the content field, the payload count, number of unique cookie ids per URL and number of unique index counts per URL.

Methods

I used the 2010 HTTP CSIC dataset and R for my analysis. CSIC data contains more than 36,000 normal and greater than 25,000 anomalous web requests, encompassing attacks like SQL injection, buffer overflow, information gathering, files disclosure, CRLF injection, XSS, and parameter tampering. Data features/variables included HTTP request method (whether request if of type GET/PUT/POST), URL, protocol type (which is HTTP/1.1 for all requests), cookie id, content type, content length and payload. Additionally where the URL of a HTTP request had multiple key=value pairs, authors Carmen Torrano Giménez, Alejandro Pérez Villegas and Gonzalo Álvarez Marañón chose to split those pairs in multiple packets and created an index attribute to track the packet number. A label attribute to classify each request as either normal or anomalous was also created by the authors.

In addition to the features present, I created a countJSession predictor, which assessed number of unique cookie ids for every URL and the countIndex predictor, which assessed the unique index counts for every URL. I also created a count payload attribute (total number of characters in the payload field), which could be a potential indicator of anomalies since a hacker may flood user's system with fields of large payload size. In conclusion, I constructed a model to predict whether a HTTP request is normal or anomalous based on method, content length, count payload, countJSession and countIndex features.

Analysis and Code

```
load("cyber1.Rda") #Rda random sample of the original data
```

Data Characteristics: I performed my analysis on a sample of 100,000 HTTP requests from the original data. The sampled data labeled cyber contained about 53,340 anomalous and 46,660 normal requests.

```
cyber <- cyber1
names(cyber) #generates variable names
```

```
##  [1] "X"              "index"          "method"         "url"
##  [5] "protocol"       "userAgent"      "pragma"         "cacheControl"
##  [9] "accept"         "acceptEncoding" "acceptCharset"  "acceptLanguage"
## [13] "host"           "connection"     "contentLength"  "contentType"
## [17] "cookie"         "payload"        "label"
```

```
nrow(cyber) #total number of observations
```

```
## [1] 100000
```

```
nrow(filter(cyber, label == "anom")) #total number of anomalous requests
```

```
## [1] 53340
```

```
nrow(filter(cyber, label == "norm")) #total number of normal requests
```

```
## [1] 46660
```

Feature Selection: I first create a countPayload predictor by converting the payload to a character field (using as.character) and then counting the number of characters in the field (using nchar). The methodChar feature is a numeric representation of the categorical method predictor. The countJSession and countIndex attributes are created by summing the unique cookie and index attributes over a URL, respectively, using the length, unique and by function specifications. Lastly, I select all the predictors used to predict the category for a label in dataset labeled cyber2 based on the select function in dplyr package.

```
cyber$charPayload <- as.character(cyber$payload)
cyber$countPayload <- nchar(cyber$charPayload) #total characters in payload
cyber$countPayload <- as.numeric(cyber$countPayload)

cyber$contentLength <- as.numeric(cyber$contentLength)
cyber$methodChar <- ifelse(cyber$method == "GET", 0, ifelse(cyber$method == "POST", 1, 2))
cyber$label <- as.factor(cyber$label)

#number of unique cookie ids by url
cyber <- as.data.table(cyber)[, countJSession := length(unique(cookie)), by = url][]
#number of unique index ids by url
cyber <- as.data.table(cyber)[, countIndex := length(unique(index)), by = url][]

cyber2 <- cyber %>% dplyr::select(countPayload, countJSession, countIndex,
                                  contentLength, methodChar, label)
head(cyber2) #glimpse of the dataset
```

```
##    countPayload countJSession countIndex contentLength methodChar label
## 1:           20          4772       4663           383          0  anom
```

```
## 2:           26          4772        4663           383          0  anom
## 3:           11          4514        4431            71          1  anom
## 4:           23          4779        4674           383          0  anom
## 5:           12          4772        4663           214          1  anom
## 6:           13          4772        4663           132          1  anom
```

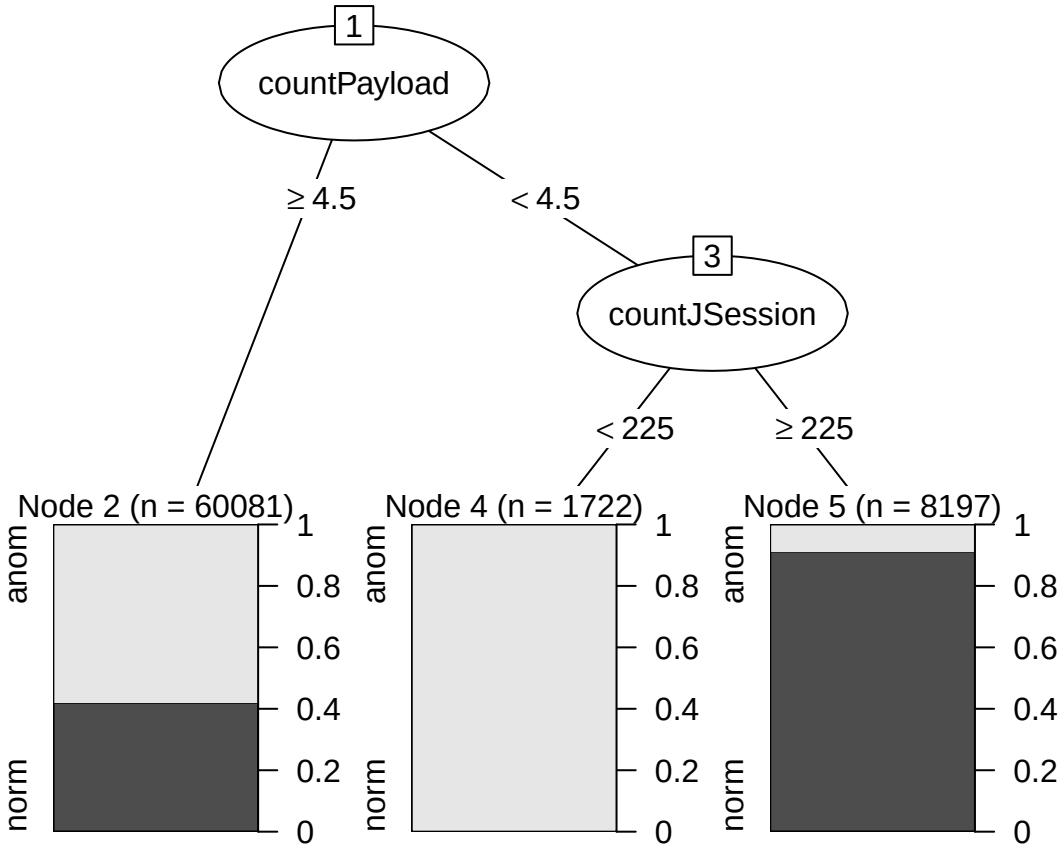
Training and Test sets: A key feature of machine learning algorithms involves partitioning the original dataset in test and training sets so that the model can be trained on the training set and subsequently tested on the test set. Below, I perform a 70/30 split with 70% of the data as the training and 30% as the test set.

```
n <- nrow(cyber2)
shuffled <- cyber2[sample(n),] #data shuffled before splitting
train <- shuffled[1:round(0.7 * n),]
test <- shuffled[(round(0.7 * n) + 1):n,]
```

Supervised Learning Decision Tree: Since a label field detailing whether an incoming request was normal or anomalous was present in the dataset, supervised learning algorithms were used for model building. Given that the label field contained only two classifications of normal and anomalous, a binary decision tree was used. In the code below, I regress label against all the predictors in the training dataset (which include countPayload, countJSession, countIndex, contentLength and methodChar) to predict whether a request is normal or anomalous. I then plot the tree using the party package in R. Theoretically, in a decision tree, each leaf corresponds to a predicted class for an observation.

The decision tree shows that an anomalous request is characterized by a low payload count (<=4.5) and a low countJSession (<225). This indicates that a HTTP request is more likely to be anomalous if the number of characters in the payload field is greater than or equal to 4.5 and if the number of unique JSession ids for a URL is less than 225. In comparison, a low payload count (<=4.5) and high countJSession (>=225) is characteristic of a normal request.

```
tree <- rpart(label ~ ., data = train, method = "class")
plot(as.party(tree))
```



Assessing Decision Trees: I assess the accuracy of the decision tree model above by testing it on the test set. The predictions for my model are compared to the actual labels in the test set (test$label) in a confusion matrix. Sum of the diagonals over the total number of values in the matrix indicates the model accuracy which evaluates to about 63%.

```
pred <- predict(tree, test, type = "class")
conf <- table(test$label, pred) #building confusion matrix
print(sum(diag(conf)) / sum(conf)) #model accuracy
```

```
## [1] 0.6354333
```

Supervised Learning Neural Networks: I then explored another classification algorithm, neural networks, to predict whether a request is normal or anomalous based on countPayload, countJSession, countIndex, contentLength and methodChar. A seed is set first to preserve/remember the random iteration of the network. The label response variable is converted to a numeric and a neural net function is used with 3 layers. In general, neural networks show the convergence of input predictor variables to the output (label) with highlighted weights and specified hidden layers. Akin to a biological neuron, this algorithm remains much of a black box in regards to theoretical details.

```
set.seed(13)
cyber2$label <- as.numeric(cyber2$label)
nn <- neuralnet(
  label~countPayload+contentLength+countJSession+countIndex + methodChar,
  data=cyber2, hidden=3, linear.output=FALSE)
plot(nn)
```

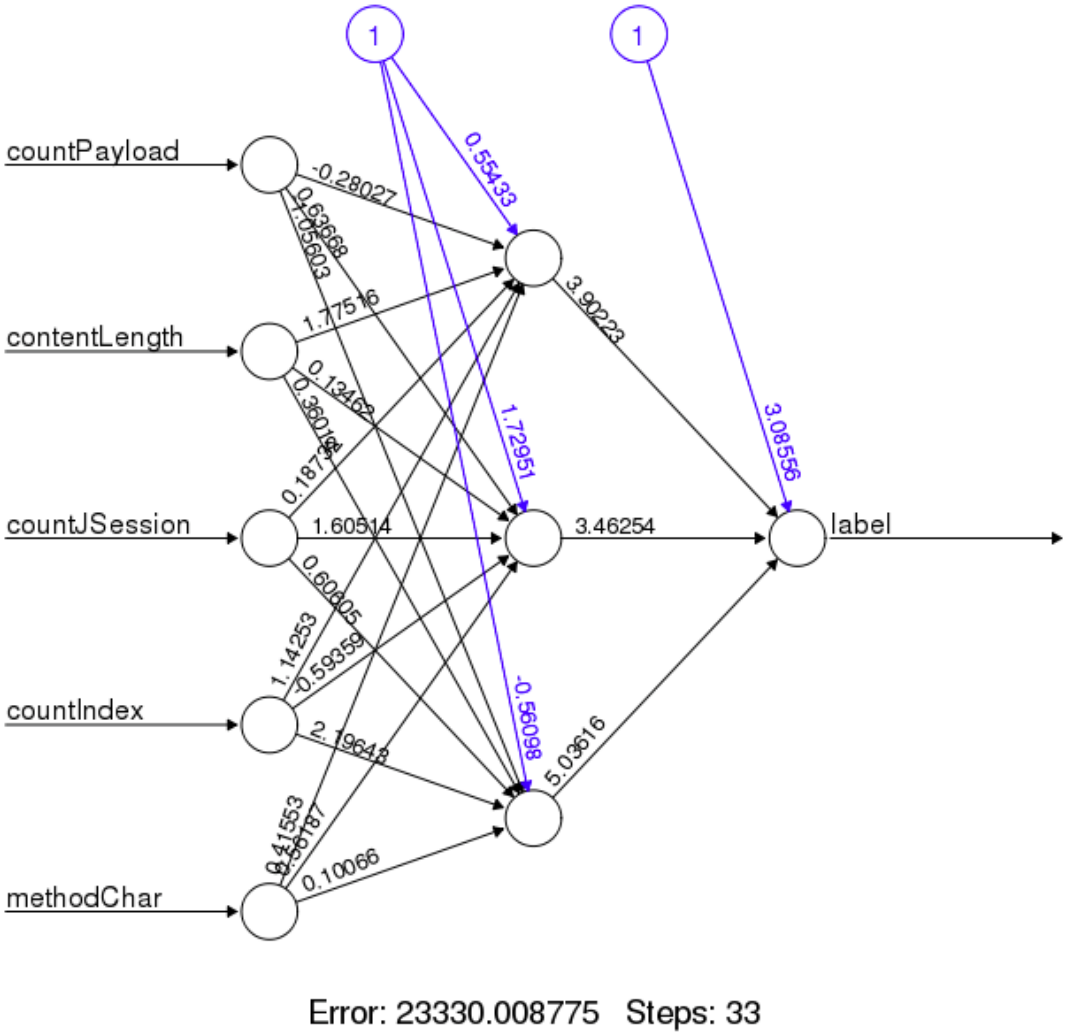

1
1
countPayload
contentLength
countJSession
countIndex
methodChar
label
0.55433
-0.28027
-0.62668
-1.05603
1.77516
0.13462
0.36018
0.18738
1.60514
0.60805
1.14253
-0.59359
2.19643
0.41553
0.58187
0.10066
3.90223
1.72951
3.08556
3.46254
-0.56098
5.03616
Error: 23330.008775 Steps: 33

Figure 1:

Discussion

I only analyzed supervised learning techniques in this project since the dataset already contained labeled normal and anomalous HTTP requests. An extension of this project could include analyzing unsupervised learning techniques (where predetermined knowledge of whether a request is normal and anomalous is not present) like k-means clustering to detect an anomalous event. Unsupervised techniques essentially entails use distance metrics like euclidean to cluster observations as similar/dissimilar to each other. Each cluster is then indicative of a label or classification (normal or anomalous). Other machine learning algorithms include Random Forests and Naive Bayes (intuitive overview of machine leaning and NLP techniques can be found from Sebastian Raschka's blog, link in reference). Another extension could include studying how sentiment analysis and natural language processing is used to detect attacks in deep and dark web infrastructures. Machine learning and cognitive intelligence is also pretty useful in industry for threat analytics (see references for a link to a threat analysis dashboard from IBM).

Since machine learning is the process of extracting insights from preexisting data, it may not be a viable and practical option in highly unpredictable scenarios. A machine learning algorithm can therefore produce false positives where an observation is categorized as anomalous when it is not. A more conjoined human and machine effort, inspired by cognitive and artificial intelligence, is needed to address these false positives as well as detect attacks with uncharacteristic signatures.

Summary

The main inspiration behind this project was to show the integrated nature of statistical analytics and computing disciplines like cybersecurity. Machine learning provides the ability to navigate both realms as data is used to detect anomalies and identify/prevent future cyber attacks. Though as hackers become more savvy, savvier dynamic algorithms with the ability to detect predictive anomalies as well as unforeseeable attacks using both machine and human input are wanted.

References and Code Link

"An Anomaly-Based Web Application Firewall." Proceedings of the International Conference on Security and Cryptography (2009): n. pag. Web. Dua, Sumeet, and Xian Du. Data Mining and Machine Learning in Cybersecurity. Boca Raton: CRC, 2011. Print.

Giménez, Carmen Torrano, Alejandro Pérez Villegas, and Gonzalo Álvarez Marañón. "HTTP DATASET CSIC 2010." HTTP DATASET CSIC 2010. N.p., n.d. Web. 18 Dec. 2016.

Giménez, Carmen Torrano, Alejandro Pérez Villegas, and Gonzalo Álvarez. "An Anomaly-Based Web Application Firewall." Proceedings of the International Conference on Security and Cryptography (2009): n. pag. Web.

IBM's Threat Analysis Dashboard: https://abny2015.shinyapps.io/ThreatDashboard/

Torrano-Gimenez, Carmen, Hai Thanh Nguyen, Gonzalo Alvarez, Slobodan Petrovic, and Katrin Franke. "Applying Feature Selection to Payload-based Web Application Firewalls." 2011 Third International Workshop on Security and Communication Networks (IWSCN) (2011): n. pag. Web.

Sebastian Raschka's (author of Python Machine Learning) blog for a perceptive and unpedantic overview of machine learning algorithms: https://sebastianraschka.com/blog/index.html

Github link to the code: https://github.com/ajavaid17/MLCyberSec/blob/master/CodePresDec12.Rmd