

# Malware Classification

This code snippet shows you how to train a Naive Bayes (NB) algorithm for malware classification. For this purpose, we can use different features to train our algorithm. This includes features that are obtained using both static code analysis and/or analysis of the dynamic behavior of the malware.

**The dataset:** We will use the dataset published at <https://data.mendeley.com/datasets/w2w8gjsnt/1> (<http://dx.doi.org/10.17632/w2w8gjsnt.1#file-6806d890-e13f-4644-abc2-630cca78216f>). This data set contains a total of 1944 features that are obtained from the static and dynamic analysis of ~ 19400 malware samples including malware samples from APT attacks. We pre-processed the dataset by removing the first three columns and the last seven columns. Instead, we added a label column at the end of the dataset.

**Machine learning (ML) task:** We want to create an NB model to identify (classify) the type of malware, so we are solving a classification problem here.

```
myDataOrg <- read.csv("malware.csv", header=T)
dim(myDataOrg) # check dimensions of myData
```

```
## [1] 19457 1935
```

```
levels(myDataOrg$label) # check different levels (values) for each class
```

```
## [1] "Backdoor" "OtherType" "Rootkit" "Spyware" "Trojan" "Unknown"
## [7] "Worm"
```

**Check the balance of the dataset :** Class imbalance is very common issue in Cyber Security dataset, and it's pretty common to expect ratio like 1:100000 between classes (e.g. attack:normal) due to the scarcity of attack data.

Class imbalance is a very common problem in cyber security datasets, and it is quite common that a 1:100000 ratio between classes (e.g. attack: normal) due to the scarcity of attack data. If the class imbalance occurred then it can be affected on model performance. Let's look at ratio between classes in our dataset.

```
print(table(myDataOrg$label))
```

```
##
## Backdoor OtherType Rootkit Spyware Trojan Unknown Worm
##          53      1275      789      709    11034    3957    1640
```

As you can see, our dataset is a hugely imbalance, especially Backdoor: Trojan. A mix of oversampling and undersampling methods could be utilised to balance the dataset, e.g., by increasing the size of Backdoor class and reducing the size of Trojan class. However, this can be resulted information lost in the larger class. Therefore, we will split the dataset into two sets and train two NB models separately (Ensamble technique).

**Note:** It should be noted that NB would not be the best option for this type of dataset, instead we recommend to try out some ensemble techniques. However, we will train NB model in this way as our goal in this post to show you how to train the NB model for this dataset regardless of the accuracy.

```
myDataSet1<-rbind(subset(myDataOrg, label == "Backdoor"),subset(myDataOrg, label == "OtherType"),subset
myDataSet2<-rbind(subset(myDataOrg, label == "Trojan"),subset(myDataOrg, label == "Unknown"),subset(myD
```

We split the original dataset into two subsets. Let's continue with myDataSet1. You can follow the same approach for myDataSet2. The following code makes class sizes equal in myDataSet1.

```
sample.df <- function(df, n) df[sample(nrow(df), n,replace = T), , drop = F]

classSize<-500 # sample from each class size of 500 records
```

```
myData<-rbind(sample.df(subset(myDataSet1, label == "Backdoor"), classSize),sample.df(subset(myDataSet1,
```

Since we want to represent the presence or absence of a certain static/dynamic feature in the malware, we code our dataset as follows.

```
convert_counts <- function(x) {
  x <- ifelse(x > 0, "Y", "N")
}
myData <- data.frame(apply(myData[,1:1934], MARGIN = 2,convert_counts),myData[1935] )
```

**Creating training and validation datasets:** We're going to follow the convention of 80/20 samples ratio to partition the dataset to the training and validation sets. We use the createDataPartition function from the caret package for this purpose.

```
#install.packages("caret") #If the caret package is not installed on your system, uncomment this line t
set.seed(1234)
library(caret) #Loading the library
tr_index <- createDataPartition(myData$label, p=0.80, list=FALSE) # List of 80% of the rows

## Warning in createDataPartition(myData$label, p = 0.8, list = FALSE): Some
## classes have no records ( Trojan, Unknown, Worm ) and these will be ignored

trainSet <- myData[tr_index,] # select 80% of the data for the trainSet
testSet <- myData[-tr_index,] # Select the remaining 20% of data for testSet
```

**Building a NB classifier:** Now we will train our NB classifier using the above trainSet. For this purpose, we will utilize e1071 package in R. Note that the priori probabilities can be computed using the following lines of code. In this case, the priori probabilities for all classes are the same.

```
#install.packages("e1071") #If the e1071 package is not installed on your system, uncomment this line t
library(e1071)
NBclassifier <- naiveBayes(trainSet[,1:1934], trainSet$label) # train the model
```

**Make predictions:** Now let's apply the above model to assign labels for test cases in testSet. Then we create the confusion matrix, a table that is often used to describe the performance of a classifier.

```
testPrediction <- predict(NBclassifier, testSet[,1:1934]) # predict labels for test cases
confusionMatrix(testPrediction, testSet$label) # Print confusion matrix

## Confusion Matrix and Statistics
##
##              Reference
## Prediction  Backdoor OtherType Rootkit Spyware Trojan Unknown Worm
## Backdoor      91       13       16       15        0        0        0
## OtherType      0       47        7       17        0        0        0
## Rootkit        4       16       62       16        0        0        0
## Spyware        5       24       15       52        0        0        0
## Trojan         0        0        0        0        0        0        0
## Unknown        0        0        0        0        0        0        0
## Worm           0        0        0        0        0        0        0
##
## Overall Statistics
##
##              Accuracy : 0.63
##              95% CI : (0.5806, 0.6774)
##      No Information Rate : 0.25
##      P-Value [Acc > NIR] : < 2.2e-16
```

```

##
##          Kappa : 0.5067
##
##  McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##          Class: Backdoor Class: OtherType Class: Rootkit
## Sensitivity          0.9100          0.4700          0.6200
## Specificity          0.8533          0.9200          0.8800
## Pos Pred Value       0.6741          0.6620          0.6327
## Neg Pred Value       0.9660          0.8389          0.8742
## Prevalence           0.2500          0.2500          0.2500
## Detection Rate       0.2275          0.1175          0.1550
## Detection Prevalence 0.3375          0.1775          0.2450
## Balanced Accuracy     0.8817          0.6950          0.7500
##
##          Class: Spyware Class: Trojan Class: Unknown
## Sensitivity          0.5200          NA          NA
## Specificity          0.8533          1          1
## Pos Pred Value       0.5417          NA          NA
## Neg Pred Value       0.8421          NA          NA
## Prevalence           0.2500          0          0
## Detection Rate       0.1300          0          0
## Detection Prevalence 0.2400          0          0
## Balanced Accuracy     0.6867          NA          NA
##
##          Class: Worm
## Sensitivity          NA
## Specificity          1
## Pos Pred Value       NA
## Neg Pred Value       NA
## Prevalence           0
## Detection Rate       0
## Detection Prevalence 0
## Balanced Accuracy     NA

```