Sardar Patel Institute of Technology,Mumbai
Department of Electronics and Telecommunication Engineering
T.E. Sem-V (2018-2019)
ETL54-Statistical Computational Laboratory

Machine Learning approach to Network Security

Name: Yash Hemant Patil **Roll No.** 2016120036

Objective:To detect computer network intrusion using machine learning techniques.

Outcomes:

1. To load dataset in R
2. To process and prepare data
3. Build a model for intrusion detection.
4. Use machine learning algorithms to detect intrusion.
5. Calculate accuracy and confusion matrix and prediction time.

System Requirements: Ubuntu OS with R and RStudio installed and ggplot2

Problem Abstract:

This is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between ``bad'' connections, called intrusions or attacks, and ``good'' normal connections. This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment.

CODE

```
setwd("/home/student/STC/KDDcup")
getwd()
dir()

train_raw <- read.csv("traindata_10percent.csv", stringsAsFactors =
FALSE)
colnames <- read.table("names", skip = 1, sep = ":")
names(train_raw) <- colnames$V1
d <- dim(train_raw)
names(train_raw)[d[2]] <- "label"
names(train_raw)
sum_label <- aggregate(rep(1, d[1]),
                     by = list(train_raw$label),
                     FUN = sum)
names(sum_label) <- c("label", "count")
barplot(beside = TRUE, log10(sum_label$count),
     names.arg = sum_label$label, ylim = c(0,6),
     xlab = "Label", ylab = "log(Count)",
     col = "Blue", main = "The distribution of labels")

library(caret)
l <- train_raw$label
sum(is.na(l))
nzvcol <- nearZeroVar(train_raw)
train_raw <- train_raw[, -nzvcol]

#label into factor
training <- train_raw
training$label <- factor(training$label)

d <- dim(training)
test_raw <- read.csv("testdata_10percent.csv", stringsAsFactors =
FALSE)

# Process the data
names(test_raw) <- colnames$V1
names(test_raw)[dim(test_raw)[2]] <- "label"
```

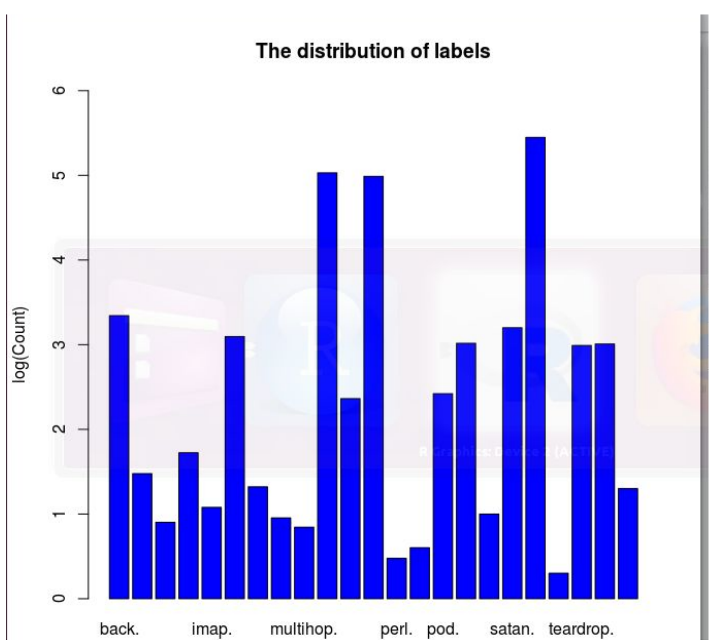
```
# Extract the same features as training data
colnames_train <- names(training)
test_raw <- test_raw[ , colnames_train]

testing <- test_raw
testing$label <- as.factor(testing$label)
library(e1071)

# Build the model
label_result = training[ ,d[2]]
training_data = training[ ,1:(d[2]-1)]
navie_bayes_tree_model = naiveBayes(as.factor(label_result)~.,
                                     training_data)

# Predict the testing and check processing time
start_time = Sys.time()
testing_data = testing[ , 1: (d[2]-1)]
navie_bayes_pred = predict(navie_bayes_tree_model, testing_data)
golden_answer = testing[ , d[2]]
navie_bayes_pred = factor(navie_bayes_pred, levels
=levels(golden_answer))
NB_accuracy <- mean(golden_answer == navie_bayes_pred,na.rm = TRUE)
endtime = Sys.time()
# Get the accuracy
td = endtime - start_time
td
pred = navie_bayes_pred
truth = testing$label
confusionMatrix(pred,truth)
```

Output



`Time difference of 2.729063 mins`

Confusion Matrix and Statistics

	Reference						
Prediction	back.	buffer_overflow.	ipsweep.	land.	neptune.	normal.	pod.
back.	83	0	0	0	0	0	0
buffer_overflow.	0	2	0	0	0	756	0
ipsweep.	0	0	537	0	0	3034	0
land.	0	0	0	3	71	41	0
neptune.	0	0	0	0	210713	46	0
normal.	14	0	0	0	0	21501	0
pod.	0	0	0	0	0	1145	20
portsweep.	0	0	0	0	9	2	0
satan.	0	0	0	0	50	3	0
smurf.	0	0	0	0	0	93	0
teardrop.	0	0	1	0	0	10244	0

	Reference			
Prediction	portsweep.	satan.	smurf.	teardrop.
back.	0	0	0	0
buffer_overflow.	0	0	0	0
ipsweep.	192	0	0	0
land.	0	0	0	0
neptune.	0	0	0	0
normal.	0	0	0	0
pod.	0	0	227	0
portsweep.	1602	0	0	0
satan.	0	0	0	0
smurf.	0	0	36125	0
teardrop.	0	0	82	100

Overall Statistics

Accuracy : 0.9442
95% CI : (0.9433, 0.945)

No Information Rate : 0.7354
P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.8712
Mcnemar's Test P-Value : NA

Statistics by Class:

	Class: back.	Class: buffer_overflow.	Class: ipsweep.	Class: land.
Sensitivity	0.8556701	1.000e+00	0.998141	1.000e+00
Specificity	1.0000000	9.974e-01	0.988727	9.996e-01
Pos Pred Value	1.0000000	2.639e-03	0.142705	2.609e-02
Neg Pred Value	0.9999512	1.000e+00	0.999996	1.000e+00
Prevalence	0.0003383	6.976e-06	0.001877	1.046e-05
Detection Rate	0.0002895	6.976e-06	0.001873	1.046e-05
Detection Prevalence	0.0002895	2.644e-03	0.013125	4.011e-04
Balanced Accuracy	0.9278351	9.987e-01	0.993434	9.998e-01

	Class: neptune.	Class: normal.	Class: pod.	Class: portsweep.
Sensitivity	0.9994	0.58324	1.000e+00	0.892977
Specificity	0.9994	0.99994	9.952e-01	0.999961
Pos Pred Value	0.9998	0.99935	1.437e-02	0.993180
Neg Pred Value	0.9983	0.94206	1.000e+00	0.999327
Prevalence	0.7354	0.12859	6.976e-05	0.006257
Detection Rate	0.7350	0.07500	6.976e-05	0.005588
Detection Prevalence	0.7351	0.07504	4.855e-03	0.005626
Balanced Accuracy	0.9994	0.79159	9.976e-01	0.946469

	Class: satan.	Class: smurf.	Class: teardrop.
Sensitivity	NA	0.9915	1.0000000
Specificity	0.9998151	0.9996	0.9639667
Pos Pred Value	NA	0.9974	0.0095905
Neg Pred Value	NA	0.9988	1.0000000
Prevalence	0.0000000	0.1271	0.0003488
Detection Rate	0.0000000	0.1260	0.0003488
Detection Prevalence	0.0001849	0.1263	0.0363695
Balanced Accuracy	NA	0.9956	0.9819833

Conclusion
We processed and plotted the data available in the dataset to identify the different types of intrusions on the network and their frequency. We created a prediction model based on Naive-Bayes Tree Model and

trained it on 10% of the data. Then we passed the test data to test the accuracy of the model. We used the `confusionMatrix()` command to get a detailed analysis of the model. From the summary we saw that the kappa value is >0.8 indicating higher accuracy of the model which was found to be 94.42%.