

Exploratory Data Analysis toward Cloud Intrusion Detection

by

Aigerim Mashkanova

B.Sc., International Information Technologies University, 2013

A Project Submitted in Partial Fulfillment of the  
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

© Aigerim Mashkanova, 2019  
University of Victoria

All rights reserved. This project may not be reproduced in whole or in part, by photocopying or other means, without the permission of the author.

Exploratory Data Analysis toward Cloud Intrusion Detection

by

Aigerim Mashkanova

B.Sc., International Information Technologies University, 2013

Supervisory Committee

---

Dr. Sudhakar Ganti, Co-Supervisor  
(Department of Computer Science)

---

Dr. Issa Traore, Co-Supervisor  
(Department of Electrical and Computer Engineering)

## ABSTRACT

Cloud computing is one of the fastest growing areas of information technologies. The growth rate and the prosperity of this industry have led to important security concerns. For the Masters Project, the ISOT Cloud Intrusion Dataset (ISOT-CID) was utilized to explore cloud anomaly detection using different machine learning techniques. The dataset involves terabytes of data that contain various attacks and normal activities gathered in a real cloud environment. Three different supervised machine learning techniques were applied to analyze anomalous behavior in network traffic. The implemented experiments demonstrated that the Logistic Regression classifier achieves the lowest false negative rate while the Naive Bayes has achieves the highest accuracy rate.

# Contents

<b>Supervisory Committee</b>	<b>ii</b>
<b>Abstract</b>	<b>iii</b>
<b>Table of Contents</b>	<b>iv</b>
<b>List of Tables</b>	<b>vi</b>
<b>List of Figures</b>	<b>vii</b>
<b>Acknowledgements</b>	<b>viii</b>
<b>Dedication</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Structure of the Project . . . . .	2
<b>2 Related Work</b>	<b>3</b>
<b>3</b>	<b>5</b>
3.1 Dataset . . . . .	5
3.2 Feature Extraction . . . . .	5
3.3 Feature Model . . . . .	6
<b>4</b>	<b>9</b>
4.1 Feature Selection . . . . .	9
4.2 Classification Algorithms . . . . .	11
4.3 Evaluation Metrics . . . . .	13
4.4 Evaluation Methods . . . . .	14
4.5 Evaluation Results . . . . .	14

5 Conclusion and Future Work	17
Bibliography	18

# List of Tables

Table 3.1	The list of flow statistics computed using Tralyzer . . . . .	7
Table 4.1	Truth Tables and Accuracy Measures for when using all features.	15
Table 4.2	Truth Tables and Accuracy Measures for using selected features.	16

# List of Figures

Figure 3.1 Example of network flow graph extracted from the dataset . . .	6
Figure 3.2 Sample attack scenario for Day 1 [11] . . . . .	6
Figure 4.1 Feature importance chart obtained using Boruta algorithm . . .	10
Figure 4.2 Correlation matrix between all features . . . . .	10
Figure 4.3 Correlation of first set of feature pairs . . . . .	11
Figure 4.4 Correlation of second set of feature pairs . . . . .	11
Figure 4.5 The Naive Bayes code snippet . . . . .	12
Figure 4.6 The Neural Network code snippet . . . . .	13
Figure 4.7 The Logistic Regression . . . . .	13

## ACKNOWLEDGEMENTS

I would like to thank:

**my supervisor, Dr Issa Traore**, for his continuous support, encouragement, patience and for giving me this great opportunity to work under his supervision.

**my supervisor, Dr Sudhakar Ganti**, for his continuous support, encouragement, patience and for giving me this great opportunity to work under his supervision.

**my father and my mother, Khaden Mashkanov and Mamila Mashkanova** for their love, friendship and acceptance of me.



## DEDICATION

I would like to dedicate this to my parents and relatives who have been a great source of inspiration.

# Chapter 1

## Introduction

Cloud computing is an innovative technology that provides dynamically scalable computing resources and applications through Internet services managed by a service provider with payment for actual services or resources received. Cloud computing is becoming increasingly popular, especially recently, when limited financial resources are forcing companies to optimize costs. There is no need to spend huge amounts of money on creating your own data centers, on paying for licensed software, on maintaining qualified personnel. You can simply automate all IT processes by purchasing ready-made SaaS, DaaS, IaaS or PaaS packages. However, the enthusiasm for cloud computing is curbed by the fact that it brings new security threats and challenges. To address these challenges cloud service providers and customers have adopted and adapted the existing security mechanisms used to protect conventional networks. One such prominent technology is intrusion detection system (IDS).

There are two common types of intrusion detection systems: signature-based and anomaly-based. The signature-based methods rely on a database patterns or signatures extracted from known attack methods. Anomaly detection rely on a characterization of the normal behavior of the network and digital assets. Any deviation from such behavior is flagged as an intrusion. Signature-based methods can accurately detect known attack patterns; but they are ineffective in identifying novel intrusion patterns. In contrast, while anomaly detection has the potential to detect novel attack methods, they are often plagued by huge amount of false alarms.

Most existing cloud intrusion detection systems have been designed and tested using conventional network dataset. However, due to architectural differences between cloud networks and conventional networks, there are important in the traffic pattern, which impact the effectiveness of the intrusion detection process. The ISOT Lab has

collected a new dataset for cloud intrusion detection. The dataset was collected in a real cloud environment, and involved a variety of attack methods, including both conventional and cloud-specific attack methods. The goal of the current project is to conduct an early exploration of cloud anomaly detection using a small subset of the dataset and by investigating different machine learning algorithms. To achieve the goal of the study, the following tasks were set and solved: Construction and feature extraction based on an existing feature model. Comparison of different machine learning methods for detecting abnormal network traffic behavior. Evaluation of the effectiveness of the detection methods.

## 1.1 Structure of the Project

**Chapter 1** contains an overview of the project followed by a summary of the structure of the project.

**Chapter 2** describes the research problem and related work.

**Chapter 3** describes the dataset and the feature model.

**Chapter 4** explains the experiments and the methodology used for the project. It also includes the results of the experiments.

**Chapter 5** contains the conclusion and discusses future work.

## Chapter 2

# Related Work

The main problem of intrusion detection in the cloud environment is the lack of public datasets for Cloud Intrusion Detection. Researchers around the world are developing various security techniques to detect malicious traffic for cloud computing environments. Most of the conducted experiments for intrusion detection uses datasets, such as DARPA 99, NSL-KDD and KDD99, which were designed or simulated based on conventional network architectures [1][2][3].

Gerhard et al. [4] presented a new method for detecting anomalies in traffic using a flow analysis based on the K-means algorithm. Training data that contains unsigned stream entries are divided into clusters of legitimate and malicious traffic. The corresponding cluster centroids are used as patterns to effectively detect anomalies in the analyzed traffic flow. The authors indicated that the application of the algorithm separately for different services (using different protocols and port numbers) increases the detection accuracy.

Lane and Brodley [5] examined user behavior by analyzing user profiles and their activity rather than classifying attack types. They inspect the sequence of command lines of the user and compare sequences by measuring the similarity of command line traces. User command data were collected from five people in the Purdue MILLENIUM lab. Then, for each user profile instance selection algorithm was applied to find the differences in behavior.

Zolotukhin et al. [6] analyzed high-speed network traffic through the application of anomaly-based detection approach to network packet statistics to detect DDoS attacks. The algorithm is based on filtering noise in data and clustering to detect malicious traffic. Models were trained on data obtained from a real computer environment. The proposed online training scheme allows one to reconstruct this model

whenever a new portion of network traffic for analysis is available. The authors concluded that the proposed model allows detecting all malicious streams with a very small number of false positives.

Yao et al. [7] proposed an intrusion detection system using enhanced Support Vector Machine based on weighted kernel. The work was evaluated using the KDD Cup 1999 and UNM datasets. However, the work does not consider the possibility of dynamic changes in traffic, which is important in cloud environment.

Moradi et al. [8] proposed an intrusion detection system using multilayer perceptron neural network model. A multilayer perceptron (MLP) is a neural network consisting of several hidden layers of computational neurons between the input and output layers. The DARPA intrusion detection dataset was used for the experiments. The evaluation yielded accuracy of 91 % with two hidden layers of neurons and 87 % with one hidden layer of neurons.

Idhammad et al. [9] proposed a detection model for HTTP DDoS attacks by analyzing the entropy of incoming network traffic. The CIDDS-001 public dataset was used for the experimental evaluation, yielding accuracy of 99.54 %.

Brooks et al. [10] presented an intrusion detection model that computes statistics on network traffic, such as entropy of incoming packet header fields, for instance, source IP addresses or type of protocol. It calculates the statistical characteristic observed and triggers an alarm when there is an extreme deviation.

# Chapter 3

## 3.1 Dataset

The ISOT cloud intrusion detection dataset was cloud instances hosted on Westgrid, one of Compute Canada cloud data centers. It contains various types of attacks such as denial of service attacks, masquerade attacks, and so on. The ISOT Cloud Intrusion Dataset involves normal activity data and a wide variety of attack vectors, collected in two phases and over several months for the Virtual Machine instances, and several days and time slots for the Hypervisors and the Network traffics. The dataset contains also various types of data including network traffic, system call traces, CPU and disk utilizations, etc. In the project, we consider only the network packets for the first day. Figure 3.2 depicts a sample attack scenario for day 1 as described in PhD thesis of Abdulaziz Aldribi [11].

## 3.2 Feature Extraction

For feature extraction, we used a freeware called Tranalyzer [12] that generates flow statistics from network data. Tranalyzer can process large pcap files and generate more than 130 statistical information about the flow. However, we reduced the number of features from 130 to 78, due to the fact that some features contain all 0s and missing values. The data was analyzed with the time window of 10 minutes, and frequency, entropy features were calculated in that specific time window. The features extracted from Tranalyzer can be found in Table 3.1.

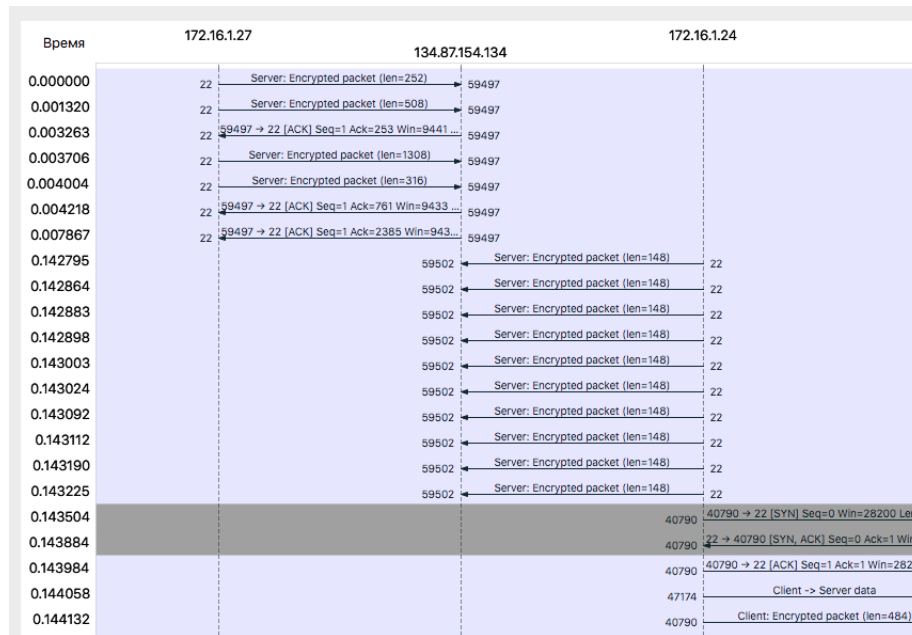


Figure 3.1: Example of network flow graph extracted from the dataset

#### Attack Scenarios for Day 1, 2016-12-09

Table A.6: Attack scenario steps for day 1

Day 1, 2016-12-09			
Seq	Attacks	Time	IPs
			<b>Src:</b> Attacker's machine <b>Des:</b> 1- 172.16.1.21 (VM9) (206.12.96.141) 2- 172.16.1.24 (VM8) (206.12.96.143) 3- 172.16.1.27 (VM10) (206.12.96.150)
1	Scanning subnet 206.12.96.140-150 from Attacker's machine	09:37am	
2	Dictionary attack from Attacker's machine against VM8	09:41am	<b>Src:</b> Attacker's machine <b>Des:</b> 172.16.1.24 (VM8)

Figure 3.2: Sample attack scenario for Day 1 [11]

### 3.3 Feature Model

We use a subset of the features proposed by Abdulaziz Aldribi in his PhD thesis [11]. The proposed features include three types of statistical features, namely, frequency features, load features, and entropy features. The features are defined for packets flows. Each flow consists of a group packets exchanged by cloud instances over some predefined time window. Frequency feature information takes into account the fre-

Feature	Description
Duration	flow duration
numPktsSnt	number of sent packets
numPktsRcvd	number of received packets
minPktSize	minimum packet size
maxPktSize	maximum packet size
avePktSize	average packet size
tcpMinWinSz	TCP minimum effective window size
tcpRTTackTripAve	TCP Ack Trip Average
tcpMSS	TCP Maximum Segment Length
pktAsm	Packet stream asymmetry
connSrc	Number of connections from source IP to different hosts

Table 3.1: The list of flow statistics computed using Tranalyzer

quency distribution from network flows. We computed the frequency of packets from a specific source to specific destinations, from all source to all destination, a maximum number of packets occurred in the flow, and so on.

The frequency, entropy, and load features calculated based on the thesis work of Aldribi [11] are defined as follows:

$$f_{s,sp,i,ip}^{in}(t) = \frac{|F_{in}^{t,\delta t}(s, sp, i, ip)|}{\delta t} \quad (1.1)$$

$$f_{s,i,ip}^{in}(t) = \frac{|\sum_{sp} F_{in}^{t,\delta t}(s, sp, i, ip)|}{\delta t} \quad (1.2)$$

$$f_{s,sp,i}^{in}(t) = \frac{|\sum_{ip} F_{in}^{t,\delta t}(s, sp, i, ip)|}{\delta t} \quad (1.3)$$

$$f_{s,i}^{in}(t) = \frac{|\sum_{sp,ip} F_{in}^{t,\delta t}(s, sp, i, ip)|}{\delta t} \quad (1.4)$$

$$f_{sp,i}^{in}(t) = \frac{|\sum_{s,ip} F_{in}^{t,\delta t}(s, sp, i, ip)|}{\delta t} \quad (1.5)$$

$$f_{i,ip}^{in}(t) = \frac{|\sum_{s,sp} F_{in}^{t,\delta t}(s, sp, i, ip)|}{\delta t} \quad (1.6)$$



$$f_i^{in}(t) = \frac{|F_{in}^{t,\delta t}(i)|}{\delta t} \quad (1.7)$$

Entropy analysis is used in intrusion detection to form a statistical criteria that is valuable information and to improve the performance of proposed machine learning methods.

$$H_i(f_{s,i}^{in}, t) = - \sum_s \left( \frac{f_{s,i}^{in}}{\sum_{s,i} f_{s,i}^{in}} \right) * \log_2 \left( \frac{f_{s,i}^{in}}{\sum_s f_{s,i}^{in}} \right)$$

$$H_i(f_{sp,i}^{in}, t) = - \sum_{sp} \left( \frac{f_{sp,i}^{in}}{\sum_{sp,i} f_{sp,i}^{in}} \right) * \log_2 \left( \frac{f_{sp,i}^{in}}{\sum_{sp} f_{sp,i}^{in}} \right)$$

$$H_i(f_{i,ip}^{in}, t) = - \sum_{ip} \left( \frac{f_{i,ip}^{in}}{\sum_{i,ip} f_{i,ip}^{in}} \right) * \log_2 \left( \frac{f_{i,ip}^{in}}{\sum_{ip} f_{i,ip}^{in}} \right)$$

The load features are computed for a network flow, by dividing the total number of incoming packets by the total number of outgoing packets for a given cloud instance as follows:

$$L_j(t) = \frac{f_j^{in}(t)}{f_{j'}^{out}(t)}$$

where j and j' defined as subscripts of the corresponding in and out frequency features.

# Chapter 4

## 4.1 Feature Selection

This section describes the supervised classification of machine learning methods. In the classification, the data set usually includes a large number of features that may be relevant, irrelevant or redundant. Redundant and irrelevant traits are not suitable for classification, and they may even reduce effectiveness. To find the features that are carrying the most significant information, it is proposed to use the Boruta algorithm in R language. Boruta is an all-relevant selection wrapper algorithm that can work with any classification method that outputs variable significance measure [13]. The method produces a search for relevant features by comparing the importance of original attributes with randomly attainable importance, estimating using their permuted copies, and gradually eliminating irrelevant features to stabilize the test. The Boruta algorithm shuffles the data and finds the features that have higher means. The features with the level of importance can be found in Figure 4.1. According to the result we can see that the features with the highest important include flowInd, maxIAT, aveIAT, ipMaxTTL, f.i.ip.d.dp, f.i.ip.d, connSip, connF, f.s.sp.i.ip, f.sp.i, tcpRTTSseqAA, entropy, and the load features. The Figures 4.2, 4.3 and 4.4 show the correlation of features to better interpret the data and understand the relationship between variables. The correlation is referred to as the statistical relationship between two variables. Discovering and quantifying the degree to which variables are dependent on each other in the dataset is important. This knowledge can help to better prepare your data for the machine learning algorithms. A correlation could be positive, meaning that both variables are moving in the same direction, or negative, meaning that the values of the other variables decrease when the value of one variable increases. In figure 4.3, the variable the number of packets flowing from source IP

to destination IP and variable the number of packets flowing from destination IP to source IP are positively correlated. However, in Figure 4.4, the variable numBytesRTAggr (The number of received and transmitted bytes) and variable minPktSize (Minimum layer 3 packet size) are negatively correlated.

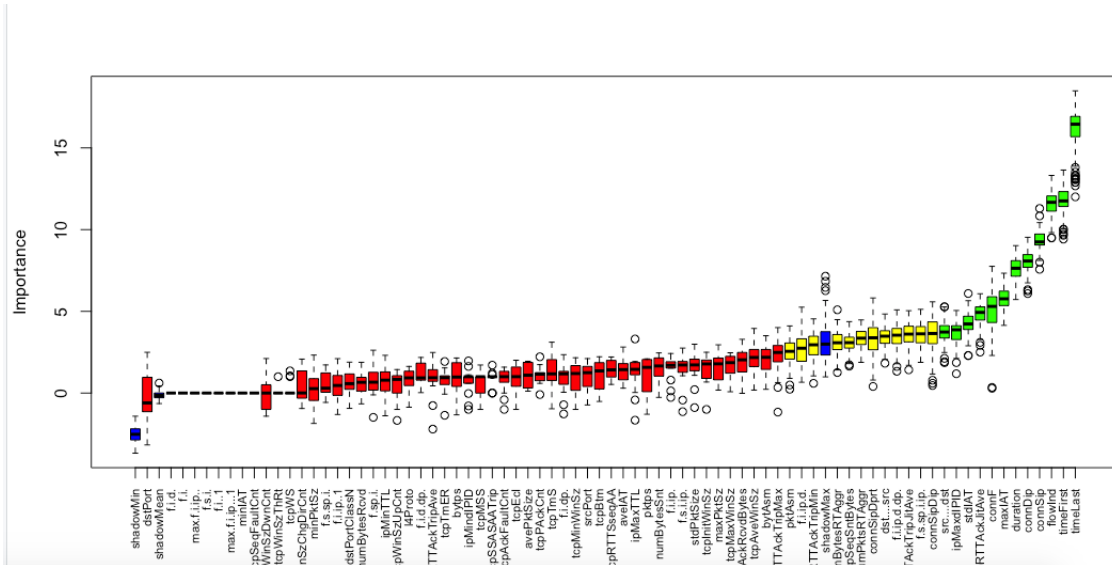


Figure 4.1: Feature importance chart obtained using Boruta algorithm

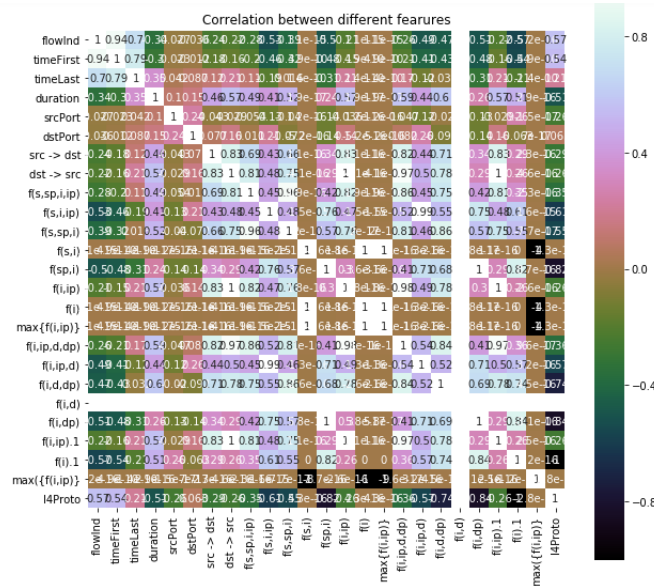


Figure 4.2: Correlation matrix between all features

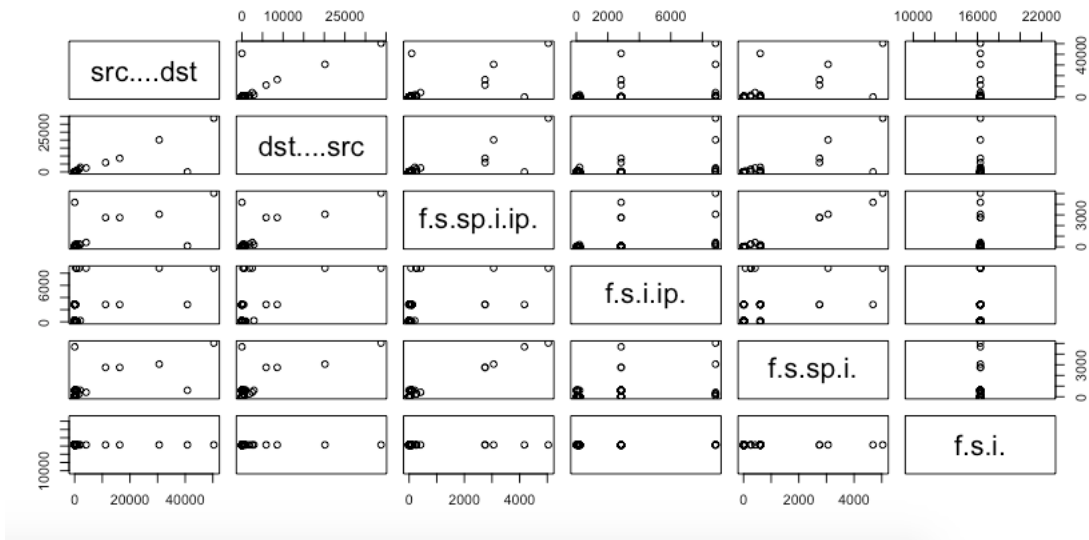


Figure 4.3: Correlation of first set of feature pairs

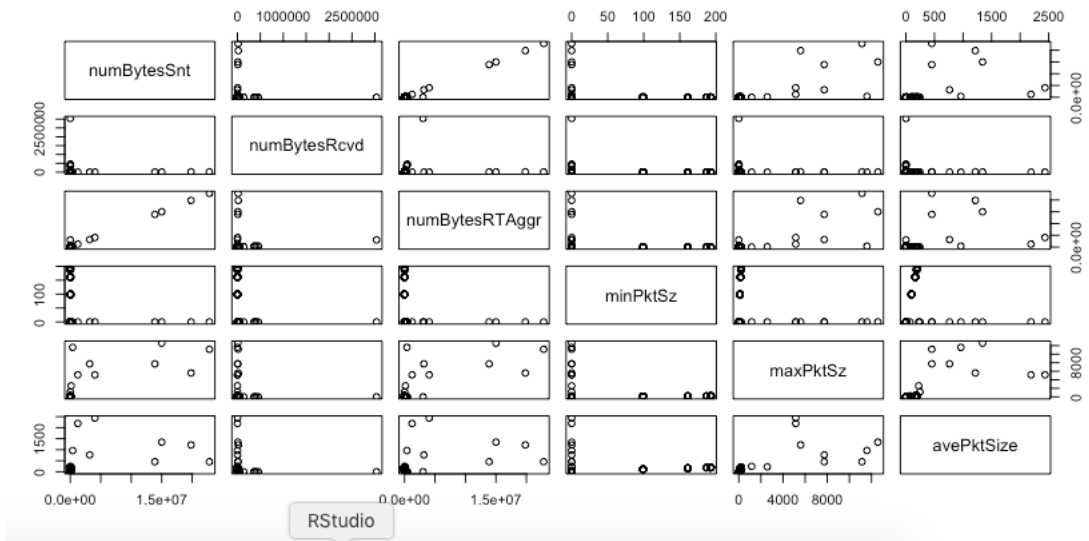


Figure 4.4: Correlation of second set of feature pairs

## 4.2 Classification Algorithms

We study in this project three different classification algorithms, including, naive bayes, neural networks, and logistic regression.

**Naive Bayes.** Naive Bayes is the simplest version of Bayesian networks used in classification problems [14]. The Naive Bayes algorithm is constructed on Bayes theorem, which first calculates the conditional probability of each feature for a label

and then applies the theorem of Bayes to obtain the probability. The Naive Bayes classifier operates in a way that features have different probabilities in malicious and normal network traffic. The main idea is to calculate the feature probability value for each class (normal/malicious) and choose the one that turned out to be the maximum. In order to be effective in classifying the data, the Bayesian classifier requires a lot of training data. Figure 4.5 shows the code snippet of implementation of Naive Bayes classifier.

```

from sklearn.naive_bayes import GaussianNB
import pandas as pd
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn import preprocessing

df = pd.read_csv('/Users/aika/Desktop/merged_dataset.csv')

X = df.iloc[:, :-1].values
from sklearn.preprocessing import Imputer
trainDataVecs = Imputer().fit_transform(X)
print(trainDataVecs)
Y = df.iloc[:,4].values
print(Y)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(trainDataVecs,Y,test_size=0.3, random_state=109)

from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
model = gnb.fit(X_train,y_train)
preds = gnb.predict(X_test)
print(preds)
from sklearn.metrics import accuracy_score

print(accuracy_score(y_test, preds))

```

Figure 4.5: The Naive Bayes code snippet

**Neural Networks.** Neural networks have arisen from research in the field of artificial intelligence, namely, from attempts to reproduce the ability of biological nervous systems to learn and correct errors, simulating a low-level brain structure [15]. A neural network is a learning system that consists of many interacting processors (neurons). From a mathematical point of view, a neural network is a function that has a certain number of input signals and one or several output signals. We use a Multilayer Perceptron for training. Inputs and outputs are represented by real numbers. The output will be 0 for benign and 1 for malicious. For our experiment, we use 30 hidden layers. Each neuron consists of weights. When passing through a neuron, all input signals are multiplied by their respective weights, then the resulting value is fed to the input of the neuron activation function, which in turn determines the output signal. Neural networks allow to identify typical characteristics of network traffic, learn the pattern of normal traffic and identify statistically significant deviations from the established mode of operation. Figure 4.6 shows a code snippet for the neural network implementation.

```

from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(30,30,30), max_iter=1000)
mlp.fit(X_train, y_train)
predicti = mlp.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
print("Accuracy", metrics.accuracy_score(y_test, predicti))

print(confusion_matrix(y_test,predicti))
print(classification_report(y_test,predicti))

```

Figure 4.6: The Neural Network code snippet

**Logistic Regression.** The Logistic Regression is an algorithm used to predict the likelihood of a categorical dependent variable. The logistic regression classifier helps us to solve situations where the output can only take two values, 1 or 0, with the help of sigmoid function. Unlike conventional regression and classification tasks, the logistic regression method does not predict the value of a numeric variable based on a sample of initial values. Instead, the value of a function is the probability that a given initial value belongs to a particular class. Figure 4.7 shows a code snippet for logistic regression.

```

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(trainDataVecs,Y,test_size=0.3, random_state=109)

from sklearn.metrics import classification_report, confusion_matrix

clf = LogisticRegression(penalty='l2', C=0.1)
#X_test.fillna(X_train.mean(), inplace=True)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Accuracy", metrics.accuracy_score(y_test, y_pred))
y_pred_proba = clf.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()

```

Figure 4.7: The Logistic Regression

### 4.3 Evaluation Metrics

In our study, we use several metrics of evaluation: Accuracy, Precision, and Recall. Accuracy is the percentage of predictions that are true. Precision tells us how many of the predictions for a certain class were correct. Recall, on the other hand, tells us how many instances of a specific class are correctly predicted in the test set. Related metrics include also the true positive rate (TPR) and false positive rate (FPR). FPR measures the rate at which the detector classifies benign traffic as malicious. On

the other hand, the false negative rate (FNR) is obtained by computing the ratio between the number of malicious events missed by the detector and the total number of malicious events. Equations (1.8) and (1.9) define FPR and FNR as follows:

$$FPR = \frac{\textit{number of false positives}}{\textit{number of false positives} + \textit{number of true negatives}} \quad (1.8)$$

$$FNR = \frac{\textit{number of false negatives}}{\textit{number of false negatives} + \textit{number of true positives}} \quad (1.9)$$

Therefore, the true positive rate (TPR) and true negative rate (TNR) can be defined as:

$$TPR = 1 - FNR \quad \textit{and} \quad TNR = 1 - FPR$$

## 4.4 Evaluation Methods

For training purposes, we used our constructed subset of the ISOT-CID network dataset. A free software machine learning library called Scikit-learn of the Python programming language was used for the implementation of machine learning algorithms. The data set was randomly split into training and testing data using a scikit-learn function called `test train split()`. 70 % of data was taken for training and the remaining 30 % of data was taken for testing. The same splitting principle was applied to all three machine learning models. The test data set was used to predict benign or malicious network traffic data. We used the label of 1 for malicious and 0 for benign data to distinguish prediction. In all machine learning models test train split function were implemented.

## 4.5 Evaluation Results

Using the newly constructed data set, we conducted the evaluation for the 3 classifiers considered in the project, namely, Naive Bayes, Logistic Regression and Neural Network machine learning techniques.

In the first step of the evaluation, all the 36 attributes extracted from Tranalyzer

tool were selected for the experiment. Neural Network with 36 attributes achieved the highest accuracy of 69.7 %. Logistic Regression and Naive Bayes achieved accuracy of 60.6 % and 63.6 % respectively.

In the second step, in order to reduce the complexity of preprocessing, 15 attributes were selected from boruta feature selection algorithm for the experiment. Neural Network with 15 attributes achieved the highest accuracy of 81.7 % and Logistic Regression, Naive Bayes achieved the accuracy of 81.4 %, 80.6 %, respectively. The conducted experiments show that by selecting features that have higher importance is better than considering all attributes. Table 4.2 shows the classification results that contained only the selected features that were extracted above.

<i>Truth</i>	<i>Predicted</i>	Logistic Regression	Naive Bayes	MLP
T	T	20	21	23
T	F	3	2	0
F	T	10	10	10
F	F	0	0	0

<i>AccuracyMetric</i>	Logistic Regression	Naive Bayes	MLP
Precision	0.46	0.47	0.49
Recall	0.61	0.64	0.70
Accuracy	60.6%	63.6%	69.7%

Table 4.1: Truth Tables and Accuracy Measures for when using all features.

Based on the results, we can conclude that the use of machine learning methods in network attack detection systems is quite effective. The resulting experimental data prove this statement. The conducted experiments shows that by selecting features that higher importance are better than considering all attributes.



<i>Truth</i>	<i>Predicted</i>	Logistic Regression	Naive Bayes	MLP
T	T	5	0	1
T	F	69	74	73
F	T	0	8	0
F	F	316	316	324

<i>AccuracyMetric</i>	Logistic Regression	Naive Bayes	MLP
Precision	0.66	0.74	0.82
Recall	0.81	0.81	0.85
Accuracy	81.4%	80.6%	81.7%

Table 4.2: Truth Tables and Accuracy Measures for using selected features.

## Chapter 5

# Conclusion and Future Work

In this project, we have conducted a preliminary study toward developing cloud anomaly detection models using machine learning. The study was conducted on the Naive Bayes, Logistic Regression and Neural Network classification methods for detecting anomalies in cloud network traffic. The experiments were carried out on a small subset of the ISOT-CID network data, which includes legitimate and malicious activity. We presented the results of the different machine learning classifiers and the outcome of this preliminary study is that of Neural Network had the best result. The future work could be the possibility of using hybrid machine learning methods, for instance, by combining all three methods, as well as different feature selection algorithms to improve the performance of the models.

## REFERENCES

- [1] S. Revathi, A. Malathi. A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection, Vol. 2 Issue 12, December - 2013, ISSN: 2278-0181, August 1980.
- [2] Gaurav Meena, Ravi Raj Choudhary. A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA, 2017 International Conference on Computer, Communications and Electronics (Comptelix), ISBN: 978-1-5090-4709-3, 2017.
- [3] Preeti Aggarwal, Sudhir Kumar Sharma. Analysis of KDD Dataset Attributes - Class wise for Intrusion Detection, 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015), Volume 57, 2015, Pages 842-851.
- [4] Gerhard M, Georg Carle, Wilhelm Schickard. Traffic Anomaly Detection Using K-Means Clustering, April 2019.
- [5] T. Lane and C. Brodley, Temporal Sequence Learning and Data Reduction for Anomaly Detection, ACM Transactions on Information and System Security, vol. 2, 3, pp. 295-331, 1999.
- [6] Mikhail Zolotukhin, Timo Hämäläinen, Tero Kokkonen, Jarmo Siltanen. Increasing web service availability by detecting application-layer DDoS attacks in encrypted traffic, 23rd International Conference on Telecommunications (ICT), May 2016.
- [7] JingTao Yao, Songlun Zhao, and Lisa Fan, An Enhanced Support Vector Machine Model for Intrusion Detection, Rough Sets and Knowledge Technology, First International Conference, RSKT 2006, Chongqing, China, July 24-26, 2006, Proceedings (pp.538-543)
- [8] M.Mohadi. A Neural Network Based System for Intrusion Detection and Classification of Attacks, February 2014.
- [9] M. Idhammad, K.Afdel, M.Belouch. Detection System of HTTP DDoS Attacks in a Cloud Environment Based on Information Theoretic Entropy and Random Forest, June 2018
- [10] Illker Ozelik R. Brooks. Deceiving Entropy Based DoS Detection, Computers & Security. 48. 10.1016/j.cose.2014.10.013.
- [11] Abdulaziz Aldribi. Cloud intrusion detection based on change tracking and a new benchmark dataset, August 2018.
- [12] Tranalyzer. Lightweight open-source flow generator and packet analyzer for practitioners and researchers: <https://tranalyzer.com/>
- [13] Miron Bartosz Kurska, Witold Remigiusz Rudnicki. Wrapper Algorithm for All Relevant Feature Selection, <https://cran.r-project.org/web/packages/Boruta/Boruta.pdf>, July 2018.
- [14] Jiang Su, Harry Zhang, "Full Bayesian Network Classifiers"
- [15] Andrea Soltoggio, Kenneth O. Stanley, Sebastian Rusi. Born to Learn: the Inspiration, Progress, and Future of Evolved Plastic Artificial Neural Networks, August 2018