# Comparison of Supervised, Semi-supervised and Unsupervised Learning Methods in Network Intrusion Detection System (NIDS) Application

Nari S. Arunraj, Robert Hable and Michael Fernandes
Deggendorf Institute of Technology
Technology Campus Grafenau
Hauptstraße 3
D-94481 Grafenau, Germany
Email: nari.arunraj@th-deg.de

Karl Leidl and Michael Heigl
Deggendorf Institute of Technology
Institute for Protection of Industrial Technology
Dieter-Görlitz-Platz 1
D-94469 Deggendorf, Germany
Email: karl.leidl@th-deg.de

## ABSTRACT

With the emergence of the fourth industrial revolution (Industrie 4.0) of cyber physical systems, intrusion detection systems are highly necessary to detect industrial network attacks. Recently, the increase in application of specialized machine learning techniques is gaining critical attention in the intrusion detection community. A wide variety of learning techniques proposed for different network intrusion detection system (NIDS) problems can be roughly classified into three broad categories: supervised, semi-supervised and unsupervised. In this paper, a comparative study of selected learning methods from each of these three kinds is carried out. In order to assess these learning methods, they are subjected to investigate network traffic datasets from an Airplane Cabin Demonstrator. In addition to this, the imbalanced classes (normal and anomaly classes) that are present in the captured network traffic data is one of the most crucial issues to be taken into consideration. From this investigation, it has been identified that supervised learning methods (logistic and lasso logistic regression methods) perform better than other methods when historical data on former attacks are available. The results of this study have also showed that the performance of semi-supervised learning method (One class support vector machine) is comparatively better than unsupervised learning method (Isolation Forest) when historical data on former attacks are not available.

## KEYWORDS

Network Intrusion Detection System, Classification, Anomaly detection, Imbalanced classes

## INTRODUCTION

After the previous three industrial revolutions powered by steam engine, mass production with invention of electricity, and process automation with development of computers and information technology, the fourth industrial revolution (Industrie 4.0) of cyber physical systems is emerging currently (Meshram and Haas, 2017). With the advent of fourth industrial revolution of cyber physical systems, the high networked production lines within companies and the connectivity even with external partners will increase over the coming years. Recent technological developments of information and communication technology (ICT), sensor technology, digitalization of manufacturing processes, Internet of Things (IoT), Big Data analytics and cloud computing enable a huge innovation in industrial production. These developments have created an integrated information grid which tightly connects systems and humans together, which further evolves into a big data environment in the industry  (Lee et al., 2014; Meshram and Haas, 2017). In these highly networked modern production lines, integrated embedded intelligence systems and software communicate with each other independently. Cloud-based planning systems calculate service needs and machine utilization. Plant operators monitor and control the system remotely, maintenance staff can access and change the configuration of the plant from anywhere (Lee et al., 2014; Meshram and Haas, 2017; Möller, 2016; Plattform Industrie 4.0, 2016). In order to evade damage and production stoppages, suitable measures are needed to prevent security incidents. The Industrie 4.0 applications in the near future will be extended to other sectors where the network attack damages cause not only production and property loss (financial loss), but also human and environmental loss, e.g. nuclear, aviation, defense, spacecraft etc. If the probability of network attacks and their resulting damages increase, the security risk also increases. One aspect of security is the monitoring of network traffic to detect anomalies that might be caused by network attacks.

Network intrusion detection systems (NIDS) refer to detection of intrusions in network data (malicious activity in network). Intrusions typically occur as anomalous patterns though certain techniques model the data in a sequential manner and detect anomalous subsequences. The primary reason for these anomalies is attacks launched by outside attackers who want to gain unauthorized access to the network to steal information or to disrupt or harm the network. NIDS can be broadly classified based on their style of detection as follows (Sommer and Paxson, 2010; Zhang et al., 2016): (i) *misuse-detection systems* monitor activity with precise descriptions of known malicious behavior and (ii) *anomaly-detection systems* form a view of normal activity and identify deviations from that profile. The anomaly detection systems are *not* targeting to identify malicious behavior but just report what has not been seen before, whether normal or not. However, the important objective of an intrusion detection system is to find attacks. If a detector has such a gap, then it does not meet its operational expectations. Network traffic often exhibits more diversity than people intuitively expect, which leads to misconceptions about what and how anomaly detection technology can realistically achieve in operational environments. One way to reduce this diversity of network traffic is to employ *aggregation*. The highly varying network traffic properties over small to medium time intervals tend to have greater stability when observed over longer time intervals (Sommer and Paxson, 2010). Anomaly detection has extensive applications in areas such as fraud detection for credit cards, intrusion detection for cyber security, and military surveillance for enemy activities.

Anomaly detection is usually done by one of the following (Mok et al., 2010):

- Threshold detection, i.e., detecting abnormal activity on the server or network.
- Statistical learning methods with historical values.
- Rule-based methods with expert systems (e.g. association rule mining).
- Non-linear algorithms (e.g. Artificial Neural Networks or Genetic Algorithms).

The first problem about anomaly detection systems, in particular NIDS, is the excessive number of reported false positives. Although the anomaly detection systems do not necessarily make *more* mistakes , the high cost associated with each false negatives (if exactly predicted anomalies are assumed as true positives, then anomalies that are categorized along with normal data are false negatives) is often important to consider. Therefore, reducing false negatives must be a significant task for any anomaly detection system (Garrette, 2006; Leung and Leckie, 2005; Sommer and Paxson, 2010). In addition, the characteristics of anomaly detection systems that are not well aligned with the requirements of machine-learning. These include: *(i)* a very high cost of errors; *(ii)* lack of training data; *(iii)* a semantic gap between results and their operational interpretation; *(iv)* variability in input data; and *(v)* fundamental difficulties for conducting sound evaluation (Sommer and Paxson, 2010).

The machine-learning algorithms perform much better at finding similarities than at identifying activity that does not belong there, i.e., a *classification* problem, instead of identifying outliers as required by an anomaly detection system. According to this view, the anomaly detection is a classification problem of two classes, "normal" and "anomaly", and the objective is to determine which of the two classes more likely matches an observation (Bhuyan et al., 2014; Chandola et al., 2009; Dokas et al., 2002). However, a basic rule of machine learning is that one needs to train a system with samples of both classes, and the amount of data instances found in the training set for *each* class should be balanced. On the other hand, the anomaly detection tries to find novel attacks with training only possible on normal traffic and not on the attacks of interest. However, it is not practically beneficial for real world problems. Because the assumption of training on normal traffic which covers all possible cases is not certain.

In this paper, a comparative study of supervised, semi-supervised, and unsupervised anomaly intrusion detection methods is carried out to investigate their performance in detecting anomalies. Logistic and lasso logistic regression are used for supervised anomaly intrusion detection. One class support vector machine (OCSVM) is used for semi-supervised anomaly intrusion detection. Isolation Forest is used for supervised anomaly intrusion detection. The proposed approach is evaluated using network intrusion dataset generated from Airplane Cabin Demonstrator. Our experimental results from the proposed anomaly detection techniques are compared.

The paper is organized as follows. In Section 2, the related work are discussed as literature. The methods and performance measures are presented in Section 3. In Section 4, the data are described in experimental setup. In Section 5, the results are discussed. Finally, the paper is concluded in Section 6.

## LITERATURE REVIEW

In a well-known classification setup, training data is used to train a model and test data measures performance. However, there are multiple setups possible in anomaly detection with reference to the labels available in the dataset (training and testing). Based on the availability of labels, the anomaly detection methods can be classified into supervised, semi supervised and unsupervised methods.

### Supervised Anomaly Detection

Techniques trained in supervised mode assume the availability of a training data set which has labeled instances for a normal as well as an anomaly class. The datasets of all the anomaly scenarios of all the devices are combined. The data are labeled for their scenarios and devices. The anomalies are also labeled. Then, the training and testing datasets have to be selected for cross validation. A supervised learning method is used to model the training dataset. A typical approach in such cases is to build a predictive model for normal vs. anomaly classes (Görnitz and Rieck, 2013). Then, the testing dataset is used to evaluate the model.

Logistic regression and Decision Trees are the most commonly used supervised learning algorithms in anomaly detection due to its simplicity, high detection accuracy and fast adaptation (Hastie et al., 2009).

Another well-known supervised learning technique used in anomaly detection is Naïve Bayes classifiers. Because Naïve Bayes assumes the conditional independence of data features, which is often not the case for intrusion detection, correlated features may degrade its performance (Jha and Ragha, 2013). Besides, Support Vector Machine (SVM) is also a well-known anomaly detection system which is capable of real-time detection, deal with large dimensionality of data (Chandola et al., 2009; Jha and Ragha, 2013; Manandhar, 2014).

There are two major practical challenges that arise in supervised anomaly detection. First, the anomalous instances are far fewer compared to the normal instances in the training dataset. Issues that arise due to imbalanced class distributions have been addressed in the data mining and machine learning literature (Bhuyan et al., 2014; Chandola et al., 2009; Dokas et al., 2002; Garrette, 2006; Goldstein et al., 2016). Second, obtaining accurate and representative labels, especially for the anomaly class is usually challenging. For most of the applications, the anomalies are not known in advance or may occur spontaneously as novelties during the test phase. Except for these two issues, the supervised anomaly detection problem is similar to building predictive models.

*Balancing Imbalanced Classes*

In a classification approach of anomaly detection, the classification algorithm attempts to generate a classifier which separates two classes. If the classes are imbalanced, the anomalies are highly outnumbered by normal data. In such cases, huge population of normal cases are sorted out to find anomaly cases. When a conventional algorithm is used in this situation, the algorithms are often biased towards the majority class because their loss functions attempt to optimize quantities such as error rate, not taking the data distribution into consideration. The minority cases are treated as outliers of majority class and ignored. The algorithm generates a classifier that classifies every example as the majority class. There are several ways in order to cope up with imbalanced classes as follows (Fawcett, 2016):

i. Balancing the training set
   a. Oversampling the minority class
   b. Undersampling the majority class
   c. Synthesizing the minority class
ii. Adjusting the algorithm
   a. Adjusting the decision threshold
   b. Adjusting the class weights
   c. Identify an algorithm to perform well on imbalanced data

The undersampling throws away data and results in loss of information. The oversampling results in duplicating the minority instances that makes variables appear to have lower variance than they do. The negative consequence of duplication in oversampling is that it also duplicates the number of errors. The over- and undersampling selects examples randomly to adjust their proportions. In order to overcome the drawbacks of undersamlping and oversampling, a synthetic minority oversampling technique (SMOTE) was developed to synthetically generate the minority class (Chawla et al., 2002). Instead of balancing the training set, the parameters in the selected algorithm can also be adjusted to overcome the problems of imbalanced classes. When the classes are imbalanced, the decision threshold or cut-off should not be assigned with the default 0.5. Instead, the decision threshold can be increased according to the proportion of minority instances in total (Fawcett, 2016). Otherwise, the class weights can also be assigned based on the proportion of imbalanced classes. OCSVM, Isolation Forest, Box Drawing algorithm are some of the machine learning methods which are frequently used for anomaly detection with imbalanced classes (Goh and Rudin, 2014; Liu et al., 2012; Scholkopf et al., 2000).

### Semi-supervised Anomaly Detection

This typical approach used in such techniques is to build a model for the class corresponding to normal behavior, and use the model to identify anomalies in the test data. A limited set of anomaly detection techniques exists that assume availability of only the anomaly instances for training. Such techniques are not commonly used, primarily because it is difficult to obtain a training data set which covers every possible anomalous behavior that can occur in the data. The semi-supervised methods are more widely applicable when there is no historical accident or attack data. Even if the data are available, they are not repeating in future. For example, in space craft fault detection, an anomaly scenario would signify an accident or failure, which is not easy to model. Therefore, it is also called as novelty detection, instead of anomaly detection. One class support vector machine and support vector data description are widely used in case of semi-supervised anomaly detection and novelty detection (Heller et al., 2003; Scholkopf et al., 2000).

### Unsupervised Anomaly Detection

Techniques that operate in unsupervised mode do not require training data, and thus are most widely applicable. The datasets of all the anomaly scenarios of all the devices are combined. The data are labeled for their scenarios and devices. Then, the training and testing datasets have to be selected for cross validation. An unsupervised learning technique is used to model the combined training dataset (Hastie et al., 2009). The techniques in this category make the implicit assumption that normal instances are far more frequent than anomalies in the test data. If this assumption is not true then such techniques suffer from high false alarm rate. Many semi-supervised techniques can be adapted to operate in an unsupervised mode by using a sample of the unlabeled data set as training data. Such adaptation assumes that the test data contains very few anomalies and the model learnt during training is robust to these few anomalies. Isolation Forest is one of the new entry in unsupervised anomaly detection methods and is highly suggested to be used in case of imbalanced classes (Liu et al., 2012).

*Clustering*
The underlying assumption in this approach is that if the data are clustered, the normal data belongs to clusters while anomalies do not belong to any cluster or belong to small clusters. Then to detect anomalies the data are clustered, and the centroids and the density of each cluster are calculated. When there is a test data point, the distance between the new data point and the known large clusters is calculated. If the distance is too far, then it is an anomaly (Chandola et al., 2009; Laskov et al., 2005; Leung and Leckie, 2005).

*Nearest neighbor*
The underlying assumption is that new anomalies are closer to known anomalies. This can be implemented by using distance to k-anomalies or using the relative density of other anomalies near the new data point (Gogoi et al., 2011; Leung and Leckie, 2005).

## METHODS FOR ANOMALY DETECTION

In the following subsections, we briefly present the classification methods that are used for NIDS in our comparative study.

### Logistic and Lasso Logistic Regression

Logistic regression is the most widely used statistical model in many fields for binary data (0/1 response) prediction. It has been widely applied due to its simplicity and great interpretability. As a member of generalized linear models it is based on the logit function.

Let us assume that the predicted variable and predictors are denoted by *Y* and *X* respectively and the two classes of interest such as normal and anomaly classes are denoted by 1 and 0 respectively. We wish to model the conditional probability that the outcome Y is 1, given that the input variables are X. The conditional probability is denoted by p(*Y=1*|*X*) which we will abbreviate as p(*X*) since we know we are referring to the positive outcome *Y* = 1. The probability of class membership lies between 0 and 1. The function assumed in logistic regression is (Hastie et al., 2009):

$$\log\left[\frac{p(X)}{1-p(X)}\right] = \beta_0 + \beta_1 X_1 \ldots + \beta_i X_i \qquad (1)$$

One way to frame this problem is to maximize the product of these probabilities, often referred to as the likelihood and this approach is called maximum likelihood estimation:

$$log\left[\prod_{i:Y_i=1} p(X_i).\prod_{i:Y_j=0}\left(1-p(X_j)\right)\right] \qquad (2)$$

where $\prod$ represents the products over i and j, which run over the 1 and 0 classed points respectively. Instead of maximizing, the equation (2) can be also written as

$$L = -log\left[\prod_{i:Y_i=1} p(X_i).\prod_{i:Y_j=0}\left(1-p(X_j)\right)\right] \quad (3)$$

to minimize the negative log likelihood.

Lasso regularization works by adding a *penalty term* to the log likelihood function. In the case of lasso regression, the penalty term is $|\beta_1|$ and is minimized as (Hastie et al., 2009):

$$L + \lambda \sum |\beta_1| \qquad (4)$$

where $\lambda$ is a parameter which is usually selected in such a way that the resulting model minimizes the out of sample error.

### OCSVM

The goal in anomaly detection is to detect anomalies by finding a concise description of the normal data, so that deviating observations become outliers. Let $x_i (i = 1,2,\ldots,n)$ denote the training examples (normal or non-anomalous instances). $\phi: X \to H$ is a kernel map which transforms the data into an inner product space $H$. The problem of separating the data set from the origin and maximizing the distance from the hyperplane to the origin, is essentially the problem of optimizing the following quadratic programming (Berwick, 1990; Manandhar, 2014; Scholkopf et al., 2000; Vlasveld, 2013; Zhang et al., 2016):

$$min \frac{1}{2}\|w\|^2 + \frac{1}{vl}\sum_{i=1}^{n} \xi_i - \rho$$

$$s.t. (w.\phi(x_i)) \geq \rho - \xi_i \qquad (5)$$

$$for\ all\ i = 1,2,\ldots,n, \qquad \xi_i \geq 0$$

where the parameter $\nu$ also serves as an estimate of the ratio between anomalies and normal data. In the above formula, the parameter $\nu$ is an upper bound on the fraction of outliers and is a lower bound of the fraction of support vectors relative to the number of training examples. This method thus creates a hyperplane characterized by $w$ and $\rho$ which has maximal distance from the origin and, separates all the data points from the origin. The decision function for an data point is (Vlasveld, 2013):

$$f(x) = sign\left(\left(w.\phi(x_i)\right) - \rho\right) \qquad (6)$$

$$sign\left(\sum_{i=1}^{n} \alpha_i K(x, x_i) - \rho\right)$$

where $\alpha_i$ are the Lagrange multipliers and $K(x, x_i)$ is the kernel function. If the kernel function is radial basis function, then it is $\exp\left(-\frac{\|x-x_i\|^2}{2\sigma^2}\right)$.

## Isolation Forest

The basic concept of Isolation Forest is to measure susceptibility or proneness of individual data instances to be isolated. (Liu et al., 2012; Ting, 2009). Since anomalies are few and far from normal instances and therefore they are more inclined to isolation. This random partitioning produces noticeable shorter paths for anomalies. The assumptions are: the fewer instances of anomalies result in a smaller number of partitions (shorter path length) and the instances with distinguishable attribute-values are more likely to be separated in early partitioning. Hence, when a forest of random trees collectively produce shorter path lengths for some particular points, then they are highly likely to be anomalies. The important input parameters to the Isolation Forest algorithm: are the subsampling size, the number of trees and the height of the tree.

According to Liu et al., (2012), the subsampling size has to be smaller so that the algorithm is faster and the detection accuracy is also better. The depth of the tree can be approximately calculated by $\log_2$ (number of data instances). Sub-sampling size ($\psi$) controls the training data size. There is no need to increase the sub-sampling size beyond the desired value because it increases processing time and memory size without any gain in detection accuracy (Fayet et al., 2017; Liu et al., 2012). According to Liu et al. (2012), the sub-sampling size of $2^8$ or 256 is enough to perform anomaly detection empirically. Number of tree ($T$) controls the ensemble size. According to Liu et al. (2012), the path lengths usually converge well before t = 100.

Let $x$ be a sample, $n$ the number of samples on which the Isolation Trees are built. Let $f$ be the function of Isolation Forest $f(t_1, t_2, t_2 \dots t_T)$. Let $h(t, x)$ be the number of edges of the Isolation Tree $t$ between the root and the leaf which isolates $x$. Let $c(n)$ be the average path length of unsuccessful search in a binary search tree. $c(n)$ estimates the average path length of an Isolation Tree. The anomaly score $s$ of an instance $x$ can be estimated as follows (Fayet et al., 2017; Liu et al., 2012):

$$s(x, f, n) = 2^{-\frac{\sum_{k=1}^{T} h(f_k, x)}{T * c(n)}} \qquad (7)$$

## Performance Measures

It is apparent that in this case overall classification accuracy is not sufficient as a standard performance measure. Metrics such as sensitivity, specificity, prevalence, detection rate, and positive prediction value have been used to understand the performance of the learning algorithm on the anomaly class. A confusion matrix as shown in Table 1 is typically used to evaluate performance of a machine learning algorithm.

*Table 1: Confusion matrix*

| | **Actual Non-anomalies** | **Actual Anomalies** |
|---|---|---|
| **Predicted Non-anomalies** | TN (D) | FN (C) |
| **Predicted Anomalies** | FP (B) | TP (A) |

In this study, it is considered that anomaly class is positive and normal (non-anomaly) class is negative. Therefore, True Negative (D) is the number of correctly predicted non-anomalies. True Positive (A) is the number of correctly predicted anomalies. False Negative (C) is the number of actual anomalies which are predicted as non-anomalies. False Positive (B) is the number of non-anomalies which are predicted as anomalies. The performance measures considered in this study can be calculated using the formulas as follows:

$$Sensitivity = \frac{A}{A + C} \qquad (8)$$

$$Specificity = \frac{D}{B + D} \qquad (9)$$

$$Prevalence = \frac{A + C}{A + B + C + D} \qquad (10)$$

$$Detection\ rate = \frac{A}{A + B + C + D} \qquad (11)$$

$$Positive\ Prediction\ Value = \frac{A}{A + B} \qquad (12)$$

## EXPERIMENTAL SETUP

### Data Description

This section discusses the types of available data. Anomaly detection methods are usually based on volume and features. This study focuses only on features of network traffic. The datasets of seven devices (D1, D2, D3, D4, D5, D6, and D7) placed on dedicated places within the network of an Airplane Cabin Network Demonstrator are collected. The logical network flow structure of the Airplane Cabin Network Demonstrator is shown in Figure 1. The main task of these measuring probes is to capture the ongoing traffic passing from mirroring ports of the network forwarding elements. Each dataset consists of a fixed set of basic features as shown in Table 2. There are eleven basic features (numerical feature - 2 and categorical features - 9). The categorical variables are converted to binary variables, according to their levels. The anomalies are generated in every dataset based on six anomaly scenarios as shown in Table 3.

Research on imbalanced classes often assumes that the minority class is around 10 – 20% (Fawcett, 2016; Goh and Rudin, 2014). However, in reality, datasets are far more imbalanced (less than 2%). Therefore, in this study, the anomalies are generated around 2 – 5% on an average in each dataset per device. For the supervised approach, the training dataset is labeled with anomalies. In the semi-supervised approach, the normal dataset without any anomalies is used as a training set and the dataset labeled with anomalies is used for testing. In the unsupervised approach, the labels for anomalies are not used. In this experimental study, all the scenarios are considered on all the devices.

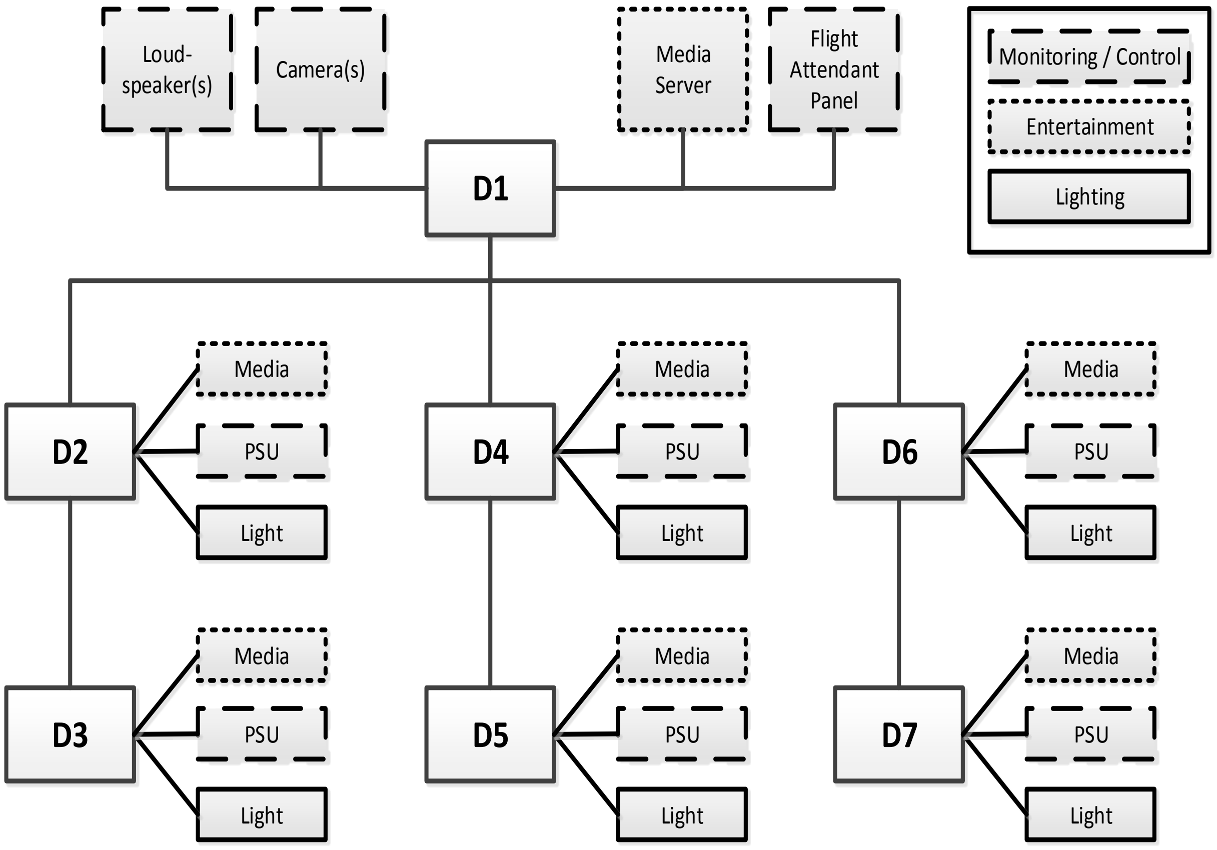

*Figure1: Logical network flows of the Airplane Cabin Network Demonstrator*

*Table 2: Basic features*

| **Variables** | **Type** |
|---|---|
| Time | Numerical |
| Source IP | Categorical |
| Source MAC address | Categorical |
| Source Port | Categorical |
| Destination IP | Categorical |
| Destination MAC address | Categorical |
| Destination Port | Categorical |
| Protocol | Categorical |
| Length | Numerical |
| Service ID | Categorical |
| Method ID | Categorical |

*Table 3: Anomaly scenarios and their details*

| No. | Scenarios | Protocol | Anomaly tools | Anomaly labels |
|---|---|---|---|---|
| 1 | MAC-Address-Manipulation | SOME/IP | Fuzzer | MAC-Address AB:CD:EF:12:34:56 |
| 2 | ARP-Spoofing | ARP | arpspoof | MAC-Address AB:CD:EF:12:34:56 |
| 3 | Unallowed access | HTTP | Browser | MAC-Address AB:CD:EF:12:34:56 |
| 4 | Light manipulation | UDP | scapy | MAC-Address AB:CD:EF:12:34:56 |
| 5 | Packet-Replay | SOME/IP, HTTP, UDP | tcprewrite, tcpreplay | VLAN-Tag 255 dez |
| 6 | Host availability | ICMP | ping | Protocol type ICMP |

*Table 4: Performance measures of logistic regression*

| Device | Prevalence | Detection rate | Sensitivity | Specificity | Positive prediction value |
|---|---|---|---|---|---|
| D1 | 0.01499 | 0.01499 | 1 | 0.93409 | 0.1877 |
| D2 | 0.00257 | 0.00257 | 1 | 0.98701 | 0.1655 |
| D3 | 0.00437 | 0.00437 | 1 | 0.97725 | 0.1616 |
| D4 | 0.00246 | 0.00246 | 1 | 0.9874 | 0.1638 |
| D5 | 0.00472 | 0.00472 | 1 | 0.97681 | 0.1698 |
| D6 | 0.00272 | 0.00272 | 1 | 0.98724 | 0.1762 |
| D7 | 0.00479 | 0.00479 | 1 | 0.97706 | 0.1676 |

*Table 5: Performance measures of lasso logistic regression*

| Device | Prevalence | Detection rate | Sensitivity | Specificity | Positive prediction value |
|---|---|---|---|---|---|
| D1 | 0.01514 | 0.01484 | 0.9801 | 0.93506 | 0.18349 |
| D2 | 0.00248 | 0.00247 | 0.9947 | 0.98710 | 0.17247 |
| D3 | 0.00217 | 0.00215 | 0.9900 | 0.97693 | 0.15853 |
| D4 | 0.0008 | 0.0008 | 1 | 0.98753 | 0.17891 |
| D5 | 0.00121 | 0.0012 | 0.9910 | 0.97686 | 0.17134 |
| D6 | 0.00047 | 0.00047 | 1 | 0.98745 | 0.16659 |
| D7 | 0.00085 | 0.00083 | 0.9758 | 0.97672 | 0.16316 |

*Table 6: Performance measures of OCSVM*

| Device | $\nu$ | $\gamma$ | Prevalence | Detection rate | Sensitivity | Specificity | Positive prediction value |
|---|---|---|---|---|---|---|---|
| D1 | 0.04 | 0.002 | 0.01485 | 0.01059 | 0.71356 | 0.90088 | 0.09788 |
| D2 | 0.05 | 0.001 | 0.00261 | 0.00186 | 0.71207 | 0.95944 | 0.04384 |
| D3 | 0.06 | 0.001 | 0.00445 | 0.00276 | 0.62046 | 0.93959 | 0.04386 |
| D4 | 0.05 | 0.001 | 0.00257 | 0.00183 | 0.71007 | 0.95914 | 0.04290 |
| D5 | 0.06 | 0.001 | 0.00466 | 0.00250 | 0.53769 | 0.95360 | 0.05353 |
| D6 | 0.06 | 0.001 | 0.00263 | 0.00201 | 0.76638 | 0.95726 | 0.04509 |
| D7 | 0.07 | 0.001 | 0.00478 | 0.00295 | 0.61799 | 0.95313 | 0.05950 |

*Table 7: Performance measures of Isolation Forest*

| Device | Prevalence | Detection rate | Sensitivity | Specificity | Positive prediction value |
|---|---|---|---|---|---|
| D1 | 0.00102 | 0.00038 | 0.37168 | 0.62881 | 0.00102 |
| D2 | 0.00024 | 0.00012 | 0.52000 | 0.58482 | 0.00030 |
| D3 | 0.00015 | 0.00004 | 0.27780 | 0.62990 | 0.00012 |
| D4 | 0.00012 | 0.00005 | 0.41670 | 0.53930 | 0.00011 |
| D5 | 0.00012 | 0.00006 | 0.53850 | 0.63820 | 0.00017 |
| D6 | 0.00026 | 0.00016 | 0.62963 | 0.56404 | 0.00038 |
| D7 | 0.00020 | 0.00013 | 0.65217 | 0.59073 | 0.00032 |

## RESULTS AND DISCUSSION

In this section, the supervised, semi-supervised, and unsupervised methods are applied on the datasets described in the previous section. In case of supervised learning, logistic and lasso logistic regression methods are used. For semi-supervised learning, OCSVM is used. For unsupervised learning, Isolation Forest is used. In supervised learning, the anomaly labels are assigned using all the anomaly scenarios and introduced as an independent variable (Y). In the application of logistic regression method, the training and the testing dataset are divided into the ratio of 5:1 respectively. In order to balance the imbalanced classes, weights are allotted for the independent variable. The weights are calculated from the proportion of anomaly instances to normal instances. The results are estimated as performance measures and presented in Table 4. The sensitivity of this method is impressive and shows that correctly predicted anomalies out of actual anomalies, which is 100%. The specificity of this method shows that 1 - 6% of non-anomalies are detected as anomalies. Similar to the logistic regression, in the lasso logistic regression application, the weights are allotted to the independent variable based on the proportion of anomaly instances to normal instances. The 5-fold cross validation is used to improve the results. In order to apply the lasso logistic regression, *glmnet* package from R is used (Friedman et al., 2017). The results are estimated as performance measures and presented in Table 5. When compared to the logistic regression, the lasso regression is faster during testing period without sacrificing much of its performance.

To apply the OCSVM method, *e1071* package from R is used (Meyer et al., 2017). In this method, the training and testing dataset are divided into the ratio of 5:1 respectively. The normal data instances without anomalies are first used as a training dataset to model. Then, the unlabeled data instances with anomalies are subjected to test the trained model. In OCSVM method, the parameters $\nu$ and $\gamma$ are selected for tuning to obtain the optimal solution. The initial value of parameter $\nu$ is the fraction of outliers. As the kernel function is radial basis function in OCSVM, the initial value of parameter $\gamma$ can be taken as the reciprocal of number of selected features in this study. A low value of $\gamma$ improves the smoothness of model, while the high value reduces the smoothness and tightly fits the training data. The selected parameters and performance measures are presented in Table 6. The sensitivity of this model shows that the method can correctly detect anomalies out of actual anomalies around 62 – 77%. Around 5 – 10% of non-anomalies are detected as anomalies.

To apply the Isolation Forest, *Isolation Forest* package from R is used (Liu et al., 2012; Ting, 2009). The training and testing dataset are divided into the ratio of 6:1 respectively. Like any other unsupervised learning method, Isolation Forest first uses the training dataset without label in order to model. Then, the unlabeled data instances with anomalies are used to test the trained model. The parameters such as number of trees and height of tree are selected by tuning in order to obtain the optimal solution. In this study, the parameter settings such as number of trees, height or depth of tree are 100 and 8 respectively. The sub-sampling size is taken as one third of the total data instances from the training dataset. The performance measures are presented in Table 7. When compared to the OCSVM, the performance of Isolation Forest is poor. However, it has very low computational time during test phase. The complexity of Isolation Forest increases with the increase in number of categorical variables and their levels. According to the sensitivity, this method can detect anomalies out of actual anomalies around 30 – 65%. More than 40% of non-anomalies are predicted as anomalies.

The performance of supervised learning methods such as logistic and lasso logistic regressions is very good. In case of supervised learning methods, the accuracy of detecting TP is almost 100% without sacrificing the accuracy of detecting TN. However, the provision of anomaly labels in the training dataset is highly impractical. Therefore, the semi-supervised and unsupervised learning methods are more reasonable in practice. Moreover, the main challenge of anomaly intrusion detection is to minimize false negatives when the attacks are novel and/or unknown. In this study, the OCSVM method fulfills the above mentioned task and is also effective in reducing false negative rate with a desirable false positive rate. On the other hand, the Isolation Forest is not effective in detecting both TN and TP, when compared to OCSVM.

## CONCLUSION

Our experiments demonstrate that the supervised learning methods significantly outperform the unsupervised ones if the test data contains no unknown attacks. Furthermore, among the supervised methods such as logistic and lasso logistic regression, the best performance is achieved by logistic regression, but its anomaly detection using testing dataset is slower than lasso logistic regression. On an average, the lasso logistic regression has better positive prediction value than logistic regression, i.e., the false alarms are less. In the presence of unknown attacks in the test data, the performance of unsupervised learning methods will be significant definitely. The performance of unsupervised learning will not degrade by the unknown attacks. In this study, among unsupervised learning methods, the performance of OCSVM is better than Isolation Forest. The findings from this study suggest that the semi-supervised and unsupervised methods, which do not require a laborious unreasonable labelling process, clear forerunners for practical purposes if unknown attacks can be expected. As a future research, there is need to improve Isolation Forest, when it handles more categorical variables with a higher number of levels. In case of imbalanced classes, an emerging field of semi-supervised learning, i.e., novelty detection, also offers a promising direction of future research NIDS especially in the fields were highly interconnected components produce a massive amount of data, e.g. in future Industrie 4.0 applications.

## ACKNOWLEDGMENT

This research is part of the project SmartDefense of the “Internetkompetenzzentrum Ostbayern” funded by the Bavarian Ministry of Economic Affairs and Media, Energy and Technology. The work further has been supported by the research projects SiNeMA (no. 13FH009I3) and DecADe (no. 16KIS0539) of the German Federal Ministry of Education and Research.

## REFERENCE

Berwick, R., 1990. An Idiot ’ s guide to Support vector machines ( SVMs ).

Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K., 2014. Network anomaly detection: methods, systems and tools. IEEE Commun. Surv. Tutorials 16, 303–336.

Chandola, V., Banerjee, A., Kumar, V., 2009. Anomaly detection: A survey. ACM Comput. Surv. 41, 1–58.

Chawla, N. V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. SMOTE: Synthetic minority over-sampling technique. J. Artif. Intell. Res. 16, 321–357.

Dokas, P., Ertoz, L., Kumar, V., Lazarevic, A., Srivastava, J., Tan, P.-N., 2002. Data mining for network intrusion detection. In: National Science Foundation Workshop on Next Generation Data Mining. pp. 21–30.

Fawcett, T., 2016. Learning from Imbalanced Classes [WWW Document]. URL https://www.svds.com/learning-imbalanced-classes/ (accessed 8.7.17).

Fayet, C., Delhay, A., Lolive, D., Marteau, P.-F., 2017. First Experiments to Detect Anomaly Using Personality Traits vs. Prosodic Features. In: Karpov, A., Potapova, R., Mporas, I. (Eds.), Speech and Computer: 19th International Conference, SPECOM 2017, Hatfield, UK, September 12-16, 2017, Proceedings. Springer International Publishing, Cham, pp. 379–388.

Friedman, A.J., Hastie, T., Simon, N., Tibshirani, R., Hastie, M.T., 2017. Package glmnet.

Garrette, D.H., 2006. Unsupervised Learning to Improve Anomaly Detection.

Gogoi, P., Bhattacharyya, D.K., Borah, B., Kalita, J.K., 2011. A survey of outlier detection methods in network anomaly identification3. Comput. J. 54, 570–588.

Goh, S.T., Rudin, C., 2014. Box drawings for learning with imbalanced data. Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD ’14 333–342.

Goldstein, M., Goldstein, M., Uchida, S., 2016. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data 1–31.

Görnitz, N., Rieck, K., 2013. Toward Supervised Anomaly Detection 46, 235–262.

Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Leasrning: Data Mining, Inference, and Prediction, Second. ed. Springer.

Heller, K., Svore, K., Keromytis, A.D., Stolfo, S., 2003. One class support vector machines for detecting anomalous windows registry accesses. Work. Data Min. Comput. Secur. (DMSEC), Melbourne, FL, Novemb. 19, 2003.

Jha, J., Ragha, L., 2013. Intrusion Detection System using Support Vector Machine. Int. J. Appl. Inf. Syst. 2013, 25–30.

Laskov, P., Düssel, P., Schäfer, C., Rieck, K., 2005. Learning intrusion detection: Supervised or unsupervised? Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 3617 LNCS, 50–57.

Lee, J., Kao, H.A., Yang, S., 2014. Service innovation and smart analytics for Industry 4.0 and big data environment. Procedia CIRP 16, 3–8.

Leung, K., Leckie, C., 2005. Unsupervised anomaly detection in network intrusion detection using clusters. Proc. Twenty-eighth Australas. Conf. Comput. Sci. - Vol. 38 38, 333–342.

Liu, F.T.., Ting, K.M.., Zhou, Z.-H.., 2012. Isolation-based anomaly detection. ACM Trans. Knowl. Discov. Data 6, 1–44.

Manandhar, P., 2014. A Practical Approach to Anomaly - based Intrusion Detection System by Outlier Mining in Network Traffic By.

Meshram, A., Haas, C., 2017. Anomaly Detection in Industrial Networks using Machine Learning: A Roadmap. In: Beyerer, J., Niggemann, O., Kühnert, C. (Eds.), Machine Learning for Cyber Physical Systems: Selected Papers from the International Conference ML4CPS 2016. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 65–72.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C.-C., Lin, C.-C., 2017. R Package e1071 (version 1.6-8).

Mok, M.S., Sohn, S.Y., Ju, Y.H., 2010. Random effects logistic regression model for anomaly detection. Expert Syst. Appl. 37, 7162–7166.

Möller, D.P.F., 2016. Digital Manufacturing/Industry 4.0. In: Guide to Computing Fundamentals in Cyber-Physical Systems: Concepts, Design Methods, and Applications. Springer International Publishing, Cham, pp. 307–375.

Plattform Industrie 4.0, 2016. Digitization of Industrie – Platform Industrie 4.0. Plattf. Ind. 4.0.

Scholkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., Platt, J., 2000. Support Vector Method for Novelty Detection. Adv. Neural Inf. Process. Syst. 1–7.

Sommer, R., Paxson, V., 2010. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. 2010 IEEE Symp. Secur. Priv. 305–316.

Ting, K.M., 2009. Adaptive Anomaly Detection using Isolation Forest.

Vlasveld, R., 2013. Introduction to One-class Support Vector Machines [WWW Document]. URL http://rvlasveld.github.io/blog/2013/07/12/introduction-to-one-class-support-vector-machines/ (accessed 9.1.17).

Zhang, M., Xu, B., Gong, J., 2016. An Anomaly Detection Model Based on One-Class SVM to Detect Network Intrusions. Proc. - 11th Int. Conf. Mob. Ad-Hoc Sens. Networks, MSN 2015 102–107.