Article

A Comparative Analysis of Semi-Supervised Learning in Detecting Burst Header Packet Flooding Attack in Optical Burst Switching Network

Md. Kamrul Hossain [1], Md. Mokammel Haque [2] and M. Ali Akber Dewan [3,*]

[1] Institute of Information and Communication Technology, Chittagong University of Engineering and Technology, Chattogram 4349, Bangladesh; muhammadkamrul@cuet.ac.bd

[2] Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chattogram 4349, Bangladesh; mokammel@cuet.ac.bd

[3] School of Computing and Information Systems, Faculty of Science and Technology, Athabasca University, Edmonton, AB T5J 3S8, Canada

* Correspondence: adewan@athabascau.ca

Abstract: This paper presents a comparative analysis of four semi-supervised machine learning (SSML) algorithms for detecting malicious nodes in an optical burst switching (OBS) network. The SSML approaches include a modified version of K-means clustering, a Gaussian mixture model (GMM), a classical self-training (ST) model, and a modified version of self-training (MST) model. All the four approaches work in semi-supervised fashion, while the MST uses an ensemble of classifiers for the final decision making. SSML approaches are particularly useful when a limited number of labeled data is available for training and validation of the classification model. Manual labeling of a large dataset is complex and time consuming. It is even worse for the OBS network data. SSML can be used to leverage the unlabeled data for making a better prediction than using a smaller set of labelled data. We evaluated the performance of four SSML approaches for two (Behaving, Not-behaving), three (Behaving, Not-behaving, and Potentially Not-behaving), and four (No-Block, Block, NB- wait and NB-No-Block) class classifications using precision, recall, and F1 score. In case of the two-class classification, the K-means and GMM-based approaches performed better than the others. In case of the three-class classification, the K-means and the classical ST approaches performed better than the others. In case of the four-class classification, the MST showed the best performance. Finally, the SSML approaches were compared with two supervised learning (SL) based approaches. The comparison results showed that the SSML based approaches outperform when a smaller sized labeled data is available to train the classification models.

Keywords: optical burst switching network; burst header packet flooding attack; semi-supervised learning; self-training; clustering

Citation: Hossain, M.K.; Haque, M.M.; Dewan, M.A.A. A Comparative Analysis of Semi-Supervised Learning in Detecting Burst Header Packet Flooding Attack in Optical Burst Switching Network. *Computers* **2021**, *10*, 95. https://doi.org/10.3390/computers10080095

Academic Editor: Paolo Bellavista

Received: 18 June 2021
Accepted: 30 July 2021
Published: 4 August 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Machine learning (ML) and data mining has been extensively used in communication networks for its ability to respond dynamically to the changes in networks without repetitive human intervention. Nowadays, ML is a common choice in addressing various problems and challenges pertaining to computer networking. From the existing literatures, we find that ML has been used in network traffic prediction, network traffic classification, payload-based traffic classification, host behavior-based traffic classification, flow feature-based traffic classification, encrypted traffic classification, misuse-based intrusion detection, anomaly-based intrusion detection, deep learning for anomaly detection, hybrid intrusion detection, etc. ML tries to construct algorithms and models that can learn to make decisions using hidden correlations discovered from historical data. In [1], De Sanctis et. al. presented a summary of the applications of data mining for communication network control and optimization by providing a comprehensive and structured review of the works in this area. The authors identified that the behavior of a communication network is complex, and numerous parameters that control this behavior are present. Besides, modern networks change their structure and scale very frequently over time. Hence, in order to achieve optimal performance, controlling the parameters of a network in real time is important, which is very hard to achieve using traditional network controlling software and hardware. The authors suggested that using data mining algorithms, one can find the hidden relation or pattern of behavior related to network performance and network control parameters. In [2], M. A. Ridwan et al. presented a detailed review on the current trends in the application of machine learning in communication networks. The authors efficiently surveyed the existing literatures published in the period between 2017 and 2020 and listed the significant works on the application of machine learning in vulnerability prediction, routing, Quality of Service enhancement, intrusion detection, resource management, etc. This showed that machine learning models can be used to efficiently reduce the gap between the computational complexity of modern communication networks and their performance. Since machine learning models can respond dynamically to the changes in networks without repetitive human intervention, it can be leveraged to satisfy the bandwidth-hungry and rigorous delay demand of modern communication networks. In [3], Boutaba et al. mentioned a wide-ranging applications of ML in communication networks.

In this paper, we chose to study a particular security problem in the optical network, more specifically in the Optical burst switching network. In an Optical burst switching (OBS) network, when a sender transmits a packet for destination, it first goes to an ingress node, which is an optical router. Other packets from various sources may join it in the aforementioned node. This collection of packets are then called data burst (DB). After waiting for a very short period, a control packet (also known as burst header packet (in short, BHP)) is generated from the ingress node, which attempts to allocate resources for the waiting packets. If the BHP can successfully manage resources, then the DB can start travelling to the destination using the allocated path. Optical communication technology found one of its most effective leverages in the optical burst switching network (OBS). OBS is a compromise between optical packet switching (OPS) and optical circuit switching (OCS), but at the same time it removes some major drawbacks in the aforementioned technologies [4]. In an OBS network, when packets are received by optical routers (also called ingress nodes), they are buffered for a small amount of time (not as long as OCS). This data payload is called a data burst. Then, a BHP is generated in the corresponding ingress node. Its aim is to travel towards the destination and ask the optical switches along the path to reserve a wavelength for the upcoming payload in the buffer. The BHP includes necessary information about the DB packets, such as the arrival time, burst length, offset time, etc. After waiting for a threshold time, the DB within the buffer is transmitted towards the destination. The DB and BHP are sent in a separate channel and the DB does not wait for any feedback from the corresponding BHP. The success of this scheme depends on the outcome of the reservation request by the BHP. If the BHP cannot reserve resource due to unavailability, then the corresponding DB will be dropped. OBS is preferred over OPS and OCS because it can ensure low setup latency, high bandwidth utilization, finer granularity, etc.

In this study, we particularly chose one notable attack, called the BHP flooding attack. Here, an adversary compromises an edge node and sends from it a sufficiently large number of BHPs to the core switches in order to cause a denial of service (DoS) state. These BHPs are intended to cause harm and they do not associate with the DB. This is a difference between a legitimate node and a compromised node. The BHPs sent from compromised node tries to reserve a resource in core switches. The core switches cannot distinguish a legitimate node and a compromised node and hence, allocate resources for the malicious BHPs, which results in exhaustion of resources. Figure 1 illustrates a scenario when this can occur [5]. BHP flooding attack is a significant threat, which can adversely impact the

Quality of Service (QoS) of an optical network and bring more critical problems such as DoS. To our knowledge, very few research studies are available on the application of ML in detecting a BHP flooding attack in an OBS network, and the application of SSML is even fewer. Besides, the existing techniques available to counter the BHP flood attack, exhibit low accuracy in detecting malicious nodes responsible for the BHP flooding attack. In addition, some notable work exists on supervised classification of OBS network nodes, but SL demands a lot of labeled data for satisfactory accuracy. Obtaining a fully labeled dataset for SL is challenging and time consuming. On the other hand, SSML which uses unlabeled data and simultaneously provides high accuracy result, is certainly a better choice. Moreover, semi-supervised learning methods have not been thoroughly examined for this scenario in existing literature. This motivated us to perform a comparative study of the existing works on SSML based BHP flooding attack detection, so that the outcome of this study will help the researchers to better diagnose the existing SSML based methods for OBS network related issues.

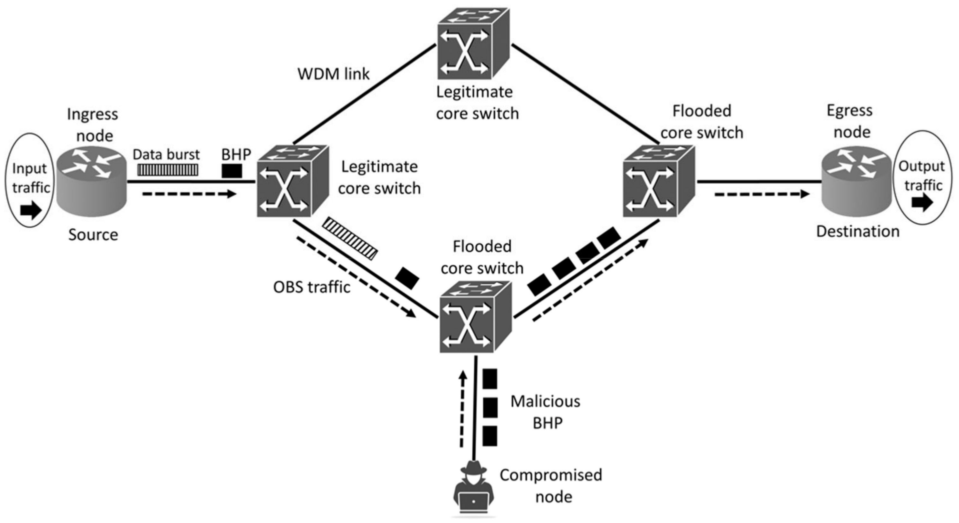

Figure 1. BHP flooding attack model.

This paper presents a comparative analysis of four semi supervised machine learning (SSML) approaches on an OBS network dataset for detecting a BHP flooding attack. In our previous studies [6–8], we applied four SSML approaches for detecting a BHP flooding attack. In [6], K-Means clustering technique was used in an SSML framework. In [7], Gaussian mixture model (GMM) was used in the same fashion. In [8], a classical self-training (ST) algorithm was examined and a modified version of the classical self-training algorithm was proposed for BHP flooding attack detection. In this study, we present a comparative analysis of these four SSML approaches on an OBS network dataset for detecting a BHP flooding attack. The contributions of this paper are outlined as follows:

- This paper presents a comparative study of the existing works on semi-supervised machine learning (SSML) based BHP flooding attack detection in OBS network. The outcome of this study will help the research community to better understand the SSML-based models for optical burst switching network related vulnerabilities.
- This study found that the SSML-based approaches outperformed the supervised learning-based approaches when a smaller sized labeled data is available to train the classification models.

The rest of the paper is organized as follows: Section 2 discusses some related works from the same problem domain. Section 3 describes the methodology for this study. Section 4 presents the experiments and analysis with the mentioned algorithms, and Section 5 concludes this study with a few remarks.

2. Literature Review

OBS networks are susceptible to various attacks. Very few studies exist on the application of machine learning to classify OBS nodes. In [9,10], authors studied the application of ML in OBS networks. In [6], authors classified the nature of data burst loss in the OBS network into two categories, i.e., loss due to contention and loss due to congestion. After computing observed losses, called the number of bursts between failures (NBBF), they used both SL and USL (unsupervised learning) techniques on the observed losses. They showed that the results had 95% confidence level. The authors in [7] proposed a proactive approach for contention resolution in OBS network. They introduced a new routing system called 'Graphical Probabilistic Routing Model' that chooses less frequently used links with the help of a Bayesian network. Using simulation software, they showed that the approach outperformed other static approaches in terms of burst loss ratio. In [11], the authors proposed a new approach to prevent such attacks by routing the light paths in a different manner. In [12], the authors studied the subject of data traffic with a view to reinforce network resource management by allowing the network admin to identify different types of data traffic. Precisely, the authors intended to determine a key issue relative to network performance with respect to source and destination. They collected data traffic features such as connection duration, byte counts, packet size, and inter-arrival statistics. Using EM based clustering algorithm, the dataset was clustered. The traffic flows were divided into a set of groups. The outcome showed that at least 6 key clusters (based on single and bulk transactions) were differentiable between the traffic flows. In a survey [13], the authors described various ML approaches dealing with categorization of IP traffic within diverse types of computer networks. They centered their study on the existing classification methods. Both SL and USL approaches were examined for traffic flow classification across several commonly used computer networks. They observed USL using the concepts of clustering, EM, and K-Means. For SL, they used several techniques such as Genetic Algorithm, Naïve Bayes, and K-Nearest-Neighbor.

The ML approaches discussed above focused on data traffic recognition, while this study is solely focused on BHP flooding in OBS network and its SSML-based classification. In [14], the authors explored the BHP flooding attack and proposed some solution to filter the malicious BHPs. They proposed a countermeasure module for a DoS attack that performs the fake BHP filtering at the optical layer using the idea of optical "codewords". When a received BHP comes from an illegitimate source node indicated by codewords, it was dropped. In [15], authors designed an architecture of a firewall node to defend OBS networks against physical layer attacks such as BHP flood attack. The firewall filters by comparing the offset time of BHP and the real delay between the BHP and the accompanying DB. In [2], the authors designed and implemented an algorithm to classify the ingress nodes of an OBS network into three classes, i.e., Trusted, Blocked, and Suspicious. Based on the node's behavior and the amount of unutilized reserved resources, the classification was performed. According to the authors, the model can be integrated in the OBS core switch, which can enable it to classify the nodes. The methods described above make heavy use of expert's opinion to assign labels to the behavior and collected data, i.e., marking whether a behavior is good or bad. Getting an expert's hand-assigned data is not always feasible. In [16], authors proposed a decision tree-based supervised classification model. A decision tree algorithm was used to extract If-Then, rules which were used to classify the ingress nodes of OBS network into either Behaving or Misbehaving nodes. The authors further classified the Misbehaving ingress nodes into four sub-classes, i.e., Misbehaving-No Block, Misbehaving-Wait, Misbehaving-Block, and Behaving-No Block. First, they built a dataset of 1075 records using network simulation software and expert's hand-assigned target class labels. Then, the dataset was used to perform the training and testing of decision tree model. Their model showed 93% detection accuracy for two class classification, and 87% detection accuracy in case of four class classification. In [17], the authors applied deep learning method in a supervised fashion. The malicious OBS network nodes were detected from an existing BHP flooding attack dataset. The authors concluded their work with a comparison between their work and work based on support vector machine, Naïve Bayes, and k-nearest neighbors.

The research studies discussed above are either rule-based or based on supervised learning (SL). However, SL demands a lot of labeled data for satisfactory accuracy. Obtaining a fully labeled dataset with thousands of data for SL is expensive, time consuming, and sometimes impossible. In comparison, Semi-supervised machine learning (SSML) which can leverage unlabeled data and at the same time provides high accuracy result, is certainly a better choice. Moreover, semi-supervised learning methods have not been thoroughly examined for attacks in the OBS network. Various methods are found in existing literatures for exploiting unlabeled data in SSML. Among them, generative models, low-density separation, Laplacian regularization, and heuristic approaches are mostly used. Generative model is an effective method that works by approximating the distribution of data points for each class of a dataset. The probability that any given point 'a' has label 'b' is then proportional to p(a | b)p(b) by Bayes' rule. In [18], M. Lopez-Martin et al. presented a novel method to synthesize intrusion data using generative model based on a modified variational autoencoder (VAE). This method can produce synthetic data (for both categorical and continuous feature) with behavioral and probabilistic structure similar to the original data. In order to find the best model, the authors tried different VAE architecture alternatives. The authors concluded that the model based on conditional VAE with Gaussian and Bernoulli distributions exhibited the best performance. This outcome was evaluated in terms of similarity of the generated data to real data and its capacity to be used as new training data for machine learning models. Furthermore, this proposed method was compared to several over-sampling algorithms. Synthetic data produced by both the methods were used along with the NSL-KDD training dataset. Four common classifiers (random forest, linear SVM, logistic regression, and Multilayer Perceptron) were trained on the data and the classification result showed that the proposed method outperformed the others in terms of classification accuracy. In [19], D. P. Kingma et al. presented a noble probabilistic method of semi-supervised learning using deep generative models. The authors leveraged approximate Bayesian inference combined with scalable variational methods in order to produce synthetic data with a behavioral and probabilistic structure similar to the original data. Besides, in order to make their proposed system scalable, a new stochastic variational inference algorithm was designed. The proposed deep generative model was evaluated on multiple benchmark datasets and it outperformed the previously known best methods. The authors outlined that a limitation of the proposed model is that it linearly scales according to the number of classes in a dataset, which requires to re-estimate the generative likelihood for each of the classes during training. In [6–8], the authors discussed some SSML approaches for detecting BHP flooding attack using recorded OBS network data. In [6], K-Means clustering technique was used in an SSML framework. The authors applied K-means to the whole dataset and found some groups of data. Then, using available labeled data, those groups were identified with class names. In [7], Gaussian mixture model (GMM) was used in the same fashion as mentioned in [6]. In [8], a classical self-training (ST) algorithm and a modified version of the classical self-training algorithm was proposed for a BHP flooding attack detection using an OBS network dataset. The self-training [20–22] algorithm is a common choice for SSML. It uses available truly labeled data to initially train a base classifier and then uses the model to predict the unlabeled data. The predicted data are then filtered to choose only high confidence predictions. These high confident samples are then added to the original truly labeled data. This process continues until all the unlabeled data are labeled or a max iteration is reached. In [5], the authors proposed a new algorithm based on the self-training algorithm, called Modified self-training algorithm. The method applies multiple classifiers on the same dataset and combines the outcomes using some heuristics. These methods ([6,7]) gave good accuracy for binary classification of an OBS network dataset but were not good for three- and four-class. The self-training and its modified version were found to be good for all cases.

As per our study, very few works exist on the application of ML in detecting BHP flooding attack of OBS network. Among them, semi-supervised learning-based works are fewer. To the best of our knowledge, the works [6–8] are the only available works on SSML approach for detecting BHP flooding attack in an OBS network dataset. SSML is very helpful when the available amount of labeled data is relatively much smaller than the unlabeled data. Manually labeling a large dataset is complex and time consuming, especially for OBS network data. SSML can be used to leverage the unlabeled data for making a better prediction than what would be possible if only the labeled data were used. In this work, we presented a comparative analysis of the SSML approaches used in [6–8] for detecting BHP flooding attack in an OBS network dataset.

3. Materials and Methods

In this section, we present the underlying algorithms and their setups for each of the four SSML approaches that were examined in this paper. For convenience, this section is divided into four sub-sections.

3.1. Study 1: SSML with K-Means Clustering

A K-means based SSML approach is used for detecting BHP flooding from an OBS network dataset in [3]. With some modifications, we examined the method for a comparative analysis. In this approach, a dataset of OBS network node's data was selected. The dataset was split into two parts: a smaller validation set and a large test set. The whole dataset was used for the K-means clustering [23]. The associated true labels were removed from the test set. The validation set was very small compared to the whole dataset and it holds true labels of each sample. After the K-means clustering, the obtained clusters were given labels based on the validation set. Using this validated model, the test set was classified (predicted) and assigned labels. Finally, the evaluation metrics for the model were calculated. Figure 2 depicts the whole procedure in a block diagram and in Algorithm 1, the K-means algorithm [24] is presented. The symbols used in Algorithm 1 is explained in Table 1.

Algorithm 1 k-Means Algorithm [19]

```
   Input: data point for dataset $X$, i.e. ($x \in X$) and number of centers $k$
1. Select an initial partition with $k$ clusters
2. Repeat
3.     Compute a new partition by assigning each data point to its nearest cluster center
4.     Generate new cluster centers
5. Until cluster membership become stable
```

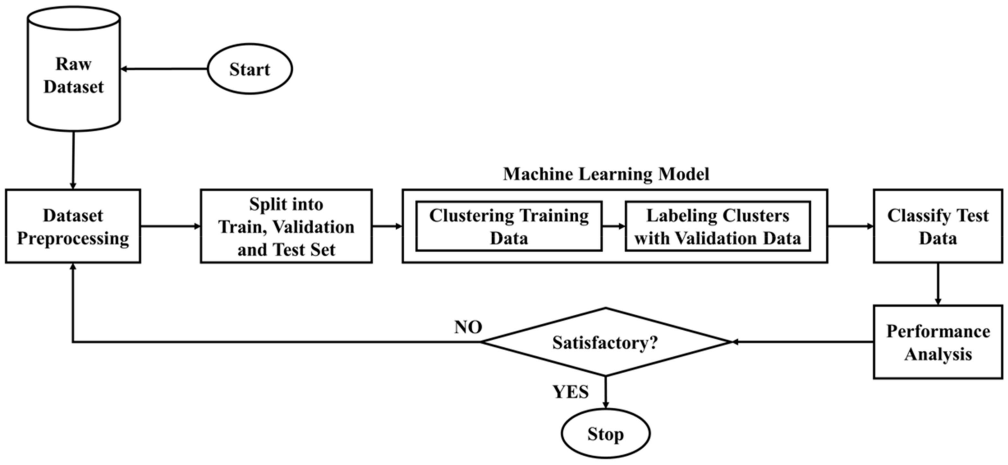

Figure 2. Block diagram of K-means algorithm in the SSML approach.

Table 1. List of symbols of Algorithm 1.

Symbol	Meaning
X	Unlabeled dataset
x	single data point from X
k	number of clusters

3.2. Study 2: SSML with Gaussian Mixture Model (GMM) Clustering

A GMM based SSML approach is used for detecting BHP flooding from OBS network dataset in [4]. With some modifications, we examined the method for the comparative analysis. The Gaussian mixture model [25] is a probabilistic model that calculates the joint probability for a data point to determine the most probable cluster by leveraging the expectation maximization (EM) algorithm [26,27]. Here, the means and covariance of GMM are initialized using available labeled data. In Figure 3, the block diagram of the method used by the author for clustering OBS network data is depicted. Additionally, the EM for the GMM algorithm is shown in Algorithm 2. The symbols used in Algorithm 2 are explained in Table 2.

Algorithm 2 **EM Algorithm for Mixture of Gaussians**

```
1.  Given: All points $x \in X$ that are mixtures of $K$ Gaussians
2.  Goal: Find $\pi_1, \ldots, \pi_k$ and $\Theta_1, \ldots, \Theta_k$ such that
    $L(\Theta) = \sum_{i=1}^{n} \ln\left\{ \sum_{k=1}^{K} \pi_k F(x_i|\Theta_k) \right\}$ is maximized
3.  Initialize the means $\mu_k$, variances $\Sigma_k$ for each component
4.  Initialize the mixing coefficients $\pi$ and evaluate the initial value of log likelihood $L(\Theta)$
5.  Expectation step: Evaluate weights $w_{ik}$
6.  Maximization step:
    Re-evaluate parameters $\mu_k^{new}$, $\Sigma_k^{new}$ and $\pi_k^{new}$
7.  Evaluate $L(\Theta^{\mathrm{new}})$
8.  if $L(\Theta^{\mathrm{new}})$ converged then stop
9.      else goto 4
10. end if
```

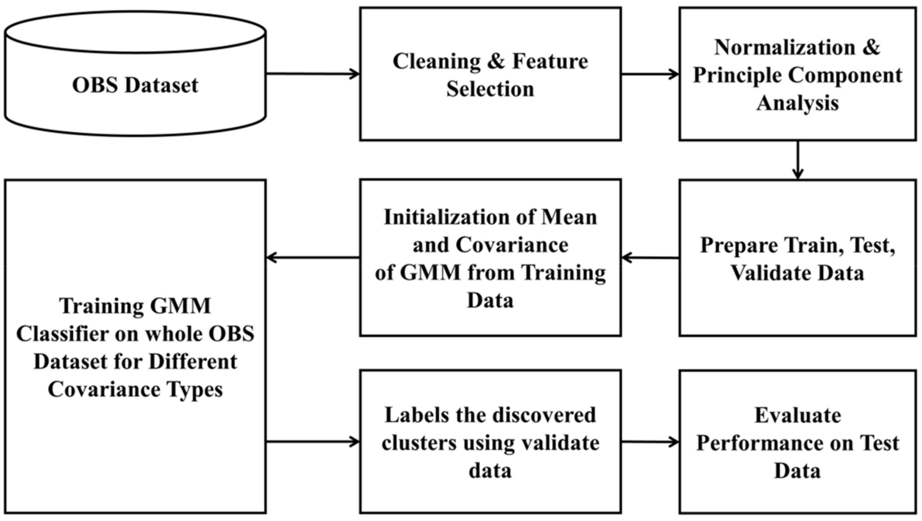
OBS Dataset
Cleaning & Feature Selection
Normalization & Principle Component Analysis
Prepare Train, Test, Validate Data
Initialization of Mean and Covariance of GMM from Training Data
Training GMM Classifier on whole OBS Dataset for Different Covariance Types
Labels the discovered clusters using validate data
Evaluate Performance on Test Data

Figure 3. Block diagram of the GMM-based SSML approach.

Table 2. List of symbols of Algorithm 2.

Symbol	Meaning
μ_k	mean of k-th gaussian
Σ_k	variance of k-th gaussian
x	data point for dataset X, i.e., (xϵX)
n	total data points in dataset X
k	mixture components
w_{ik}	probability that point x_i is generated by the k-th Gaussian
N_k	$\sum_{i=1}^{n} w_{ik}$ i.e., the effective number of data points assigned to k-th Gaussian
π_k	prior probability(weight) of k-th gaussian
Θ_k	(μ_k, Σ_k)
$F(x_i \mid \Theta_k)$	probability distribution of observation x_i, parameterized on Θ

In this paper, we implemented the GMM for SSML in the same fashion as the K-means-based SSML approach. Before applying the method, some preprocessing, such as Normalization and Principal component analysis (PCA) was done in order to make the dataset more suitable for the model.

3.3. Study 3: SSML with Self-Training

In [5], the authors discussed the traditional self-training algorithm in comparison to a modified version of it. In this work, we shall examine the method in context of OBS network dataset classification for a comparative analysis. Self-training is an old but effective method for SSML [20,21,28]. A classical version of the self-training method (say it, ST) makes use of the true-labeled data (L) in order to train a base classifier (C). Here, some true-labeled data are needed before the algorithm can start its job.

Next, C is used to predict pseudo labels for unlabeled data (U). Then, a calculated portion (P) from those pseudo labeled instances is separated from U with their associated labels. The calculation we are talking about typically works based on prediction confidence (H). The separated portion can then be combined with the truly labeled data L. The outcome, i.e., P + L dataset is stored and then used to re-train C. This aforementioned process is repeatedly done until either U is empty, or a preset number of iterations is achieved. This whole procedure and steps are shown formally in Algorithm 3 and a block diagram is shown in Figure 4.

Algorithm 3 **Classical Self-Training Algorithm (ST) [17]**

```
1. $N$: Iteration counter = 1; $C$: Base classifier, $L$: Labeled data, $U$: Unlabeled data, $maxCount$:
   number of iterations allowed, $H$: Confidence Threshold
2. while ($U$! = empty) and ($N$ < maxCount) do
3.         Train $C$ on $L$
4.         for each $d_i$ in$U$ do
5.               Assign pseudo-label to $d_i$ based on prediction confidence
6.         end for
7.         Select a set $P$ of the high-confidence predictions from $U$ based on threshold $H$
8.         Update $N = N + 1$; $U = U - P$; $L = L \cup P$
```

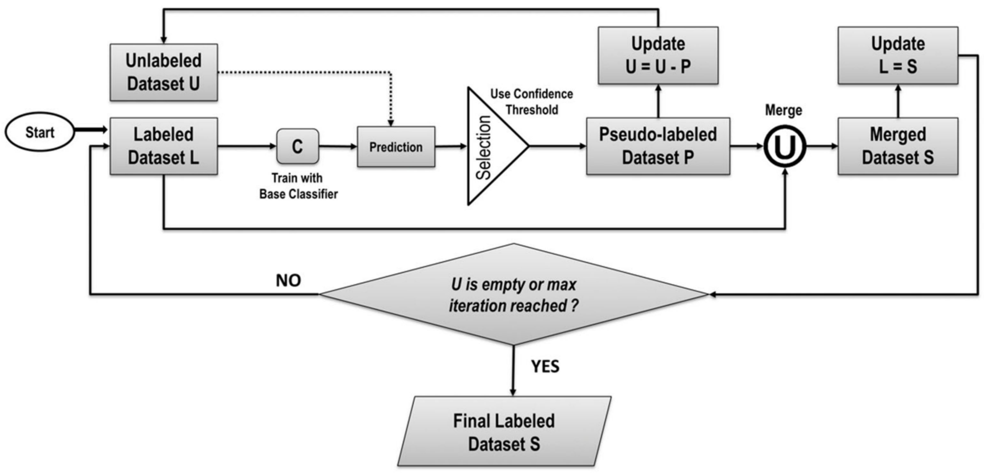

Figure 4. Block diagram of self-training algorithm.

However, the problem with this method is that it is very slow, especially for big datasets because it keeps on iterating until the constraints are met. Another shortcoming outlined by researchers is that inaccuracy in primary stages may be reinforced in future iterations [29]. In this work, the Extra Tree classifier [30] was used in Algorithm 3 as the base classifier. Then, following the method suggested in [22,31], confidence threshold H was determined. Accordingly, after the model predicted labels for U, the top 10% from the highly-confident predictions are separated and the mean value of their probability estimation was computed. This mean value was set as H. In the algorithm, the value for the variable 'maxCount' was set to 40 based on the recommendation mentioned in [32].

3.4. Study 4: SSML with Modified Self-Training

In [5], a method developed from the idea of self-training was proposed. It is called modified self-training method (in short, MST). The method applies multiple (N) classifiers (A_1, A_2, ... , A_N) on the same dataset and combines the outcomes using some heuristics. Moreover, the agreement among those classifiers was exploited in a two-stage selection pipeline. At stage one, different ML classifiers are trained by the instances available at hand (L). The resulting models then work on the unlabeled dataset (U) and each model produces a pseudo-labeled dataset from U. After that, the first level decision fusion was applied by preserving only those instances whose labels were agreed by a fixed number of models among the available models. Assume P is the output which is the subset of U chosen by a fixed number of models label agreement. This concludes the first stage voting. In the next stage, different classifiers are trained by L as the first stage. Then, the resulting models work on P and assign labels. This produces a collection of labeled datasets. At this point, a subset from P was selected based on two thresholds: prediction probability and label agreement. The labels of P are chosen based on these two thresholds. For example, if for an instance in P, a label was agreed by a specific number of models and each of those models had prediction probability above a specific threshold, then the instance (along with the label) was selected and preserved. Let us say that the subset of P selected in this way is called S. Then, S is merged with L. The resulting dataset is the final dataset (say, F) which can be used for training a classifier in future. Algorithm 4 presents the steps and Figure 5 depicts the methodology in a block diagram.

Algorithm 4 **Modified Self-Training Algorithm (MST) [5]**

```
1.  C1: A set of N classifiers, C2: A set of M classifiers, F: Final classifier,
    L: Labeled data, U: Unlabeled data, K: Set of empty datasets, P, S: Empty dataset, V1: 1st stage
    pseudo-label agreement threshold, V2: 2nd stage pseudo-label agreement threshold, H:
    Classification confidence threshold
2.  $C1 = \{C_{11},C_{12},...C_{1N}\}$, $C2 = \{C_{21},C_{22},...C_{2M}\}$, $i = 1$, $K = \{K_1,K_2,...K_N\}$
3.  for each c in C1 do
4.          Train c by training set L
5.          for each d in U do
6.              Assign pseudo-label to d based on prediction confidence
7.              Save d along with the pseudo-label in set $K_i$
8.          end for
9.          $i = i + 1$
10. end for
11. for each d in U do
12.         compute pseudo-label agreement (votes) for d in K
13.         if votes >= V1 then
14.             copy d and save in set P
15.         end if
16. end for
17. $i = 1$ and $K_1,K_2,...K_N = \{\}$
18. for each c in C2 do
19. Train c by training set L
20.         for each d in P do
21.             Classify d by model c
22.             if classification confidence >= H then
23.                 Save d along with the pseudo-label in set $K_i$
24.             end if
25.         end for
26.         $i = i + 1$;
27. end for
28. for each d in P do
29.         compute pseudo-label agreement(votes) for d in K
30.         if votes >= V2 then
31.             copy d in set S
32.         end if
33. end for
34. Update U by removing S from U: $U = U - S$
35. Update L by joining S with L: $L = L \cup S$
36. Train classifier F by training set L
37. Classify U by F and predict labels for all the points in U
38. Output: Fully labeled dataset
```

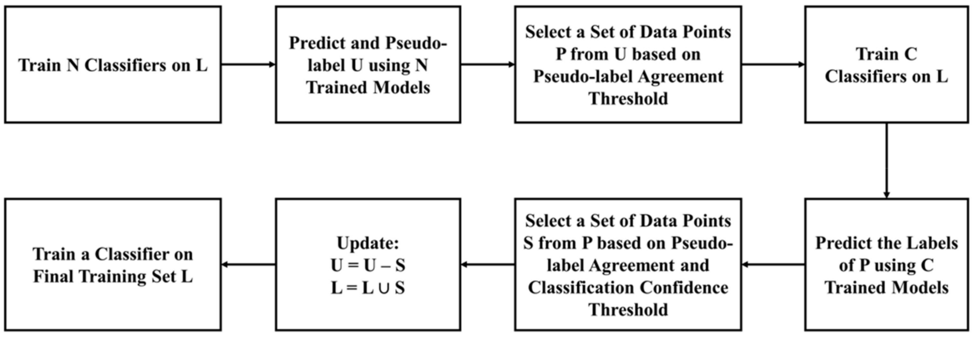
Train N Classifiers on L
Predict and Pseudo-label U using N Trained Models
Select a Set of Data Points P from U based on Pseudo-label Agreement Threshold
Train C Classifiers on L
Predict the Labels of P using C Trained Models
Select a Set of Data Points S from P based on Pseudo-label Agreement and Classification Confidence Threshold
Update: U = U – S L = L ∪ S
Train a Classifier on Final Training Set L

Figure 5. Modified self-training (MST) method in block diagram.

In this paper, according to Algorithm 4, eight classifiers (i.e., n = 8) were used at the first stage of voting. It includes Extra Tree classifier [30], Gaussian Naive Bayes [33], Gradient boosting classifier [34], Logistic regression classifier [35], Random Forest classifier [36], XGBoost classifier [37], Quadratic discriminant analysis [38] and Multi-layer perceptron classifier [39]. At the second phase of the voting, six ML classifiers were used (i.e., M = 6), which included the above listed classifiers except for the Extra tree classifier and Random Forest classifier. The reason for not including these two in the second stage is that the standard decision tree classifier produces unreliable probability estimation to its predictions, which cannot be an appropriate selection criterion in self-training [17]. The value of the three tuning parameters (or, Thresholds) mentioned in parameters (or, Thresholds) mentioned in Algorithm 4 (V1, V2, and H) were obtained empirically following the suggestion given by [5]. Once the final training dataset (fully labeled by MST) was produced, an Extra Tree classifier was trained by it for the intended comparative analysis.

4. Experiment and Analyses

In this section, the experimental results and related analyses are presented. For convenience, this section was divided into following sub-sections: experiment setup, dataset preparation, evaluation metrics, and results and analyses.

4.1. Experimental and Environmental Setup

During the experiments, we used a Lenovo PC with an Intel core i-7, 3.40 GHz processor, 8 GB RAM. We performed ML tasks in Windows 10 ($\times$64). For coding environment, we used 'Jupyter Notebook' which is available in Anaconda [40] distribution.

It was stated earlier that the dataset has a target attribute named 'Node Status'. It holds one of three values for each row: B, NB, and PNB. It also has a target attribute named 'Class' which holds one of four values for each row: No-Block, Block, NB-wait, and NB-No-Block. The label 'PNB' means 'potentially not behaving', i.e., the node has high packet drop rate but not as high to be labelled as malicious. The label 'NB-No-Block' means that the node is misbehaving but we do not block it immediately. The label 'No- Block' means that the node is not malicious and hence should not be blocked. The label 'Block' means that the node is malicious and hence should be blocked. The label 'NB-Wait' means that the node is misbehaving and should be blocked temporarily (i.e., for a time specified). Based on this, for each of the four studies mentioned in Section 3, we divided our test cases into the following three categories:

- First Case: we trained SSML models to classify the nodes (i.e., data instances) in the dataset into two distinct classes: Behaving (B) and Not Behaving (NB) based on the target attribute 'Node Status'.
- Second Case: models were trained to classify the nodes into three distinct classes: Behaving (B), Not Behaving (NB) and Potentially Not Behaving (PNB) based on the target attribute 'Node Status'.
- Third Case: models were trained to classify the nodes into four distinct classes based on the target attribute 'Class', which has four distinct class values: No-Block, Block, NB- wait, and NB-No-Block.

It should be noted that the target attribute 'Node Status' was used for the 'First Case' and 'Second Case' while the target attribute 'Class' was used for the 'Third Case'. The original OBS dataset has 760 instances for 'First Case' and 1075 instances for 'Second Case' and 'Third Case'.

In Table 3, the ML algorithms and their parameters used in this study are listed. All the ML algorithms were implemented using Scikit-learn [41] which is a free machine learning library for the Python programming language. In this experiment, the parameters for the ML algorithms were kept in default setting as specified by Scikit-learn, except for a few, which are listed below.

Table 3. List of modified parameters of algorithms used in this work.

Algorithm Name	Modified Parameters
Gaussian Mixture Model	covariance_type= ['spherical', 'diag', 'tied', 'full'], maximum iteration = 100, radom_state or seed = 120
K-Means	No parameter modified
Extra Trees Classifier	random_state or seed = 120
Random Forest Classifier	random_state or seed = 120
XGBoost Classifier	random_state or seed = 120
Gradient Boosting Classifier	random_state or seed = 120
Gaussian	No parameter modified
Logistic Regression	maximum iteration = 1000, random_state or seed = 120
Multi-Layer Perceptron Classifier	maximum iteration = 1500, random_state or seed = 120
Quadratic Discriminant Analysis	No parameter modified

4.2. Dataset Preparation

This work utilizes an OBS network dataset related to BHP flooding attack available in UCI Machine Learning Repository [42]. The dataset has 1075 records with 22 attributes. Below, a brief intro of the attributes is provided in Table 4.

Table 4. Meaning of the attributes of the OBS dataset.

Sl.	Attribute name	Meaning
i.	Node	This is the label of edge node
ii.	Full bandwidth	This is a user allocated initial reserved bandwidth for an individual node. It is also called reserved bandwidth
iii.	Utilized bandwidth rate	The amount which can be reserved from the allocated bandwidth, i.e., from full bandwidth column
iv.	Packet drop rate	Packet drop rate for individual node, in percentage
v.	Percentage of lost packet rate	Packets drop rate, in percentage for individual node
vi.	Average delay time per sec	Average of delay per second for individual node
vii.	Packet received rate	Total packets received per second for individual node on the basis of reserved bandwidth
viii.	Percentage of lost byte rate	Lost byte rate, in percentage for individual node
ix.	Amount of used bandwidth	The amount each individual node could reserve from allocated bandwidth
x.	Packet size byte	Packets size allocated in byte for individual node to send
xi.	Received byte	This is the total byte received per second for an individual node on the basis of reserved bandwidth
xii.	Lost bandwidth	The lost amount of assigned bandwidth
xiii.	10-run-avg- drop-rate	This is the average of packet drop rate for ten successive iterations in simulation
xiv.	10-run-delay	The average of delay time for 10 successive iterations in simulation
xv.	10-run-avg- bandwidth use	This is the average of bandwidth utilized for 10 successive iterations in simulation
xvi.	Packet transmitted	The amount of total packets transmitted per second for individual node on the basis of allocated bandwidth
xvii.	Packet received	The amount of total packets received per second for individual node on the basis of reserved bandwidth

Table 4. *Cont.*

Sl.	Attribute name	Meaning
xviii.	Packet lost	The amount of total packets lost per second for individual node on the basis of lost bandwidth
xix.	Transmitted byte	Total bytes transmitted per second for individual node
xx.	Flood status	The amount of flood per node, in percentage on the basis of packet drop rate
xxi.	Node Status	The classification of nodes into one of three classes, behaving, potentially not behaving, and not behaving
xxii.	Class	It is the classification of the nodes into one of four classes; NB-No- Block, block, NB-wait, no-block

The dataset was built from rigorous simulation runs using an OBS network simulator software [43]. With the assistance of a domain expert, the authors labeled the dataset's two categorical attributes, i.e., (B and NB) for the 'Node Status' attribute, and then "No-Block", "NB-No-Block", "Block", "NB-Wait" for the 'Class' attribute. The category wise label assignments were done based on the intentional false resource utilization rate and the real packet drop rate. Three attributes: 10-run-avg-bandwith- use, 10-run-avg-drop-rate, 10-run-delay, each one symbolizes an average value calculated from 10 consecutive iterations in the simulator. This was done to retain the statistical significance and minimize the bias within the node performance results [13]. The attributes 'Node Status' and 'Class' are the target attributes.

Necessary data cleaning was done and then feature selection was applied. With the help of Pearson correlation coefficient (PCC), 11 attributes were identified as redundant and hence can be removed from the dataset. As a result, apart from two target attributes 'Node Status' and 'Class', 8 attributes are available to be used in training the desired ML model. Those are:

1. Average delay time per sec
2. Amount of used bandwidth
3. Packet transmitted
4. Packet lost
5. Received byte
6. 10-run AVG drop rate
7. 10-run delay
8. 10-run AVG bandwidth use

We further scrutinized these eight attributes using CHI and CHF filtering method [44,45]. The CHI filtering technique was employed to choose significant features and the CFS filtering technique was used to confirm the result. It indicated that the following features are the most significant:

1. 10-run AVG drop rate
2. 10-run delay
3. 10-run AVG bandwidth use

These three attributes were used for training ML models in all the SSML approaches.

4.3. Evaluation Metrices

We chose four commonly referred metrics to assess the performance. Those are: accuracy score, precision score, F1 score, and recall score. Since these metrics function differently for binary and multi-class dataset, we measured them appropriately. In case of multi-class classification, the learning algorithm is wrapped in a one-vs-rest fashion to produce binary comparisons for each class. Accuracy is the ratio of correctly predicted labels to the total labels. It is expressed in percentage. Precision, for the binary labeled dataset, is the ratio of correctly predicted positive labels to the total predicted positive labels. Recall, for the binary labeled dataset, is the ratio of correctly predicted positive labels to the total observations in actual positive label. F1 Score returns the weighted average of precision and recall. In case of multi-class dataset, precision, recall and F1 score are computed differently. We calculated the metric for each label and then computed their unweighted mean (macro average) [46]. The value of recall, precision and F1 score range from 0 to 1—the bigger the better.

4.4. Results and Analyses

In this subsection, the experimented outcomes are discussed. In Tables 5–7, classification results are shown for all three test cases mentioned in the experimental setup. Table 5 shows the result of two class classification of the OBS network dataset. Four evaluation metrics are listed for all the four SSML methods described in Section 3. The number of truly labeled instances used in this part of the experiment was five. That is, every model was given five labeled samples and the remaining instances were unlabeled. From Table 5, the following observations can be shown. In terms of evaluation metrics, K-means and GMM-based SSML approach performed best among the four listed SSML approaches. Both performed identical for the dataset. The other two approaches, ST and MST, were very close to the other two approaches in terms of evaluation metrics. The main reason for the success of K-means and GMM-based SSML is that the OBS dataset for two class exhibits an easily separable spherical-like shape when plotted as illustrated in Figure 6.

Table 5. Evaluation metrics for two class classification.

SSML Method	Accuracy	F1	Precision	Recall
K-means- based	99	0.992	1.000	0.99
GMM-based	99	0.992	1.000	0.99
ST	98.013	0.973	1.000	0.947
MST	98.675	0.982	1.000	0.965

Table 6. Evaluation metrics for three class classification.

SSML Method	Accuracy	F1	Precision	Recall
K-means-based	87.32	0.879	0.890	0.891
GMM-based	64.98	0.631	0.619	0.819
ST	90.038	0.779	0.923	0.728
MST	89.089	0.784	0.818	0.762

Table 7. Evaluation metrics for four class classification.

SSML Method	Accuracy	F1	Precision	Recall
K-means-based	55.69	0.576	0.569	0.701
GMM-based	54.86	0.573	0.575	0.696
ST	89.229	0.889	0.879	0.903
MST	89.229	0.893	0.884	0.905

In Figure 7, the training time for the four methods is illustrated. Here, K-means has the fastest training time and GMM has a slightly larger time than K-means. The other two methods, ST and MST, showed significant difference in training time than the former two. ST was the slowest among the four methods as per our experiment. It is to be mentioned that GMM was run for four different covariance types. It was found that each of four covariance types show a slightly different performance for a different number of classes. Here, we only reported the one with the best result.

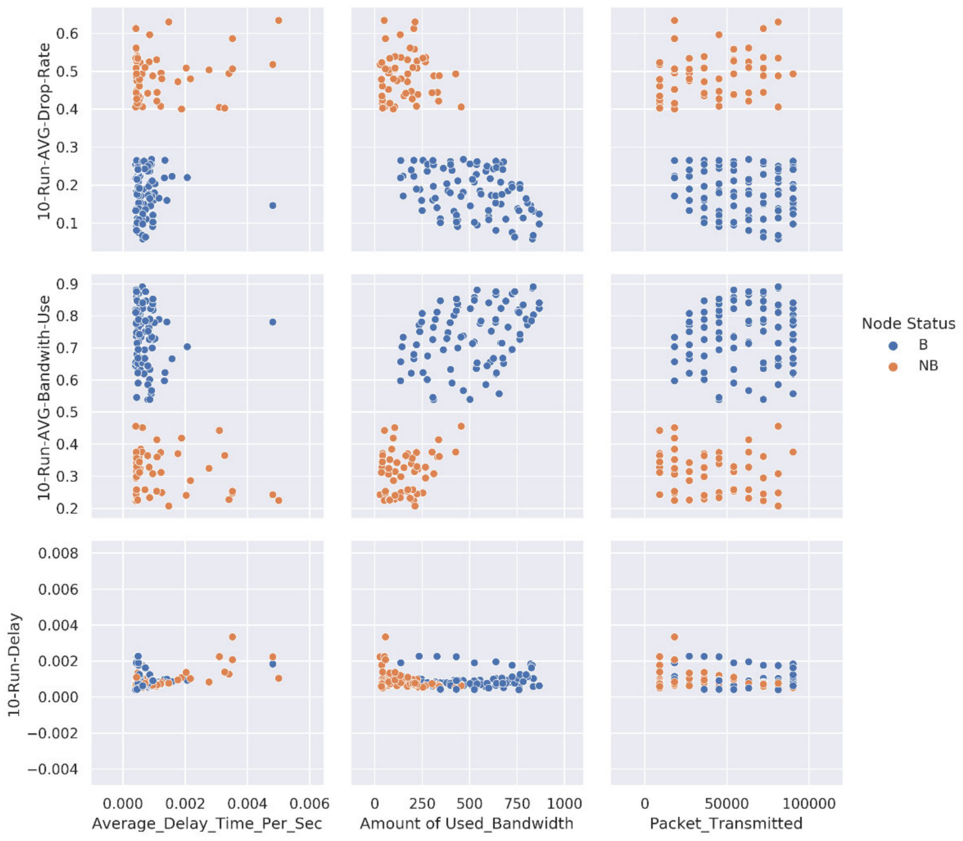

Figure 6. Plot of three attributes against other three.

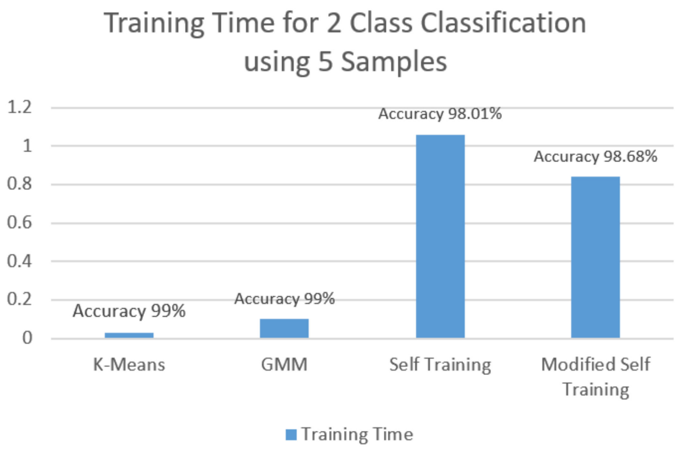

Figure 7. Training time for binary classification.

In Table 6, the result of three class classification of OBS network dataset is shown. Four evaluation metrics are listed for all the SSML methods described in Section 3. The number of truly labeled instances used in this part of the experiment was 21. That is, every model was given 21 labeled samples and the remaining instances were unlabeled. From Table 6, the following observations can be achieved. In terms of evaluation metrics, ST, MST, and K-means-based SSML performed very well among the four listed SSML approaches. They showed a very similar performance for the dataset. In terms of accuracy and precision, ST was better than the rest. In terms of F1 and recall score, K-means-based SSML was better than the rest. MST was better than ST in terms of F1 and recall score. From the table, it can be concluded that, in terms of performance, K-means-based SSML was overall better than MST and overall similar to ST. At the same time, ST was overall similar in performance to MST. Lastly, the remaining method, GMM-based SSML, performed very poor in terms of accuracy, precision, and F1 score. The problem that occurred here with GMM is that it might have converged quickly to a local minimum that is not very optimal for the three-class OBS network dataset. Using better initialization technique for GMM may help perform better.

In Figure 8, training time by the three-class dataset for the four methods is illustrated. Here, K-means has the fastest training time and GMM has slightly larger time than K-means. The other two methods, ST and MST, showed significant difference in training time than the former two. ST was the slowest among the four methods as per our experiment.

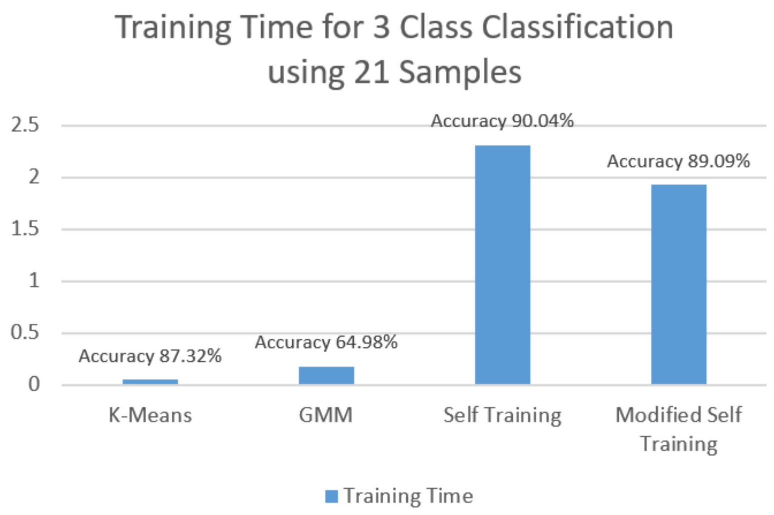

Figure 8. Training time for three class classification.

Table 7 shows the result of four class classification of the OBS network dataset. Four evaluation metrics are listed for all the four SSML methods described in Section 3. The number of truly labeled instances used in this part of the experiment was 193. That is, every model was given 193 labeled samples and the remaining instances were unlabeled. This is to be mentioned because, for each of the cases mentioned in Section 4.1, we started with a smaller number of labeled data and gradually increased the number until a high accuracy result was achieved. For the case of three-class classification, K-means and GMM based SSML methods failed to produce high accuracy even with a large number of labelled data. That means, increasing the labelled data size does not help here. For ST and MST, increasing the labelled data helped achieve better performance. So, we increased the labelled data size until the accuracy exceeded the related work's accuracy. This resulted in the amount of labelled data being 193. Hence, from Table 7, the following observations can be drawn. In terms of evaluation metrics, K- means and the GMM-based SSML approach performed the worst among the four listed SSML approaches. Both of these two performed almost identical for the dataset. The reason behind the poor performance of K-means-based SSML is that the 4 class OBS dataset has a very complex geometric shape as illustrated in Figure 9.

Besides, K-means, when attempting to minimize the intra-cluster variation, gives more weight to larger clusters than smaller ones. This situation can be improved using a method called the 'kernel method', which tries to transform the current data points to a higher dimensional representation in order to make the data linearly separable. The reasons mentioned here are more or less responsible for the poor performance of GMM-based SSML. However, GMM might have converged quickly to a local minimum here as well, which is not optimal for the data points. Using a better initialization technique for GMM may help improve the situation. Apart from this, the other two approaches, ST and MST, were far better in terms of evaluation metrics. Both ST and MST showed very good performance, though there is a difference of performance. In terms of accuracy, both were identical but in terms of other three evaluation metrics, MST performed better than ST, which makes MST superior to ST in case of four class classification of the OBS dataset. In

Figure 10, training time of the four-class dataset for the four methods is illustrated. Here, it followed the same pattern as the previous two cases. K-means has the fastest training time and GMM has slightly larger time than K-means. The other two methods, ST and MST, showed a significant difference in training time than the former two. ST was the slowest among the four methods as per our experiment.

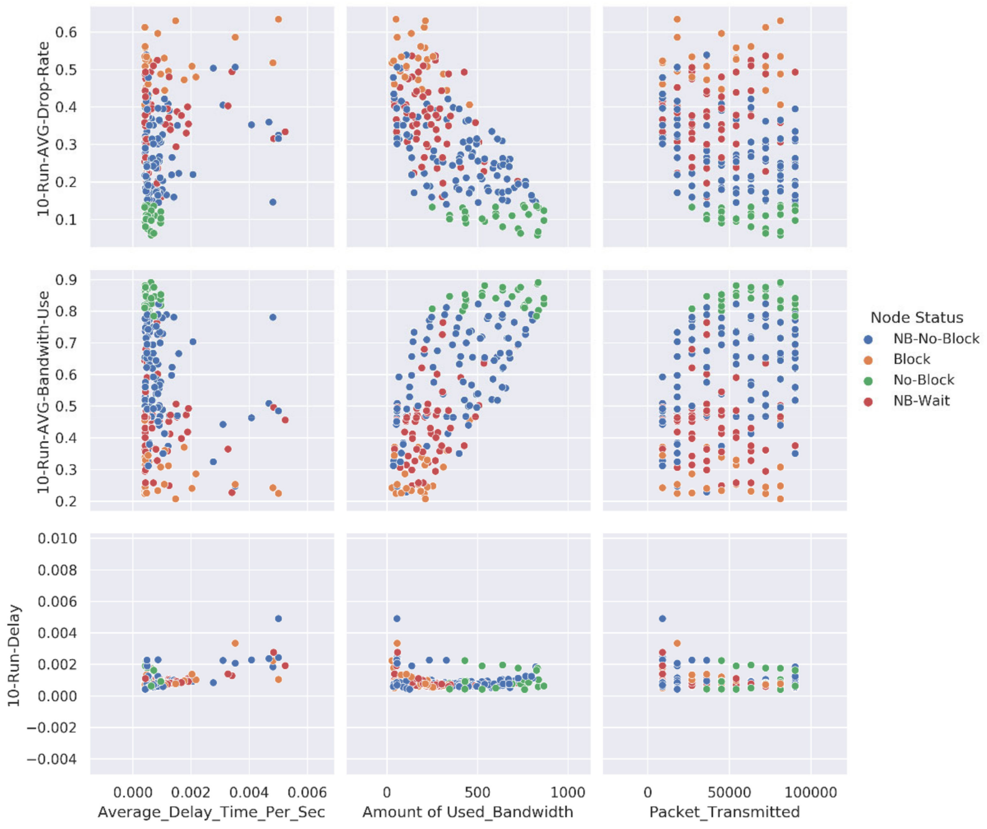

Figure 9. Plot of three attributes against other three.

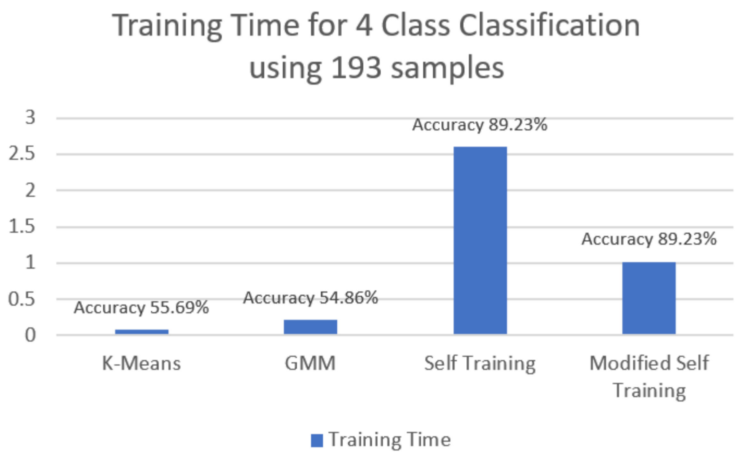

Figure 10. Training time for four class classification.

During the experiment, it was found that when the number of classes increased, more labeled data was required to detect those classes accurately. For each of the cases, we started with a smaller number of labeled data and gradually increased the number until a high accuracy result was achieved. The reasons behind the three different figures of labelled data in detecting three different number of classes are two-fold: firstly, to make a valid comparison among the four SSML methods and secondly, to produce the same accuracy as some related work in order to show a comparative comparison. For example, in case of two-class classification, less than five labelled data produce erroneous result in some SSML methods. Using five data, all the methods showed good results.

In Table 8, two related works [13,14] are listed, which are supervised learning-based classification of the same OBS network dataset used in this paper for detecting BHP flooding attack. These two works are referred here for a comparison with this study because they share the same goal, that is, detecting BHP flooding attack from OBS network dataset using machine learning. Though these works are not based on an SSML approach, they are selected for comparison because there is no SSML based work in the available literature on this dataset as per our study.

Table 8. Related works that used the same OBS dataset.

Works	**ML Method**	**Detection Accuracy (in Percentage) and the Amount of Truly Labeled Instances Used**		
		In Detecting Two Classes (B, NB)	**In Detecting Three Classes (B, NB, PNB)**	**In Detecting Four Classes (Block, No-Block, NB-Wait, NB-No-Block)**
Rajab et al. [16]	Decision tree rule learning	93% accurate using 1075 truly labeled instances	NA	87% accurate using 1075 truly labeled instances
Hasan et al. [17]	Deep neural networks	NA	NA	99% accurate using 1060 truly labeled instances
Based on Kamrul et al. [6]	K-means based SSML	99% accurate using 5 truly labeled instances	87.32% accurate using 21 truly labeled instances	55.69% accurate using 193 truly labeled instances
Based on Kamrul et al. [7]	GMM-based SSML	99% accurate using 5 truly labeled instances	64.98% accurate using 21 truly labeled instances	54.86% accurate using 193 truly labeled instances
Based on Kamrul et al. [8]	Self-training	98.01% accurate using 5 truly labeled instances	90.04% accurate using 21 truly labeled instances	89.23% accurate using 193 truly labeled instances
Based on Kamrul et al. [8]	Modified self-training	98.68% accurate using 5 truly labeled instances	89.09% accurate using 21 truly labeled instances	89.23% accurate using 193 truly labeled instances

In the table, the amount of truly labeled data used by the authors [16,17] and the obtained accuracy is shown. When comparing this with the result shown in this study (Tables 5–7), it reveals that the SSML approaches give superior performance than the SL based methods while leveraging comparatively much smaller amount of truly labeled samples.

5. Conclusions

SSML has an additional advantage over the SL approach. Here, we particularly explored the dataset classification strategy, as it is one of the efficient methods to counter a BHP flooding attack. Collecting an unlabeled dataset from an OBS network is not very difficult. Then, only a few labeled samples are needed for SSML methods to produce a good classification with reasonable accuracy. In this study, four existing SSML approaches were examined. We found that, in the case of two class classification of the OBS network dataset, K-means and GMM-based SSML approach exhibited the best performance among the four SSML approaches in terms of four evaluation metrics. Here, only five truly labeled samples were used. In the case of three class classification of the OBS network dataset, K-means based SSML and ST method performed better than the rest of the methods. Here, only 21 truly labeled samples were used. Lastly, in the case of four class classification of the OBS network dataset, MST method showed superior performance than the rest of the methods in terms of overall evaluation metrics. Here, 193 truly labeled samples were used. The OBS dataset chosen here for examining BHP flooding is very imbalanced when four class classification is concerned. So, to obtain a good performance one may try the over- or under-sampling technique. Another observation is that misclassifying a new observation as false positive or false negative has a different cost. Therefore, one may consider cost sensitive ML techniques [47] to get a realistic accuracy. The clustering-based SSML methods that have been referred and demonstrated, work well on datasets that have normal distribution for each of the possible class-label. In addition, data for each group of labels should be well separated from other groups. Besides, they should be spherical or circular in shape when plotted in a graph. Otherwise, K-means based SSML method may produce poor result. It was found that the GMM based SSML method converged quickly to a local minimum, which was not optimal for the case of 3 class OBS network dataset. Using a better initialization technique for GMM may help it perform better. In the case of modified self-training method, the values for the three thresholds were determined empirically in this study. Although the authors of MST suggested some rules to find those values, the procedure needs more research so that they can be found more quickly and optimally. Besides, optimal selection of classifiers for decision fusion from a set of candidate classifiers needs further attention from the researchers.

Author Contributions: Conceptualization, M.K.H., M.M.H. and M.A.A.D.; methodology, M.K.H. and M.M.H.; software, M.K.H.; validation, M.K.H., M.M.H. and M.A.A.D.; formal analysis, M.K.H.; investigation, M.K.H.; resources, M.K.H. and M.M.H.; data curation, M.K.H.; writing—original draft preparation, M.K.H.; writing—review and editing, M.M.H. and M.A.A.D.; visualization, M.K.H.; supervision, M.M.H. and M.A.A.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is available in a publicly accessible repository that does not issue DOIs. Publicly available datasets were analyzed in this study. This data can be found here: https://archive.ics.uci.edu/ml/datasets/Burst+Header+Packet+%28BHP%29+flooding+attack+on+Optical+Burst+Switching+%28OBS%29+Network (accessed on 2 August 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. De Sanctis, M.; Bisio, I.; Araniti, G. Data mining algorithms for communication networks control: Concepts, survey and guidelines. *IEEE Netw.* **2016**, *30*, 24–29. [CrossRef]
2. Ridwan, M.A.; Radzi, N.A.M.; Abdullah, F.; Jalil, Y.E. Applications of Machine Learning in Networking: A Survey of Current Issues and Future Challenges. *IEEE Access* **2021**, *9*, 52523–52556. [CrossRef]
3. Boutaba, R.; Salahuddin, M.A.; Limam, N.; Ayoubi, S.; Shahriar, N.; Estrada-Solano, F.; Caicedo, O.M. A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities. *J. Internet Serv. Appl.* **2018**, 1–99. [CrossRef]
4. Al-Shargabi, M. The impacts of burst assembly parameters on optical burst switching network performance. *Int. J. Emerg. Trends Eng. Res.* **2020**, *8*, 4916–4919. [CrossRef]
5. Rajab, A.; Huang, C.T.; Alshargabi, M.; Cobb, J. Countering burst header packet flooding attack in optical burst switching network. In Proceedings of the International Conference on Information Security Practice and Experience, Zhangjiajie, China, 16–18 November 2016; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; pp. 315–329.
6. Hossain, M.K.; Haque, M.M. A Semi-Supervised Machine Learning Approach Using K-Means Algorithm to Prevent Burst Header Packet Flooding Attack in Optical Burst Switching Network. *Baghdad Sci. J.* **2019**, *16*, 804.
7. Hossain, M.K.; Haque, M.M. A semi-supervised approach to detect malicious nodes in OBS network dataset using gaussian mixture model. In *Lecture Notes in Networks and Systems*; Springer: Singapore, 2020; Volume 89.

8. Hossain, M.K.; Haque, M.M. Semi-supervised learning approach using modified self-training algorithm to counter burst header packet flooding attack in optical burst switching network. *Int. J. Electr. Comput. Eng.* **2020**, *10*, 4340–4351. [CrossRef]
9. Jayaraj, A.; Venkatesh, T.; Murthy, C.S.R. Loss Classification in Optical Burst Switching Networks using Machine Learning Techniques: Improving the Performance of TCP. *IEEE J. Sel. Areas Commun.* **2008**, *26*, 45–54. [CrossRef]
10. Levesque, M.; Elbiaze, H. Graphical Probabilistic Routing Model for OBS Networks with Realistic Traffic Scenario. In Proceedings of the IEEE Global Telecommunications Conference, Honolulu, HI, USA, 30 November–4 December 2009.
11. Skorin-Kapov, N.; Chen, J.; Wosinska, L. A new approach to optical networks security: Attack-aware routing and wavelength assignment. *IEEE/ACM Trans. Netw.* **2010**, *18*, 750–760. [CrossRef]
12. McGregor, A.; Hall, M.; Lorier, P.; Brunskill, J. *Flow Clustering Using Machine Learning Techniques*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 205–214.
13. Nguyen, T.T.T.; Armitage, G. A survey of techniques for internet traffic classification using machine learning. *IEEE Commun. Surv. Tutor.* **2008**, *10*, 56–76. [CrossRef]
14. Sliti, M.; Boudriga, N. BHP flooding vulnerability and countermeasure. *Photonic Netw. Commun.* **2015**, *29*, 198–213. [CrossRef]
15. Sliti, M.; Hamdi, M.; Boudriga, N. A novel optical firewall architecture for burst switched networks. In Proceedings of the 12th International Conference on Transparent Optical Networks, Munich, Germany, 27 June–1 July 2010.
16. Rajab, A.; Huang, C.T.; Al-Shargabi, M. Decision tree rule learning approach to counter burst header packet flooding attack in Optical Burst Switching network. *Opt. Switch. Netw.* **2018**, *29*, 15–26. [CrossRef]
17. Zahid Hasan, M.; Zubair Hasan, K.M.; Sattar, A. Burst header packet flood detection in optical burst switching network using deep learning model. *Procedia Comput. Sci.* **2018**, *143*, 970–977. [CrossRef]
18. Lopez-Martin, M.; Carro, B.; Sanchez-Esguevillas, A. Variational data generative model for intrusion detection. *Knowl. Inf. Syst.* **2019**, *60*, 569–590. [CrossRef]
19. Kingma, D.P.; Rezende, D.J.; Mohamed, S.; Welling, M. Semi-supervised learning with deep generative models. *Adv. Neural Inf. Process. Syst.* **2014**, *4*, 3581–3589.
20. Li, Y.; Guan, C.; Li, H.; Chin, Z. A self-training semi-supervised SVM algorithm and its application in an EEG-based brain computer interface speller system. *Pattern Recognit. Lett.* **2008**, *29*, 1285–1294. [CrossRef]
21. Wang, B.; Spencer, B.; Ling, C.X.; Zhang, H. Semi-supervised self-training for sentence subjectivity classification. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin/Heidelberg, Germany, 2008.
22. Tanha, J.; van Someren, M.; Afsarmanesh, H. Semi-supervised self-training for decision tree classifiers. *Int. J. Mach. Learn. Cybern.* **2017**, *8*, 355–370. [CrossRef]
23. Sinaga, K.P.; Yang, M.S. Unsupervised K-means clustering algorithm. *IEEE Access* **2020**, *8*, 80716–80727. [CrossRef]
24. Jain, A.K. Data clustering: 50 years beyond K-means. *Pattern Recognit. Lett.* **2010**, *31*, 651–666. [CrossRef]
25. Marwala, T. Gaussian Mixture Models. In *Handbook of Machine Learning*; World Scientific: Singapore, 2018.
26. Xuan, G.; Zhang, W.; Chai, P. EM algorithms of Gaussian mixture model and Hidden Markov Model. In Proceedings of the IEEE International Conference on Image Processing, Thessaloniki, Greece, 7–10 October 2001.
27. Murphy, K.P. Mixture models and the EM algorithm. *Mach. Learn Probabilistic Perspect.* **2012**, *1*, 337.
28. Riloff, E.; Wiebe, J.; Phillips, W. Exploiting subjectivity classification to improve information extraction. *Proc. Natl. Conf. Artif. Intell.* **2005**, *1*, 1106–1111.
29. Zhu, X. *Semi-Supervised Learning Literature Survey*; University of Wisconsin-Madison: Madison, WI, USA, 2005.
30. Geurts, P.; Ernst, D.; Wehenkel, L. Extremely randomized trees. *Mach. Learn.* **2006**, *63*, 3–42. [CrossRef]
31. Kozodoi, N.; Katsas, P.; Lessmann, S.; Moreira-Matias, L.; Papakonstantinou, K. Shallow Self-learning for Reject Inference in Credit Scoring. In *Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2020; Volume 11908. [CrossRef]
32. Blum, A.; Mitchell, T. Combining labeled and unlabeled data with co-training. In Proceedings of the Annual ACM Conference on Computational Learning Theory, Madison, WI, USA, 24–26 July 1998.
33. Chan, T.F.; Golub, G.H.; LeVeque, R.J. Updating Formulae and a Pairwise Algorithm for Computing Sample Variances. In *COMPSTAT 1982 5th Symposium Held at Toulouse 1982*; Caussinus, H., Ettinger, P., Tomassone, R., Eds.; Physica: Heidelberg, Germany, 1982.
34. Bentéjac, C.; Csörgő, A.; Martínez-Muñoz, G. A comparative analysis of gradient boosting algorithms. *Artif. Intell. Rev.* **2021**, *54*, 1937–1967. [CrossRef]
35. Yu, H.F.; Huang, F.L.; Lin, C.J. Dual coordinate descent methods for logistic regression and maximum entropy models. *Mach. Learn.* **2011**, *85*, 41–75. [CrossRef]
36. Kullarni, V.Y.; Sinha, P.K. Random Forest Classifier: A Survey and Future Research Directions. *Int. J. Adv. Comput.* **2013**, *36*, 1144–1156.
37. Chen, T.; Guestrin, C. XGBoost: A scalable tree boosting system. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016.
38. Wang, X.; Li, X.; Ma, R.; Li, Y.; Wang, W.; Huang, H.; Xu, C.; An, Y. Quadratic discriminant analysis model for assessing the risk of cadmium pollution for paddy fields in a county in China. *Environ. Pollut.* **2018**, *236*, 366–372. [CrossRef]
39. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1026–1034.

40. Anaconda for Python. Available online: https://www.anaconda.com/distribution/ (accessed on 17 November 2019).
41. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
42. OBS Network Dataset. Available online: https://archive.ics.uci.edu/ml/datasets/Burst+Header+Packet+%28BHP%29+flooding+attack+on+Optical+Burst+Switching+%28OBS%29+Network (accessed on 9 January 2019).
43. NCTUns Network Simulator and Emulator. Available online: http://www.estinet.com/ns/?page_id=21140 (accessed on 2 August 2021).
44. Pourhashemi, S.M. E-mail spam filtering by a new hybrid feature selection method using Chi2 as filter and random tree as wrapper. *Eng. J.* **2014**, *18*, 123–134. [CrossRef]
45. Mohammad, R.M.; Thabtah, F.; McCluskey, L. An improved self-structuring neural network. In *Trends and Applications in Knowledge Discovery and Data Mining PAKDD 2016*; Lecture Notes in Computer Science; Cao, H., Li, J., Wang, R., Eds.; Springer: Cham, Switzerland, 2016; Volume 9794.
46. Ballabio, D.; Grisoni, F.; Todeschini, R. Multivariate comparison of classification performance measures. *Chemom. Intell. Lab. Syst.* **2018**, *174*, 33–44. [CrossRef]
47. Natarajan, N.; Dhillon, I.S.; Ravikumar, P.; Tewari, A. Cost-sensitive learning with noisy labels. *J. Mach. Learn. Res.* **2018**, *18*, 5666–5698.