T.C.

SELÇUK UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

NETWORK ANOMALY DETECTION USING OPTIMIZED MACHINE LEARNING ALGORITHMS

Tahira KHORRAM

MASTER THESIS

Department of Computer Engineering

June – 2019
KONYA

ACCEPTANCE and APPROVAL OF THE THESIS

Thesis titled as "Network Anomaly Detection Using Optimized Machine Learning Algorithms" prepared by Tahira KHORRAM has been accepted as the MASTER OF THESIS on 27./06./2019 by ......Konya....Selçuk...........................University,..Graduate...School of ..Natural..and....Applied..Sciences.. by unanimously of the jury members from Computer Engineering Department.

Tahira KHORRAM tarafından hazırlanan "Optimize Edilmiş Makine Öğrenmesi Algoritmaları Kullanarak Ağ Anomali Tespiti" adlı tez çalışması 27./06./2019 tarihinde aşağıdaki jüri üyeleri tarafından oy birliği / ~~oy çokluğu~~ ile ....Konya......Selçuk.. Üniversitesi ....Fen....Bilimleri....Enstitüsü......................... Bilgisayar Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jury	**Signature**
Head of Jury Assoc. Prof. Dr. Barış KOÇER	....
Supervisor Assist. Prof. Dr. Nurdan BAYKAN	....Nurdan....
Jury Member Assist. Prof. Dr. Kemal TÜTÜNCÜ	....

I approved the result below.

Director of Institution

DECLARATION PAGE

I, Tahira Khorram at this moment declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

Tahira Khorram
Date: June 2019

YÜKSEK LİSANS TEZİ

OPTIMIZE EDİLMİŞ MAKINE ÖĞRENMESİ ALGORİTMALARI KULLANARAK AĞ ANOMALİ TESPİTİ

Tahira Khorram
Selçuk Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Nurdan BAYKAN

2019, 63 Sayfa

Jüri

Doç .Dr. Barış KOÇER
Dr. Öğr. Üyesi Nurdan BAYKAN
Dr. Öğr. Üyesi Kemal TÜTÜNCÜ

ÖZET

Bilgisayarın ve internetin, günlük hayatta kullanımının artmasıyla birlikte kullanıcılar, bankacılık, iletişim, ticaret gibi önemli ve gizlilik içeren uygulamaları da artık web siteleri üzerinden yapmaktadırlar. Ancak başta kişisel bilgiler olmak üzere uygulamalarda kullanılan bilgilerin güvenliğinin sağlanması giderek hem daha önemli olmakta hem de gün geçtikçe zorlaşmaktadır. İnternet ağında gerçekleşen aktivitelerin izlenerek, sistemdeki olası atak, ihlal ve tehditleri kapsayan saldırıların tespit edilmesi, bu saldırıların en kısa sürede önlenebilmesi için önemlidir. Bu nedenlerden dolayı internet ağlarında Saldırı Tespit Sistemi (Intrusion Detection System-IDS)s, mevcut bilgisayar ağ sistemlerinde birincil gereksinimdir. STS için veri madenciliği ve makine öğrenimi yaklaşımları son birkaç yıldır ağlardaki saldırıların tespiti için yaygın olarak kullanılmaktadır. Makine öğrenmesi kullanılan bu tip STS'lere Akıllı Saldırı Tespit Sistemi (ASTS) (Intelligent Intrusion Detection System-IIDS) adı verilmektedir. Bu tez çalışması kapsamında, ASTS için Destek Vektör Makinesi (DVM) (Support Vector Machine-SVM), K-En Yakın Komşu (K-Nearest Neighbour-KNN) ve Rastgele Orman (RO) (Random Forest-RF) makine öğrenmesi algoritmaları, saldırıların sınıflandırılması için kullanılmıştır. Ancak, makine öğrenmesi algoritmalarının performansı, kullanılan parametre değerlerine bağlıdır. Parametre değerlerinin belirlenmesi amacıyla farklı optimizasyon teknikleri kullanılmaktadır. ASTS için kullanılan sınıflandırıcılardan daha yüksek tespit performansı elde etmek amacıyla Parçacık Sürü Optimizasyonu (Particle Swarm Optimization-PSO) ve Yapay Arı Kolonisi (Artificial Bee Colony-ABC) metasezgisel algoritmaları ile sınıflandırıcılar için önemli parametreler optimize edilmiştir. Önerilen yöntemler, NSL-KDD veri seti üzerinde uygulanmıştır. Elde edilen sonuçlara göre metasezgisel algoritmalar ile parametre optimizasyonu yapılan sınıflandırıcıların hem bilinen (known) hem de bilinmeyen (unknown) ağ saldırılarına ait veri sınıfları üzerinde daha iyi performans gösterdiği kanıtlanmıştır.

Keywords: Anomali Algılama, Akıllı Saldırı Tespit Sistemi (ASTS), Ağ Güvenliği, Makine Öğrenmesi, Metasezgisel Algoritmalar

ABSTRACT

MASTER THESIS

NETWORK ANOMALY DETECTION USING OPTIMIZED MACHINE LEARNING ALGORITHMS

Tahira Khorram
SELCUK UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

Advisor: Assist. Prof. Dr. Nurdan BAYKAN

2019, 63 Pages

Jury
Assoc. Prof. Dr. Barış KOÇER
Assist. Prof. Dr. Nurdan BAYKAN
Assist. Prof. Dr. Kemal TÜTÜNCÜ

ABSTRACT

With the increasing use of computers and the internet in dialy life, users are conforming to use the confidential applications such as banking, communication, and e-commerce through the websites. On the other hand, the security of the information in these applications, especially personal information, is becoming increasingly important. Monitoring the activities taking place in the Internet network, detecting attacks in the system, including possible attacks, violations and threats, are important in order to prevent these attacks. For these reasons, Intrusion Detection System (IDS) is the primary requirement in existing computer network systems. Data mining and machine learning approaches have been widely used for the detection of attacks in networks over the past few years. The IDS that use machine learning algorithms for intrusion detection is called Intelligent Intrusion Detection System (IIDS). Within the scope of this thesis, machine learning algorithms such as Support Vector Machine (SVM), K-Nearest Neighbor (KNN) and Random Forest (RF) are used for the classification of normal network traffics and anomaly network traffics. However, the performance of machine learning algorithms depends on the algorithm parameter values used. Different optimization techniques are used to determine those parameter values. In order to achieve higher detection performance by IIDS, Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) metaheuristic algorithms are used to optimize the classifiers parameters. The proposed methods were applied on the NSL-KDD data set. According to the obtained results, it has been proved that metaheuristic algorithms with machine learning algorithms perform better to classify the network attacks than the machine learning algorithms without parameter optimization.

Keywords: Anomaly Detection, Intelligent Intrusion Detection System (IIDS), Network Security, Machine Learning, Metaheuristic Algorithms

PREFACE

As computer networks become bigger and bigger, network security has become one of the most critical factors for companies to consider. Anything from software, music, and movies to books, games, personal data that posted on social networks, etc. are stolen and copied because malicious individuals breach security.

Big enterprises like Microsoft, Cisco are designing and building software products that need to be protected against foreign attacks. Nowadays they are trying to develop intelligent security appliances that can detect and prevent the attack automatically without being programmed explicitly. Machine Learning, Data Mining Algorithms and tools are there to help in designing intelligent Intrusion Detection System and Intrusion Prevention System.

In this study, we are up to use some standard machine learning techniques to detect networks. Throughout this thesis, we did different experiments to improve the idea of using machine learning techniques for anomaly detection in a network environment.

In fact, I could not have achieved this level of success without my family who has supported me with love and care, my advisor Assist. Prof. Dr. Nurdan (Akhan) Baykan who has provided me valuable advice and guidance, Turkey Scholarships (YTB) who has offered me an opportunity to study in this beautiful country with loving people. Thank you all for your consistent support and love.

Tahira Khorram
KONYA-2019

CONTENTS

SYMBOLS AND ABBREVIATIONS

Symbols:

V	Particle velocity
W	Inertia weight
pbest	Current best position
gbest	Global best position
r1, r2	Random numbers
V	Particle velocity
W	Inertia weight
c1, c2	Learning factors
x_m	Food Source for honey bee
u	Upper level of solution
l	Lower level of solution
x_k	Randomly selected food source
Q_m	Random number between (-1,1)
P	Probablity

Abbreviations

ABC	Artificial Bee Colony
AC	Accuracy
AI	Artificial Intelligence
AIS	Artificial Immune System
AR	Accuracy Rate
ANN	Artificial Neural Networks
BPSO	Binary Particle Swarm Optimization
CI	Computational Intelligence
CFS	Correlation based Feature Selection
DM	Data Mining
DT	Decision Tree
DoS	Denial of Service
DR	Detection Rate
FAR	False Acceptance Rate
FN	False Negative
FPR	False Positive Rate
HIDS	Host Intrusion Detection System
IDS	Intrusion Detection System
IDS-RS	Intelligent Dynamic Swarm based Rough Set
IIDS	Intelligent Intrusion Detection System
INID	Intelligent Network Intrusion Detector
IG	Information Gain
IPS	Intrusion Prevention System
GA	Genetic Algorithms
KNN	K-Nearest Neighbor

MCC	Mathews Correlation Coefficient
ML	Machine Learning
NB	Naïve Bayes
NIDS	Network Intrusion Detection System
PCA	Principle Components Analysis
PSO	Particle Swarm Optimization
PSO-KM	Particle Swarm Optimization with K-Means algorithm
RF	Random Forest
ROC	Receiver Operating Characteristics
R2L	Remote-to-Local
RS	Rough Set
SI	Swarm Intelligence
SMOTE	Synthetic Minority Over sampling Technique
SSO	Simplified Swarm Optimization
WLS	Weighted Local Search
SVM	Support Vector Machine
TPR	True Positive Rate
U2R	User-to-Root
VFDT	Very Fast Decision Tree
WLS	Weighted Local Search

1. INTRODUCTION

Recently, the internet and computer networks have become the inseparable part of our everyday life. A variety of network-based applications have been developed to provide services in many areas such as e-commerce, web services, social media, and enterprises. Internet user demands to store information and their data, in the internet clouds and servers are increasing day by day. A statistics up to 2017 found that there are 20.35 billion devices connected to the internet all over the world. This number will increase up to 30.73 billion devices, through 2020 (Statista, 2017) By connecting more devices to the internet, the risk of unauthorized activity such as data destruction, data modification, and data theft will be increased. The internet and distributed systems are vulnerable to both these internal and external attacks.

Any set of actions that try to bypass the security aspects of a computer system and breach the integrity, confidentiality and availability of the system are defined as "intrusions". Integrity maintains the consistency, accuracy, and trustworthiness of information on any attempt of modification and destruction by malicious. Confidentiality is there to prevent information from reaching wrong people while providing access to authorized ones. And availability makes sure that information and resources are always available to authorized users when they want to use them (Knipp et al., 2002) According to definitions provided above, a computer system considered as a "secure system", if it can protect distributed systems and its resources about these three security tokens (Mukherjee & Sharma, 2012). To protect computer systems from a variety of internet attacks; several types of services such as firewalls, fireboxes, Intrusion Prevention systems and as well as different security protocols are used.

A secure computer system prevents unauthorized access, denial of use and modification of information and resources. Intrusions (also named as anomaly traffic) try to deny the security tokens of a system. Researchers presume that anomaly traffics behavior is a bit different from the normal traffics, and the unknown anomaly traffic patterns are similar to known intrusions (Darigue, Jang, & Zeng, 2009). Based on the mentioned theorem, the detection and prevention of malicious activities or intervention in distributed systems can be considered as data analysis problem. To determine the patterns of anomaly traffic and regular traffic, data mining techniques and machine learning algorithms are the best options. Since a massive volume of network traffic is produced every day; data mining methods are there to process these traffics, determine the samples for each type of network attacks and standard network data. Mostly these methods need not be programmed explicitly. They can handle a large number of data in just a few seconds. Data Mining (DM) and Machine Learning (ML) tools are widely used in Intrusion

Detection Systems (IDS) and also in Intrusion Prevention Systems (IPS). To design an intelligent Network Intrusion Detection System (NIDS), several DM steps and ML algorithms tends to be used. Preprocessing (dimensionality reduction) is the first step of data mining to be done on collected data; then the preprocessed data is transformed into the specific format that can use for by DM-ML algorithms. Several machine learning algorithms such as Decision Trees (DTs), K-Nearest Nearest Neighbor (KNN) and Support Vector Machine (SVM) are there to classify malicious traffic from the benign one. Based on these ML algorithms, a classification model will be built to analyze and divide the existing known traffic and also future unknown network traffic.

In this thesis, three machine learning algorithms to design an intelligent intrusion detection system; K-Nearest Neighbors (KNN), Support Vector Machine (SVM) and Random Forest (RF) algorithms were used for network connection classification. In ML algorithms, appropriate parameters selection improves the performance of classifier algorithms. For this aim, Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) algorithms were used to optimize the parameters of those three ML algorithms and we compared which of these algorithms yield the better overall performance for IDS.

1.1 The aim of the Thesis

This thesis aims to design an Intelligent Network Intrusion Detector (INID). For this purpose, we seeked the DM-ML algorithms to find the efficient ML algorithms to be used by IDS to detect attacks without being explicitly programmed. The detector based on the DM-ML method that is optimized by metaheuristic algorithms could detect the future unknown attacks based on the knowledge it has learned from the training. We have tried different ML algorithms and tune their parameters to determine the efficient algorithm for intrusion detection (ID) in the network and distributed systems.

1.2 Importance of the Thesis

The statistics show that the number of cyber crimes involving the internet is increasing day by day as some reported data breaches in 2016 was 1,093 whereas the number of data breaches in 2015 was 781. Estimated global cost of cyber attacks in 2015 was 4 billion US Dollars, and this amount increased up to 8 billion in 2016. The overall cost of cyber-attacks will be increased up to 2.1 trillion US Dollars by 2020 (Miniwatts, 2017). The solution to these all is an Intelligent IDS that monitors the stream of traffic and helps information systems to deal with the attacks. This system protects the enterprise setup by identifying, logging, reporting and sending alarm whenever there is an attack. In spite of this, the big enterprises like Cisco, Microsoft makes smart security appliances to keep network admins aware of the attacks. But still, attackers attack the systems, and it is increasing day by day. Because of these, researchers have been looking into using machine learning in ID to do a better job than solutions offered by enterprises. There is some literature that proves that the idea of ML-based IDS is practical to develop. For these reason, this study investigated the performance of some different ML algorithms and indicated the most suitable optimized ML algorithm for anomaly detection. The results of the thesis have been showed that applying machine learning based ID has some advantages as follows:

- It can detect unknown attacks without being explicitly programmed.
- It has lower cost of ownership, even tiny businesses can deploy it.
- It is easier to deploy
- It can detect failed attacks
- It can detect the Real-time attacks and gives a quick response
- It does not require additional hardware

1.3 Thesis Outlines

This thesis is organized as follows. Chapter two presents theoretical background. Chapter three contains information about materials and methods about the technologies and algorithms that are used in this study. It provides information about the NSL-KDD dataset, IDS and IDS types, ML algorithms and metaheuristic algorithms. In chapter four, experimental’s result analysises are given. Finally, in chapter five we conclude the thesis and talk about the future works.

2. LITERATURE REVIEW

Several research papers, books, and thesis have been written about how to use machine learning (ML) algorithms to detect network attacks. "Network Anomaly Detection, a Machine Learning Perspective," is a book written in 2014 by Bhattacharyya and Kalita (Bhattacharyya & Sharma, 2013) provides detailed information on anomalies in networked systems and detecting those intrusions using ML algorithms. It presents ML techniques to counter network intrusion under categories such as supervised learning, unsupervised learning, probabilistic learning, and detailed analysis on abilities of these methods. ML methods cannot work without a dataset. And the dataset clearness and format plays a significant role in making a classifier model. A study that has been done in 2015, by Dhanabal and Shantharajah, (Dhanabal & Shantharajah, 2015) applying SVM on Normalized NSL-KDD dataset to detect intrusions in networks. The author uses Correlation based Feature Selection (CFS) method to reduce the detection time and to increase the detection accuracy rate. This paper provides sufficient information about the NSL-KDD dataset itself also. A Master thesis written in 2016 by Voldan (Voldan, 2016) talks about how to use SVM and KNN algorithms to detect network intrusions. NSL-KDD dataset is used for training and testing the proposed methods, and the overall system performance is measured by time consumption, classification accuracy and resource consumption. Voldan did both binary and multi-class classification. In binary, he just classified network traffic into two classes of anomaly and normal types. For this purpose, he used NSL-KDD binary dataset. And definitely, binary classification consumes fewer amounts of resources and time. But it is not very helpful because an anomaly includes different categories of attack with different characteristics. So IDS must be train which type of attacks has been happening. A survey paper was published in 2016 (Buczak & Guven, 2016) provides information on different types of ML algorithms including SVM, to be used for anomaly detection. The complexity of ML/DM algorithms is addressed, discussion of challenges for using ML/DM for cyber security is presented and some recommendations are provided. The study by Ji et al. (Ji, Jeong, Choi, & Jeong, 2016) provides better knowledge about network intrusions and focuses on designing a multi-level network detection method. Mainly, it is composed of three steps as (a) understanding hidden underlying patterns from network traffic data by creating reliable rules to identify network abnormality, (b) generating a predictive model to determine exact attack categories, and (c) integrating a visual analytics tool to conduct an interactive visual analysis and validate the identified intrusions with transparent reasons.

Another most common classification method for IDS is Random Forest (RF). A study written by Zhang et al. (Zhang, Zulkernine, & Haque, 2008) in 2008, addresses the problems in rule-based intrusion detection systems. RF by providing 94.5% accuracy is considered an excellent technique for these systems. In a study (Tesfahun & Bahaskari, 2013) published in 2013, the authors proposed a method as SMOTE (Synthetic Minority Over sampling Technique) to remove the imbalance in the NSL-KDD dataset training classes. Information Gain (IG) is used for feature reduction and RF is used for classifier as IDS framework. The results show that RF with SMOTE and IG is more efficient and effective for designing IDS. Hassan et al. (Hassan, Nasser, & Pal, 2014) in 2014 did research on RF and SVM classifier to find out either they are good algorithms to be used for intrusion detection or not. Their findings show that these algorithms if they applied on an appropriate network dataset will produce outstanding performance. Their evaluation metrics include False Positive Rate (FP), False Negative Rate (FN), Precision and Recall. Farnaaz and Jabbar (Farnaaz & Jabbar, 2016) in 2016 used RF modeling for intrusion detection and RF modeling provides a better result than most of the classification methods in term of detecting anomalies. RF deals with multi-class classification and the performance of the model evaluated regarding accuracy, False Acceptance Rate, Detection Rate (DR), and Mathews Correlation Coefficient (MCC). In the paper, the RF was compared with C4.5 decision tree, the Symmetric Uncertainty of attributes. Roy et al. (Roy, Mittal, & Biba, 2016) proposed a RF method based on the averaging techniques for detection of different intrusion types and the proposed method was compared to SVM and Nearest Centroid classification model. And RF as always performs better than any other methods in their study. So in most cases, the authors recommend using RF as a model for anomaly detection.

Not only SVM and RF, other ML algorithms was also used for intrusion detection. In 2013, Revathi and Malathi compared different ML algorithms. A detailed analysis on the NSL-KDD dataset using various machine learning techniques for intrusion detection (Revanthi & Malathi, 2013) applied RF, SVM and Naïve Bayes (NB) to detect network attacks. For feature reduction, CFS algorithm is used which provide a better result than using all the features for making the model. The classification is binary and Weka data mining tool was used to implement this solution. Among all methods used for anomaly detection, RF presented the best performance.

Determining the effectiveness of a method rely on several criteria such as accuracy, complexity, time for classifying unknown instance within a trained model, understandability of final solution and so on. Therefore, it is impossible to make one specific recommendation which method is the best to be used for detecting network attacks. Depending on the type of IDS, different methods can be selected. Also, parameters of ML algorithms affect the system performance. Because of this, researchers study on the finding the best parameters for the ML algorithms. In 2010, Mulay et al. (Mulay, Devale, & Garje, 2010) proposed a tree-structured multiclass SVM to construct a multiclass intrusion detection system. The integration of decision trees and SVM provide a better result than individual models. They assumed that tree-structured binary SVM could be faster than another method for detecting network anomalies. A study that has been done in Wang et al. (Wang, Li, & Ren, 2010) focuses on using Artificial Bee Colony (ABC) to optimize the parameters of SVM algorithm while the algorithm was used for intrusion detection. In this proposed method, the overall accuracy rate has been improved to 92.7%, and the False Acceptance Rate has been shrunk.

In recent years metaheuristic algorithms have been widely used for optimization. Also, in intrusion detection systems, parameters of classifiers are optimized with these algorithms. In 2010 Tian and Gu (Tian & Gu, 2010) proposed a novel detection framework for detecting anomalies, which combined the idea of unsupervised and the supervised learning methods. Instead of calculating accuracy, ROC (Receiver Operating characteristics) analysis used to evaluate the method performance. Area under the ROC curve (AUC) was a fitness value for each particle and PSO was executed for optimal global parameters of SVM. A boundary movement could help to improve the performance. The best combination of True Positive Rate (TPR) and False Positive Rate (FPR) was achieved after adjusting the offset of the detection function. A study that has been done by Li et al. (Li & Xu, 2011)proposes a K-means clustering algorithm based on Particle Swarm Optimization (PSO) as (PSO-KM). The proposed algorithm has overcome falling into local minima and has relatively good overall convergence, has higher Detection Rate (DR) and lower FAR. And also PSO helps K-means algorithm to show a better result than an original K-means algorithm. According to experimental results, K-means algorithm provides 81% accuracy at detecting Probe attack while PSO-KM provides 96% of accuracy. PSO-KM shows good performance on both known and unknown attacks. In a study that has been done by Malik and Aslam Khan (Malik & Aslam Khan, 2013) , a hybrid classifier based on Binary version of multi-objective PSO (BPSO) and RF algorithm for the classification of PROBE attacks in a network is proposed. PSO is an optimization method which has a strong global search capability and Multi-objective PSO approach is used for feature selection whereas RF, a highly accurate classifier, is used here for classification. The experiments are performed using the well-known KDD99Cup dataset. Aljarah and Ludwig (Aljarah & Ludwig, 2013) proposed an intrusion detection system based on a parallel PSO algorithm using the MapReduce methodology. MapReduce method helps to solve the management problem in large-scale network traffic. In this paper they have shown that the intrusion detection system can be parallelized efficiently by MapReduce methodology. Their experiment is done on real network dataset. They recommend using large training dataset to train an anomaly detection model to avoid random sampling effects. Larger training dataset leads to better detection rate and low false alarms. In study written in 2016 by Aburomman and Bin Ibne Reaz (Aburromman & Bin Ibne Reaz, 2016), proposed a novel ensemble construction method that uses PSO generated weights to create an ensemble of classifiers (SVM and KNN) with better accuracy for intrusion detection. The authors stated that weights generated by metaheuristic could yield improved accuracy for intrusion detection system.

Not only PSO, also Simplified Swarm Optimization (SSO) and ABC are used for optimization of classifiers for intrusion detection systems. Chung and Noorhaniza (Chung & Noorhaniza, 2012) in 2012, propose a new hybrid intrusion detection system by using Intelligent Dynamic Swarm based Rough Set (IDS-RS) for feature selection and SSO with Weighted Local Search (SSO-WLS) for intrusion data classification. SSO-WLS shows that it can improve the performance of anomaly detection technique. The main idea of WLS is to improve the searching process in SSO rule mining by weighting the three predetermined constants (e.g., CW, cp). The position update strategy for each particle is performed using the weighted predetermined constant value. Once the gbest has been obtained during the local search of the pbest, the WLS process will be stopped. Thus, WLS is capable of supporting SSO during the search mechanism. In order to improve the efficiency of SSO-WLS they have taken advantage of IDS-RS to eliminate the unnecessary features during the preprocessing phase of network intrusion detection system. IDS-RS helps extract the six most relevant features from 41 features of the dataset to improve the proposed system detection rate and accuracy rate. And proposed method provides excellent performance. Revathi and Malathi (Revanthi & Malathi, 2013) in 2013 wrote a paper that its main focus was on detailed analysis on NSL- KDD dataset and proposed a new technique of combining swarm intelligence SSO and RF for feature selection and reduction. SSO is used to find the more convenient set of attributes for classifying network intrusions and RF is used as a classifier. In the study, network traffic is classified into normal and four type’s attacks as DoS, Probe, U2R, and R2L. The proposed method provides better results in both all features used case and data reduction case (Enache & Patriciu, 2014). Proposed IDS model based on IG feature selection with SVM classifier. For SVM the Cost and Gamma parameters are optimized by PSO and ABC algorithms. The PSO-SVM accuracy is 98.6% on known attacks test dataset and ABC-SVM accuracy is 98.8% on known attacks test dataset where the SVM accuracy is 89% which is improved 9.6% by PSO optimization and 9.8% by ABC optimization.

Besides them, there is some literature on Swarm Intelligence and Computational Intelligence used for Intrusion Detection. A survey that is done by Wu and Banzhaf (Wu & Banzhaf, 2010) in 2010, provides an overview of the research progress in applying Computational Intelligence methods to the problem of intrusion detection. The scope of this review will contain core methods of CI, including Artificial Neural Networks, Fuzzy Systems, Evolutionary Computation, Artificial Immune Systems (AIS), SI and Soft Computing. The research contributions in each field are systematically summarized and compared, allowing us to define existing research challenges clearly and to highlight promising new research directions. Kolias at al. (Kolias & Kambourakis, 2011), explore the reasons that led to the application of SI in intrusion detection and present SI methods that have been used for constructing IDS. A significant contribution of their work is also a detailed comparison of several SI-based IDS regarding efficiency.

A master thesis study has been written by Alwan Hussian in 2014 (Alwas Hussain, 2014) proposes a new algorithm by the name of Very Fast Decision Tree (VFDT) algorithm that is used to build a classifier for intrusions. The experimental results of this thesis achieved a high classification accuracy rate of 93% by using their proposed method. It is the highest performance compared to all another algorithm except the Genetic Programming algorithm where it has a higher detection rate of 98%. The speed of training "building and testing" did not exceed 39.88 seconds by using VFDT algorithm, whereas it took long time for others systems. "A Real-time Intrusion Detection System Based on PSO-SVM" paper a hybrid-PSO feature selection algorithm is proposed in which SVM parameters are elected by SPSO. The proposed IDS system feature selection algorithm consists of search strategy, BPSO and evaluation criterion. The proposed system is not only successful in the selection of best features for the algorithm but also in providing a higher detection rate. This paper indicates that a combination of SVM and PSO algorithms provide a higher detection rate for intrusion detection systems (Wang, Hong, & Ren, 2009).

3. MATERIAL and METHODS

3.1 Dataset Description

KDD CUP99 and NSL-KDD are the two datasets used to create an intrusion detector which is able to determine whether network traffic is a regular traffic or harmful traffic (Witten, 2011). The statistical analysis showed that there are significant problems in the KDD CUP99 dataset which profoundly affects the performance of the system, and result in a very poor estimation of anomaly detection approaches. To solve this inheritance issue, the NSL-KDD dataset is proposed. This data set is an improvement of the old KDD CUP99 dataset (Data, 1999)The advantages of NSL-KDD dataset is as follows:

1) No redundant records in the train set,

2) No duplicate record in the test set which has better reduction rates,

3) The number of selected records from each problematic level group is inversely proportional to the percentage of records to the origin KDD dataset.

The NSLKDD dataset has been used by many researchers since it was publically available and contained a lot of data points. One of the many reasons of using this dataset is all network data is labeled as malicious or regular traffic. The data points that are malicious are tagged with the kind of an attack it is supposed to simulate at the end of the line in each line of traffic (Dhanabal & Shantharajah, 2015).

The training dataset is made up of 22 different attacks out of the 37 present in the test dataset (Table 3.1 and Table 3.2) (Revanthi & Malathi, 2013). The known attack types are those present in the training dataset while the novel attacks are the additional attacks in the test dataset, not available in the training dataset. The attack types are categorized into four groups described below: Denial of Service (DoS), Probe, User-to-Root (U2R) and Remote-to-Local (R2L).

1) Denial of Service (DoS): Attackers tries to attack legitimate users from accessing targeted computer systems, services and other network resources. Successful DoS attack may cut off access for millions of authorized users from the intended resources usage can lead to severe problems in the entire network.

2) Probe: It is an action taken or an object used to learn something about gaining information about the state of the net. Then the attackers use this information to identify the vulnerabilities of the system for launching attacks against the network and services.

3) User-To-Root (U2R): Attacker has local access to victim system, and tries to gain super user privileges. This type of attack is challenging to distinguish from the regular traffic network because it attempts to obtain the system's super user privilege like the daily traffic.

4) Remote-To-Local (R2L): Attackers tries by sending packets to a remote machine or host over a network. This type of attack is similar to U2R; the only difference is that the U2R tries to gain the control of super user and this attack attempts to acquire access to the machine either via super user privilege or normal user privilege.

Groups of attack types in training dataset can be seen in Table 3.1.

Table 3.1 Attacks types in training dataset

Groups of attacks	**Attack Types (22)**
DoS (6 item)	Back, Land, Neptune, Pod, Smurf, Teardrop
Probe (4 item)	IPsweep, Nmap, Portsweep, Satan
R2L (8 item)	Ftp write, Guess passwd, Multihop, Phf, Imap, Spy, Warezclient, Warezmaster
U2R (4 item)	Buffer overflow, Loadmodule, Perl, Rootkit

Also, groups of attack types in test dataset can be seen in Table 3.2.

Table 3.2 Attacks types in testing dataset

Groups of attacks	**Attack Types (37)**
DoS (10 item)	Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Processtable, Mailbomb
Probe (6 item)	IP sweep, Nmap, Portsweep, Satan, Mscan, Saint
R2L (15 item)	Ftp_write, Guess_passwd, Multihop, Phf, Imap, Spy, Warezclient, Warezmaster, Named, Xlock, Xsnoop, Sendmail, Worm, Snmpguess, Snmpgetattack
U2R (8 item)	Perl, Rootkit, Loadmodule, Buffer overflow, Http tunnel, Ps, SQL attack, Xterm

As seen in Tables 3.1 and 3.2 above, some attacks are found in the testing set that is not included in the training set. The additional attacks that are not included in training set share some characteristics with attacks in training set. So that the system builds its model to defend against possible unknown attacks as well as known attacks using these additional attacks. For example, Mailbomb is an attack in the testing set that is not included in the training set but it falls under DoS attack group and shares some characteristics with Neptune or Smurf attack that are involved in the training set and the system already knows about them. Based on the training that is given to the system, it will recognize the unknown attacks also. In real life there are many attacks that we are unaware of, the system cannot be trained based on all possible attacks. We give the system samples; the system makes a model from those examples and inspects the next unknown attack based on the training it has taken previously. Therefore the testing set has been included additional attack types that are not included in the training set.

The NSL-KDD dataset for each record has a total number of 42 columns, here 41 columns are 41 features of the dataset and the last column determine the group of the data record assigned to each either as anomaly means any attack types or as standard. These 41 features are then categorized into four categories as given above (Dhanabal & Shantharajah, 2015). The protocol type, service and flag attribute types are symbolic and the rest attributes are continuous as seen in Table 3.3.

Table 3.3 NSL-KDD dataset features description

Feature No.	Feature Name	Feature Explanation
Basic features of each network connection vector (9 item)		
1	Duration	Lenght of the time duration of the connection
2	Protocol_type	Protocol used in the connection
3	Service	Service used in destination network
4	Flag	Status of the connection as Error or Normal
5	Src_bytes	Number of data bytes transferred from source to destination in single connection
6	Dst_bytes	Number of data bytes transferred from destination to source in single connection
7	Land	If source and destination IP and port numbers are equal then; this variable takes value 1 else 0
8	Wrong_fragment	Total number of wrong fragments in the connection
9	Urgent	Number of urgent packets in the connection
Content related features of each network connection vector (13 item)		
10	Hot	Some ‘hot’ indicators in the content such as: entering a system directory, creating programs and executing them.
11	Num_failed_logins	Count of failed login attempts
12	Logged_in	Login status: 1 if successfully logged in, 0 otherwise
13	Num_compromised	A number of “compromised conditions.”
14	Root_shell	1 if root shell is obtained, 0 otherwise
15	Su_attempted	1 if so root command attempted; 0 otherwise
16	Num_root	Number of root accesses or number of operations performed as a root in the connection
17	Num_file_creations	Number of file creation operation in the connection
18	Num_shells	Number of shell prompts
19	Num_access_files	Number of operations on access control files
20	Num_outband_cmds	Number of outbound commands in an FTP session
21	Is_hot_login	1 if the login belongs to the “hot” list, e.g. root or admin; 0 else
22	Is_guest_login	1 if the login is a guest“ login; 0 otherwise
Time related traffic features of each network connection vector (9 item)		
23	Count	Number of connections to the same destination host
24	Srv_count	Number of connections to the same service
25	Serror_rate	The percentage of connections that have activated the flag (4th feature) among the connections aggregated in the count (23rd feature)

26	Srv_serror_rate	The percentage of connections that have activated the flag (4th feature) among the connections aggregated in Srv_count (24th feature)
27	Rerror_rate	The percentage of connections that have activated the flag (4th feature) , REJ among the connections aggregated in the count (23rd feature)
28	Srv_rerror_rate	The percentage of connections that have activated the flag (4th feature) , REJ among the connections aggregated in Srv_count (24th feature)
29	Same_srv_rate	The percentage of connections that were to the same service, among the connections aggregated in the count (23rd feature)
30	Diff_srv_rate	The percentage of connections that were different service, among the connections aggregated in the count (23rd feature)
31	Srv_diff_host_rate	The percentage of connections that were different destination machines, among the connections aggregated in Srv_count (24th feature)
Host based traffic features in a network connection vector (10 item)		
32	Dst_host_count	Number of connections having the same destination host IP
33	Dst_host_srv_count	Number of connections having the same port number
34	Dst_host_same_srv_rate	The percentage of connections that were to the same service, among the connections aggregated in dst_host_count (32nd feature)
35	Dst_host_diff_srv_rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32nd feature)
36	Dst_host_same_src_port_rate	The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count (33rd feature)
37	Dst_host_srv_dif_host_rate	The percentage of connections that were to different destination machines,among the connections aggregated in Dst_host_srv_count (33rd feature)
38	Dst_host_serror_rate	The percentage of connections that have activated the flag (4th feature) among the connections aggregated in dst_host_count (32nd feature)
39	Dst_host_srv_serror_rate	The percentage of connections that have activated the flag (4th feature) among the connections aggregated in dst_host_srv_count (33rd feature)
40	Dst_host_rerror_rate	The percentage of connections that have activated the flag (4th feature) among the connections aggregated in dst_host_count (32nd feature)
41	Dst_host_srv_rerror_rate	The percentage of connections that have activated the flag (4th feature) among the connections aggregated in dst_host_srv_count (33rd feature)
Class of the data (Attack type)		
42	Class	Normal, DoS, Probe, R2L, U2R

NSL-KDD dataset concludes with different data files, below shows statistical observations for data files. This NSL-dataset is used in our experiments for this thesis.

KDDTrain+: The full NSL-KDD train set with binary labels and attack-types labels. It includes 125,973 records. It was used for training the classifiers in the thesis.

KDDTest +: The full NSL-KDD test set with binary labels and attack-types labels. It includes 22,544 records. It includes the types of attacks that are not in the training set and named as "unknown attack test dataset". It was used for unknown attack testing in the thesis.

KDDTrain+_20%: The full NSL-KDD 20% train set with binary labels and attack-types labels. It includes 25,192 records. It first includes the known types given in the training set and named as "known attack test dataset". It was used for known attack testing in the thesis.

KDDTest+_21: The 50% of NSL-KDD test set with binary labels and attack-types labels. It includes 11,850 records. It is a binary test set. In the thesis, we made multi classification so it is not used in the thesis.

In the following, Table 3.4, Table 3.5 and Table 3.6 show the number of individual records in normal and four types attack in NSL-KDD datasets, both training and test set.

Table 3.4 Number of Instances in Training Dataset (KDDTrain+)

Normal	67343
Dos	45927
Probe	11656
R2L	995
U2R	52
Total	125973

Table 3.5 Number of Instances in Unknown Test Dataset (KDDTest+)

Normal	9711
Dos	7458
Probe	2421
R2L	2754
U2R	200
Total	22544

Table 3.6 Number of Instances in Known Test Dataset (KDDTrain+20%)

Normal	13449
Dos	9234
Probe	2289
R2L	209
U2R	11
Total	25192

3.2 Intrusion Detection System (IDS)

Intrusion detection is a tool for monitoring and evaluating the incidents happening in an "Information Technology" system to detect signs of security problems where the network is made up of ordinary and attack traffics (Ganapathy et al., 2013).

"Intrusion Detection System" (IDS) provides a wall of defense that prevents the attacks on computer systems on the internet and it can detect different types of attacks on network and computer systems where traditional firewalls cannot perform well. Nevertheless, IDS used to be explicitly programmed, so it just could detect known attacks. Recently, researchers are up to design IDSs that work without being explicitly programmed and identify both known and novel attacks. These types of IDSs are called "Intelligent Intrusion Detection System" (IIDS) (Ganapathy et al., 2013) .

Intrusions or anomalies are network events that deviate from standard and usual behavior and are suspect from the security perspective. It is essential to detect abnormalities in the network because anomaly detection analyzes the fraud and faults present in the system, and so that IDS has the capability of identifying the threats as they arise. As a result, IDS allows network managers to react immediately to the problems before it affects the network. The process of intrusion analysis can be divided into four parts (lappas & Pelechrinis, 2004):

1) Preprocessing: In this phase, the collected data transformed into a format that can be used by classifier algorithms. The form can be canonical or could be a structured format.

2) Analysis: After the data is preprocessed, all records of the data will be analyzed, compared to the knowledge base. These records can be considered as an intrusion or can drop as a typical event.

3) Response: When the data record is logged as an intrusion, a reaction is initiated. The answer contains an alert and information about the invasion.

4) Refinement: This phase is responsible for the exactness of the intrusion.

There are two main types of IDSs. The Host-based Intrusion Detection Systems (HIDS) is the first one which resides on a single host and monitors all the events for suspicious activity. The other category is the Network-based Intrusion Detection Systems (NIDS) which located on the network, and are designed to monitor network traffic (Ahmed, lisitsa, & Dixon, 2011).

There are two main detection techniques used in IDSs: the misuse and the anomaly detection techniques. When the network administrators apply a written rule-set to detect malicious traffic, these types of systems is called "Misuse Based Detection Systems". These rules can be written based on known facts like IP addresses, content in payload and URL. There are many standard rule-sets available and most of the software comes with some basic rules; but to be able to keep up with the latest threats, the rule-sets in systems need to be updated on a regular basis (Butun, Morgera, & Sankar, 2013).

In the misuse-based detection, attack patterns or signatures are identified and represented in such a way that the system can match these patterns with the log files or network traffic (Ahmed et al., 2011).

Typical software programs that uses misuse based detection are “Snort” and “Suricata”. These two packages are widely used all over the world and very useful when used correctly and kept update (Darigue et al., 2009). Snort is a publically available, Open Source Network Intrusion Detection Software (NIDS) created by Martin Roesch in 1998. It detects anomalies based on a set of written rules (Martin, 1998). Suricata also like Snort is free open source software that inspects network traffic using a robust and extensive rule and signature language for detecting complex threats in networks. The Suricata engine can detect and prevent network intrusions, and monitor security events (OISF, 2010).

The system that is a way to identify irregular behavior in the network or the improper traffic within a network is “Anomaly Based Detection System”. In this system, intrusions are identified as unusual behaviour that differs from the normal behaviour of the monitored system (Ahmed et al., 2011). How the IDS knows that there is improper traffic is based in measurements done before deployment of the IDS, this tests or training sets are used to simulate the traffic that is expected to be normal within the network environment. These measurements are then the basis for what "normal" traffic should look like, and if the traffic deviates from the usual traffic, there will be generated alerts based on the traffic. The training sets also used to simulate malicious traffic so that it will recognize the patterns form known threats and attacks (Ahmed et al., 2011)

The anomaly-based method is good at identifying new or unusual attacks mainly sweeps attacks and probes attacks. Therefore, it gives early warnings of possible anomalies, because probes and scans are the ancestors of all network outbreaks. And this implies equally to any new service installed on any item of hardware.

3.3 Machine Learning

Computers have been used to solve a variety of problems and doing different types of tasks that human experts are unable to obtain satisfactory solutions. recently, every hour of the day millions of Gigabytes of information are processed in a short period. And there is a need for sophisticated and powerful algorithms that learn by the computers from the given knowledge and make decisions about an unknown event without being explicitly programmed. These robust algorithms that can determine are studied in the field of machine learning. Machine learning gives ability to computers to learn from given data, make a model of the given data and predict the future upcoming events using its knowledge. Machine learning is a research field usually considered to be part of Artificial Intelligence. This consideration is due to the fact the machine should be able to make decisions (Negnevitsky, 2005). Sometimes it considers as part of data mining. Machine Learning can be used in different areas such as:

* Fraud detection and prevention
* E-commerce applications
* Data management applications
* Improved customer segmentation
* Real-time advertisements
* Email and spam filtering
* Web search filtering
* Weather forecast prediction
* Online recommendation offers
* Intrusion detection and prevention systems

As it is evident from the list above, there is a wide area that machine learning can be used. Every day the list is getting bigger and bigger, as machine learning applications are getting faster and more powerful.

Several types of machine learning methods are there to help industries, businesses and researchers to automate their systems to handle their issues, analyze the problems, find quick solutions for the problems and keep the system up-to-date without being explicitly programmed. Machine learning techniques can improve their performance by analyzing the problem by solving a repeated instance of that matter. These methods have different usage areas, as image processing, optimization, information retrieval, natural language processing and etc.

Within the field of machine learning, there are two main types of tasks: supervised, and unsupervised. The main difference between the two types is that supervised learning is done using a prior knowledge of what the output values for our samples should be. Therefore, the aim of supervised learning is to learn the best function that gives the best relationship between input and output observable in the data. Unsupervised learning, on the other hand, does not have labeled outputs, so its goal is to infer the natural structure present within a set of data points (NG, 2017).

Selecting the appropriate algorithm is a crucial part of every machine learning related projects. When doing machine learning, algorithms helps to decide what to do with the information gathered. It is used as a way of thinking for ML (Roughgarden, 2017). There are different types of ML algorithms developed for many various purposes. The most widely used supervised machine learning algorithms are Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbors (KNN) and Artificial Neural Network (ANN).

3.3.1 K-Nearest Neighbors (K-NN)

K-Nearest-Neighbor (K-NN) is one of supervised machine learning algorithms that are very simple to understand and is one of the most typical algorithms used for classification. Even it is listed as one of top ten widely used machine learning algorithms and provides good accuracy when used with datasets with many different input a few variables. Its applications range from computer vision to DNA to complex computer geometry, graphs and so on. In many papers published K-NN is a very common algorithm because of simplicity and could be applied to different types of datasets (Peterson, 2009).

K-NN works based on minimum distance from the query instance to the training sample to determine the K nearest neighbors. After K-NN is gathered, the majority vote of the nearest neighbors determines what class the new instance will be classified to. For example, if k=3, it will look the three nearest neighbors and determine the class for the new instance (Figure 3.1).

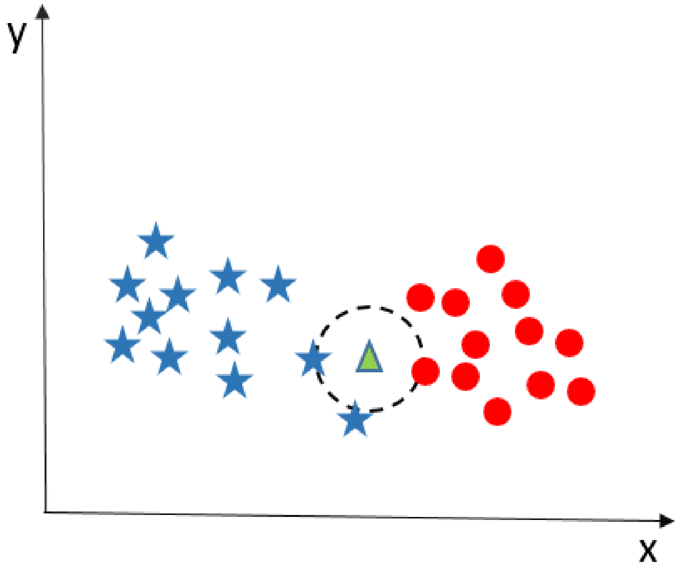


Figure 3.1 K-NN algorithm works with two sets according to 3 nearest neighbors (k=3)

K-NN classifier usually works based on the Euclidean Distance (Ahmed et al.), which is the distance between the test sample and a specified training sample. This ED could also be explained by using an equation (Liao & Vemuri, 2002) given in Eq. 3.1

$$dist((xy),(a.b)) = \sqrt{((x-a)^2} + (y-b)^2 \qquad (3.1)$$

(x,y) and (a,b) are the two points compared in Euclidean n-space. The distance (d) from (x,y) to (a,b) is calculated by the equation 3.1. There is also some other distances such as Hamming, Manhattan or Minkowski distance.

3.3.2 Support Vector Machine (SVM)

Support Vector Machine is a supervised ML algorithm which can be used for classification and regression problems. It is widely used in security software such as network anomaly detection. In this algorithm, each data item is plotted as a point in n-dimensional space. With the value of a particular coordinate, then classification is performed by finding the hyperplane that differentiates the classes very well. This algorithm is simple to apply and provides an excellent result if used correctly. It means to identify the right hyper-plane.

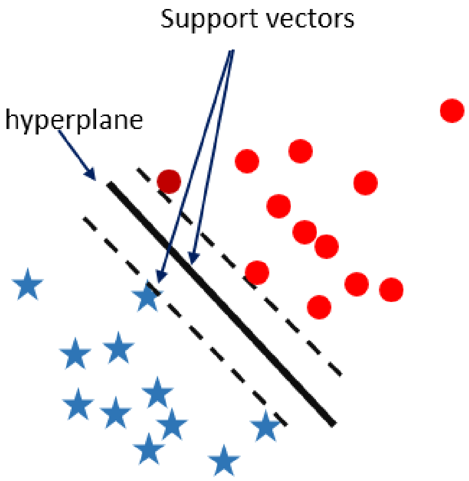


Figure 3.2 SVM linearly separable case (Manning, Raghavan, & Schutze, 2009)

Figure 3.2 displays how SVM algorithm works with two binary separable binary sets, which here represented in blue stars and red circles. In the figure, there are three lines. The one in the middle is the “hyperplane”, and the two others are what often known as “error margin” or “margin”. If there are any data points within this error margin, the points could be classified as the opposite of what it should have been. As shown in Figure 3.1, there are no data points within the margin, and therefore all of the red circles will be classified as class1, and the blue stars are classified as class2. This instance shows how SVM algorithm works (Manning et al., 2009).

3.3.3 Random Forest (RF)

Random forest is unsurpassable in accuracy among the current data mining algorithms, especially for large datasets with many features (Ganapathy et al., 2013).

Random forest establishes by a different bootstrap pattern from the initial data formed, a new instance that needs to be classified is put down each of the trees in the forest for classification. After then each tree gives a vote that indicates the tree's decision about the class of the object and the forest chooses the class with the most votes for the object (Figure 3.3).

In the Random Forest algorithm, there is no need for cross-validation. Since each tree is established using the bootstrap pattern, almost one-third of the cases are left out of the bootstrap samples and not used in training. These cases are called Out Bag (OOB) cases. These OOB cases are used to obtain a run-time unbiased estimate of the classification error as trees are added to the forest (Zhang et al., 2008).

Although random forest has very high accuracy, they are very complex and challenging to be interpreted. For example, understanding how a random forest model approves or denies a load could involve sifting through thousands of finely-tuned decisions. Nevertheless, random forest models are popular due to their higher accuracy and low computation expense. They are used for a wide variety of applications including customer segmentation and network anomaly detection (Eulogio, 2017).

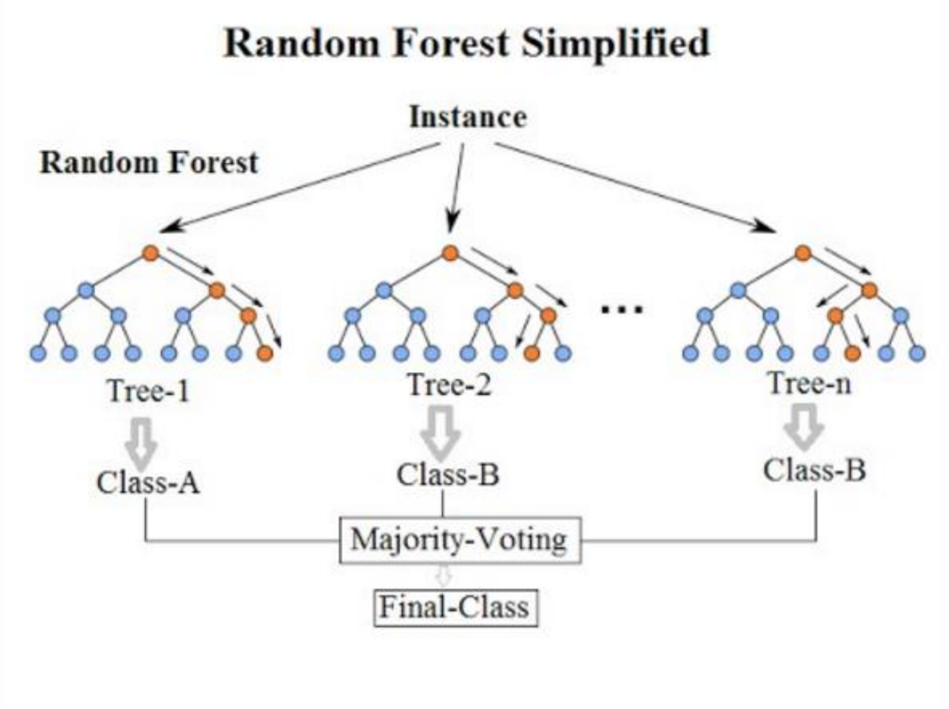


Figure 3.3 Working structure of Random Forest (Koehrsen, 2017)

3.4 Feature Selection

Feature selection is the procedure of picking a subset of the terms occurring in the training set and using only this subset as features in classification for supervised learning algorithms. Feature selection is an essential step in high dimensionality data mining techniques. Feature space of a classification process is a crucial factor that affects the overall performance of a system. In a dataset, there can be different types of features as relevant features, redundant features and irrelevant features. The independent and correlated features degrade the performance and confuse the classifier. To enhance the system performance, feature reduction is an essential task (Ahmad, 2015).

There are lots of feature reduction methods. Such as Chi-square test, Correlation criteria, Wrapper methods, Sequential Selection Algorithms, Embedded methods, Mutual Information and etc (Kumbhar & Mali, 2016)Also, Principle Components Analysis (NLPCA) is one of the most popular feature selection algorithms used to reduce the dimension of training and test datasets for designing a well-performed system.

PCA is a dimensionality reduction tool that is used to reduce a broad set of data records to a small number data which is more meaningful and usable. PCA changes redundant feature into orthogonal features. This means it combines correlated features into one feature space. So no two elements contain the same information about the data record (Fig. 3.4). It can be helpful to reduce the original feature space to a lower number of features before feeding the data as a training data or test data to the ML classifier. Indeed it can reduce the computational cost of the system tremendously.

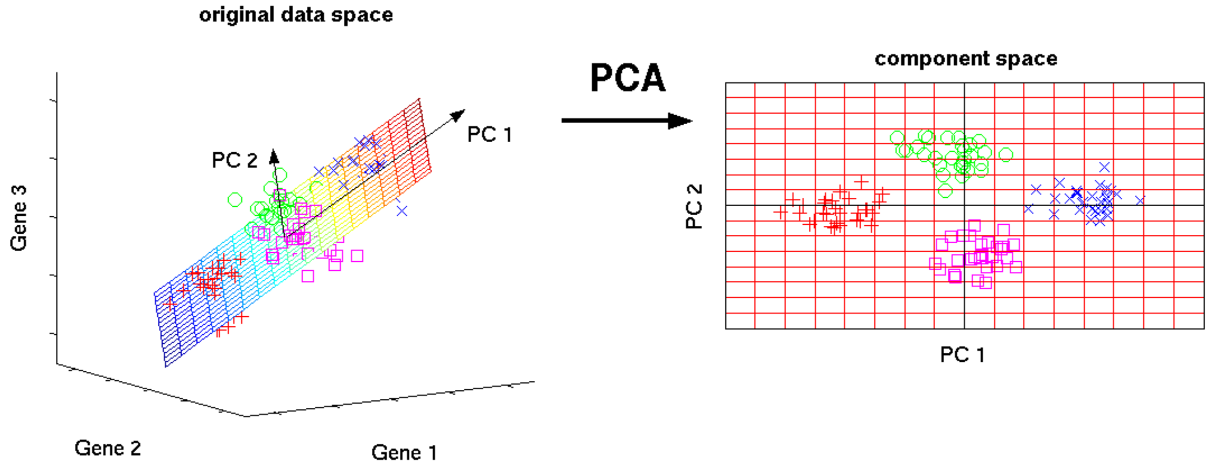


Figure 3.4 İllustration of PCA (NLPCA, 2018)

3.5 Metaheuristic Optimization Algorithms

As money resources and time are always limited, the optimal utility of these available resources is crucially important. From engineering design to economics, from holiday plans to the Internet routing, optimization is everywhere in real-world. Most real-world optimizations are nonlinear and under various complex constraints. Different objectives are often conflicting. Even for a single objective, sometimes, optimal solutions may not exist at all and finding an optimal or even sub-optimal solution is not an easy task (Yan, 2011). In short, optimization can be considered as a minimization or maximization problem to find the best solution under different constraints. Metaheuristic optimization solves these problems using metaheuristic algorithms.

Optimization algorithms are generally classified as deterministic or stochastic. If an algorithm works in a deterministic mechanical manner without any random nature, it is called deterministic. On the other side, if there is some randomness in the algorithm, in each execution, "stochastic" algorithms usually finds different solutions, even though the same initial point is used. Genetic algorithm, PSO, ACO are good examples of stochastic algorithms (Yan, 2011). The algorithms with stochastic items are referred as "Metaheuristics".

Two significant elements of any metaheuristic algorithms are “exploitation” and “exploration” (Blum & Roli, 2003). Exploration means to generate diverse solutions to explore the search space on a global scale, while exploitation means to focus the search in a local region knowing that a current proper answer is found in this area. A right balance between exploitation and exploration should be seen during the selection of the best solutions to enhance the percentage of algorithm convergence (Blum & Roli, 2003).

3.5.1 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is a population-based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995. It is inspired from the behavior of animal’s societies that don't have any leader in their group or swarm, such as the social behavior of bird flocking or fish schooling (Yan, 2011). Since it was developed, PSO has attracted lots of attention and has been applied to almost all optimization problems such as design, scheduling, security applications and so on. PSO is a robust and simple algorithm. Researchers have been developed lots of PSO variants and also PSO hybrid by combining PSO and any other metaheuristic or machine learning algorithms.

PSO algorithm follows the scenario of a group of birds searching for food randomly in an area. There are some foods in that area but only one piece has the more quality than the others. All the birds do not know where the foods are. But they do know how far the foods are in each iteration. The efficient way to find the more quality food is to follow the nearest bird to that food. PSO learn from this scenario and solve optimization problems quickly and efficiently (Xiaohui, 2006).

In PSO, every single solution which is a bird in the search area is called "Particle". All of the particles have fitness values which are examined by the fitness function to be optimized and have velocities which direct flying of the particles. The particles fly through the problem area by following the current optimum particles. First, PSO is initialized with a category of random solutions and in the next steps generations are updated for searches the optima. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. This value is called "pbest". Another "best" value that is recorded by the particle swarm optimizer is the best value, gained so far by any particle in the population. This best value is a "global best" and called "gbest". When a particle takes part of the community as its topological neighbors, the best value is a local best and is called Ibest. After finding the two best values, equation 3.2 updates particles velocity and equation 3.3 updates particle’s position.

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 * r_{1i} * (p_{id} - x_{id}^{t}) + c_2 + r_{2i} * (p_{gd} - x_{id}^{t}) \tag{3.2}$$

$$v_{id}^{t+1} = x_{id}^{t} + v_{id}^{t} \tag{3.3}$$

Where *t* shows the *th* iteration of PSO, *d* indicates search space dimension, *w* is inertia weight and c_1 and c_2 are acceleration factors, r_1 and r_2 are random numbers between (0,1), P_{id} and P_{gd} are pbest and gbest, respectively.

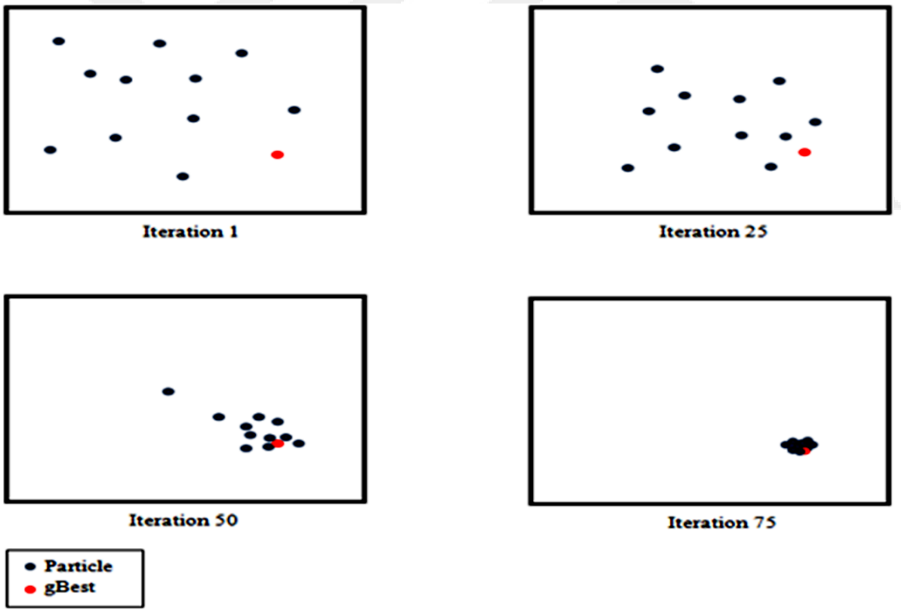


Figure 3.5 Particle Swarm Optimization movement towards global optima over iteration numbers (Ab Wahab, Nefti-Meziani, & Atyabi, 2015)

Figure 3.5 illustrates the PSO algorithm output over iterations. In the first iteration, all particles spread out in order to find the best solution (exploration) and then each particle is evaluated. The best solutions are found with respect to neighborhood topology and the personal and global best particles for each member of the swarm are updated. The convergence would be achieved through attracting all particles towards the particle with the best solution.

3.5.2 Artificial Bee Colony (ABC)

ABC algorithm is a population based optimization method that was developed by Karaboga in 2005. It is designed to optimize the continuous numerical problems .The algorithm animates the social behavior of honey bee colonies (Karaboga & Basturk, 2007). The algorithm consist of three components: employee bees find food sources, store information about the quality of the food and share the information with others. The onlooker bees receive information about food source and choose the food source with high quality. The last components are scout bees, they start working when the existing food sources are over, and they try to find new food origin (Karaboga, 2010).

The general scheme of the ABC algorithm is as these:

In the initialization phase all the vectors of the population of food sources, x_m, are initialized (m = 1...SN, SN: population size) by scout bees and control parameters are set. Since each food source, x_m, is a solution vector to the optimization problem, each x_m vector holds n variables (x_{mi} , $i = 1$ ..... n), which are to be optimized. Equation 3.4 is used for initialization phase.

$$x_m = l_i + rand(0,1) * (u_i - l_i) \tag{3.4}$$

Where x_m is the food source, u_i and l_i are the upper and lower level of the solution space. u_i, l_i and rand (0, 1) are a random number in range [0, 1].

In employee bees phase, the bees search for food sources in the neighborhood. This exploration is defined in Equation 3.5.

$$v_{mi} = x_{mi} + \varphi_{mi}(x_{mi} - x_{ki}) \tag{3.5}$$

Where i is a randomly selected parameter index, x_k is a randomly selected food source, and φ_{mi} is a random number in the range [-1, 1]. After v_{mi} is generated we can obtain the fitness value for the food origin according to Equation 3.6.

$$fit_i = \begin{cases} \dfrac{1}{f_i + 1}, & f_i \geq 0 \\ 1 + |f_i|, & f_i < 0 \end{cases} \tag{3.6}$$

Where f_i shows the objective value of *ith* solution.

In onlooker bees' phase, after employee bees have found the food source they will share the information about the food source and its quality with the onlooker bees, the probability of selecting that food source by onlooker bees is represented in Equation 3.7.

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_i} \tag{3.7}$$

Where fit_i indicates the fitness solution represented by food source i and SN indicates the total number of food sources. Finally of the effectiveness of food sources cannot be improved, then the scout bees remove the existing solution and start searching for a new solution randomly using equation 3.4.

3.6 Proposed Method

Comparing the performance of KNN, SVM, RF and optimize the parameters of these algorithms using PSO and ABC algorithms to discover the most suitable algorithm for network ID are the objective of this thesis. The parameters used for determine the best optimized algorithms are classification accuracy, detection rate, time consumption and resource consumption. General plan of the thesis is given in Figure 3.6 below.

In the thesis project, Python is used for the experimental studies. Python is a dominant, widely used, general purpose and dynamic high-level programming language developed by Guido van Rossum (python, 2017). Python's syntax is easy to understand and also allows the programmer to express their concept in shorter script lines than in Java or C++. It has several different modules and libraries that make the data analysis task easier. The critical factor for the Python, is selecting the machine that has enough resources to run the algorithms. The property of the machine used in the thesis is given in the Table 3.7.

Table 3.7 Property of the machine used in the thesis

CPU	Intel(R) CPU 1.80 GHz
Cores	i7-4500U
OS	Ubuntu 16.04
Disk	250 GB
RAM	8 GB

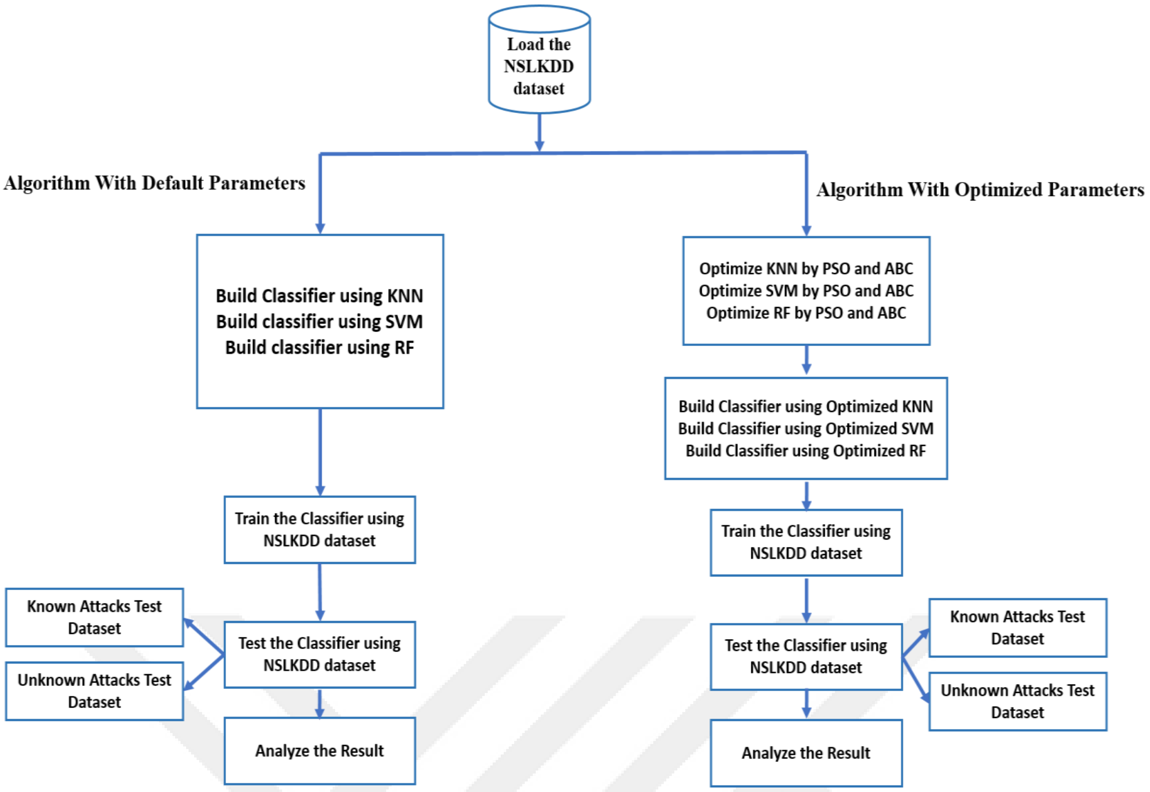

Load the NSLKDD dataset
Algorithm With Default Parameters
Algorithm With Optimized Parameters
Build Classifier using KNN
Build classifier using SVM
Build classifier using RF
Optimize KNN by PSO and ABC
Optimize SVM by PSO and ABC
Optimize RF by PSO and ABC
Build Classifier using Optimized KNN
Build Classifier using Optimized SVM
Build Classifier using Optimized RF
Train the Classifier using NSLKDD dataset
Train the Classifier using NSLKDD dataset
Known Attacks Test Dataset
Unknown Attacks Test Dataset
Test the Classifier using NSLKDD dataset
Test the Classifier using NSLKDD dataset
Known Attacks Test Dataset
Unknown Attacks Test Dataset
Analyze the Result
Analyze the Result

Figure 3.6 General Plan for thesis

In the thesis also, Ubuntu 16.04 with Python pre-installed, is used. To make sure that it is up-to-date, the Ubuntu System by “*apt-get update*” command can be used. For installing Python modules and libraries, “*pip tools*” are used. “*pip*” installs and manages the programming packages. All required packages for the project can be installed by “*pip install package-name*” command.

3.6.1 Dataset Preparation

In the thesis, NSLKDD dataset is used for intrusion detection. The dataset is noisy and contains unnecessary and correlated features, categorical records need to be converted to discrete or continuous values. Because of these, some critical processes need to be done before the dataset is used by the classifiers that run appropriately without facing errors. In the thesis, some preprocessing steps were applied to the dataset.

For the thesis study, dataset was uploaded into Python program using Pandas Python data analysis library that is an easy tool to manage and read datasets into Python. After the reading and labeling data columns according to features, the attacks were mapped into their categories. For example, in the dataset, *Back* or *Neptune* attacks that belongs to *DoS* class or

Nmap and *Satan* belongs to *Probe* class. Attacks and their categories are given in the dataset description in Table 3.1 and Table 3.2.

Dataset have different types of features like integer, zero passing or character values. But, machine learning algorithms commonly works with numeric data values. So it is essential to convert the character or categorical values into numeric values. As seen in Table 3.3 some features like protocol type, service and flag have character values that are going to be changed into integer values. After the mapping, these symbolic features were converted into numeric as given in Table 3.8.

Table 3.8 Converting the character values into numeric in NSLKDD

Feature	**Attribute with their numeric values**
Protocol Type	tcp=1, udp=2, icmp=3
Service	private=1, ftp_data=2, eco_i=3, telnet=4, http=5, smtp=6, ftp=7, ldap=8, pop_3=9, discard=10, ecr_i=11, imap4=12, domain_u=13, mtp=14, systat=15, iso_tsap=16, other=17, csnet_ns=18, finger=19, uucp=20, whois =21, netbios_ns=22, link=23, Z39_50=24, sunrpc=26, auth=27, domain=28, name=29, pop_2=30, urp_i=31, login=32, gopher=33, exec=34, time=35, remote_job=36, ssh=37, kshell=38, sql_net=39, shell=40, echo=41, pm_dump=42, IRC=43, netstat=44, ctf=45, netbios_ssn=46, tim_i=47, supdup=48, bgp=49, nnsp=50, rje=51, nntp=52, printer=53, efs=54, X11=55, ntp_u=56, tftp_u=57, red_i=58, urh_i=59, aol=60, harvest=61
Flag	REJ=1, SF=2, RSTO=3, S0=4, RSTR=5, SH=6, S3=7, S2=8, S1=9, OTH=10

The next step in dataset preparation is to normalize the dataset features. The raw dataset is composed of feature with different scales. So, to boost the performance of the algorithm it is recommended to rescale the data. In the thesis, the dataset features were normalized between range 0 and 1.

The final step in dataset preparation is feature reduction. As mentioned before the dataset contains some correlated, unnecessary features that need to be eliminated. Redundant and irrelevant attributes increase the computational cost while decreasing the performance of the classifier. To build an efficient and low-cost intrusive classifier model, PCA algorithm was applied on both train and test dataset to reduce the features and keep the essential elements. It was implemented from Sklearn that is a free Python library. The features of the dataset were reduced from 41 features to 26 features with PCA. But in the thesis, our experiments were made with 22, 23, 24, 25, 26, 27, 28 and 29 features, but 26 features provide the best performance as seen in Chapter 4.

3.6.2 Intrusion Detection using ML Algorithms with Default Parameters

There are two approaches in network anomaly detection as binary and multiclass classification. In binary classification, algorithms are trained with a binary dataset and make a binary classifier model. Binary classifier means that the model classifies the network connection into 2 groups as "*normal"* and "*attack",* respectively. This type of classification has lower computational cost and easy to implement.

In big enterprises, the networks admins need to know which type of attacks are happening in their distributed systems because each intruder has its methods of attacks. Thus, multi-class classification methods help them to identify a specific kind of attack and take action against them. In multi-class classification, the outcome of the prediction model is more than two classes. In this thesis a multi-classification that has five types of attacks (Normal, DoS, Probe, R2 and U2R) has been implemented. This is due to that it would cost a lot of resources and times to implement all kinds of attacks. For classifier model building, train dataset was used for training and for evaluating the performance of the model, two different test sets as known and unknown attacks test datasets were used.

3.6.2.1 Intrusion Detection using KNN with Default Parameters

In this thesis, the model that is made on KNN supposed to classify the dataset into five categories of Normal, DoS, Probe, R2L and U2R based on the training examples that was given to the algorithm. For the experiments, "K" parameter value was selected equal to 5. It means that KNN algorithm will take an instance and calculated its Euclidean distance from its five nearest neighbors based on majority vote of the neighbors the algorithms will decide to which category the example must be classified. Once the model is made, it was tested according to its predictability power on the test dataset. For classifier model validation we use 5-fold cross-validation. For validation, there are 125, 973 records so each time these records will be divided into five parts and each time 25,195 data records are used for testing, and 100,778 data records will be used for training. K parameter default value is 5 in Python. We used KNN algorithm with its default value for this experiment. For the evaluation, the time consumption, resource consumption, detection rate and accuracy rate on both known and unknown attack dataset have been analyzed.

3.6.2.2 Intrusion Detection using SVM with Default Parameters

After the detection used KNN, SVM was used for intrusion detection. Like K parameter in KNN, some parameters of SVM are as crucial for classification. SVM with RBF (Radius Bases Function) kernel was used for this classification task. When SVM is trained with RBF kernel two parameters must be taken into consideration, the "Cost" and "Gamma" parameters. Gamma determines the influence of training examples on the model that will be created. A low value shows that each training example does not have a high effect on the model and a higher value indicates that every training example has a high impact on the classifier model that is being created. Moreover, Cost parameter determines the cost of misclassification on the training examples. Cost with a high value make a rigorous classification, and the margin of error will be small, in this case, the classifier supposed to classify every training example correctly. Also Cost with lower values makes the margin of error a little loose; it might cause more misclassification. The optimal Cost value is a value that leaves some space for errors while it is intended to classify correctly. For the thesis, the default values for Cost and Gamma are given in Table 3.10. In fact SVM is a binary classifier, which tries to find a margin between positive and negative examples. Here the SVM handled the multi-class classification into one vs. one scheme. First, it takes two classes and classifies them into their categories, then it takes another two classes and classifies them. This process continues until the classes are finished. For comparison of the classifier accuracy, same datasets were used with 5-fold cross-validation and also the time consumption, resource consumption, detection rate and accuracy rate on both known and unknown attack dataset have been analyzed for SVM classifier.

Table 3.10 SVM parameters default values

Parameter	**Default value**
Cost	1
Gamma	1/number of features

3.6.2.3 Intrusion Detection using RF with Default Parameters

The third classifier used in the thesis is RF classifier. When RF wants to classify a dataset, it builds several decision trees, then combines them to get better accuracy and a stable prediction. Same as KNN and SVM, RF also has some important parameters. These parameters are either trying to increase the productivity of the algorithm or make the training process more manageable. Most essential parameters of RF that help the algorithm predict better are "n-estimator", "min-sample-leaf" and "random-state". One of the metrics that enhances RF prediction power is min-sample-leaf. Leaf is the end node of a decision tree. A smaller leaf makes the model more prone to capturing noise in train data. N-estimator is the number of trees that is built before taking the maximum voting or averages of predictions. A higher number of trees gives better performance but makes the algorithm slower. It must be chosen as high value as computer processor can handle because this makes the predictions stronger and more stable. The last parameter for the RF is random-state. The random-state parameter makes a solution easy to replicate. A definite value of random-state will always produce same results if given with same settings and training data. For comparison of the classifier accuracy, same datasets were used with 5-fold cross-validation and also the time consumption, resource consumption, detection rate and accuracy rate on both known and unknown attack dataset have been analyzed for RF classifier. For the thesis, the default values for the RF parameters are given in Table 3.11.

Table 3.11 RF parameters default values

Parameter	Default value
n-estimator	10
Random-state	0
Min-sample-leaf	1

3.6.3 Intrusion Detection using Optimized ML Algorithms

Feature selection eliminates unnecessary features from the dataset that are redundant or doesn't affect classifier performance. These unnecessary features confuse the algorithms and increase the computational cost.

Beside the dataset feature selection, parameters optimization has a significant effect on the ML algorithms and optimizing the machine learning algorithms parameters helps the classifier produce better results. Although ML algorithms are robust, still these algorithms need some optimization and modification to provide good classification accuracy and overall a good performance. The KNN, SVM and RF classifiers have several parameters, and the return of classification depends on the selection of the parameters and choosing appropriate values for the parameters concerning the given dataset. The parameter optimization made in the thesis is a continuous problem, so two metaheuristic algorithms, PSO and ABC swarm intelligent schemes that deals with continuous problems were used to optimize parameters of KNN, SVM, and RF classifiers. Contexts show that these two schemes are efficient for complex and complicated optimization models.

3.6.3.1 Parameter Optimization with PSO

In general, KNN classifier has several parameters such as:

- *K-neighbors*: number of neighbors for queries,
- *Weights*: weight function of each neighborhood,
- *algorithm*: used to compute the nearest neighbors,
- *p*: power parameter for the Murkowski metric,
- *metric*: used to measure the distance between instants

Among them the most critical parameter is K neighbors, so it was optimized in the thesis. First, K=5 was used as a fixed value for the KNN experiment. Here, PSO was used for finding the best value for the K.

SVM classifier has several parameters such as:

- *C:* Penalty parameter of the error term,
- *Kernel:* specifies the kernel type to be used in the algorithm
- *Degree:* degree of the polynomial kernel function
- *Gamma:* kernel coefficient
- *tolerance:* for stopping criterion
- *weight:* weight of each classes
- *max.iter:* maximum number of iteration (used as 30 in the thesis)
- *decision_function_shape:* for one-vs-rest or one-vs-one in multi - class classification

In the thesis, SVM with RBF kernel was used. Among all features given above, the most important parameters are *C* and *gamma,* according to the literatures. First, it was used that *C* as 1, *gamma* as (1/n_features) before. Also one reason is they always exist in any kernel types, these two parameters were optimized.

Like SVM, also RF has several parameters such as:

- *n_estimators*: the number of trees in the forest
- *max_features*: number of features an individual tree tries in random forest
- *n_jobs*: how many processors the engine can use
- *random_state*: makes a solution replicate easier
- *oob_score*: random forest cross validation method
- *min_sample_leaf*: minimum sample leaf size

Among all these features as mentioned the most important parameters as n_estimators and,min_sample_leaf and random_state were optimized in RF algorithm.

Figure 3.7 shows a general scheme of PSO algorithm for parameter optimization below.

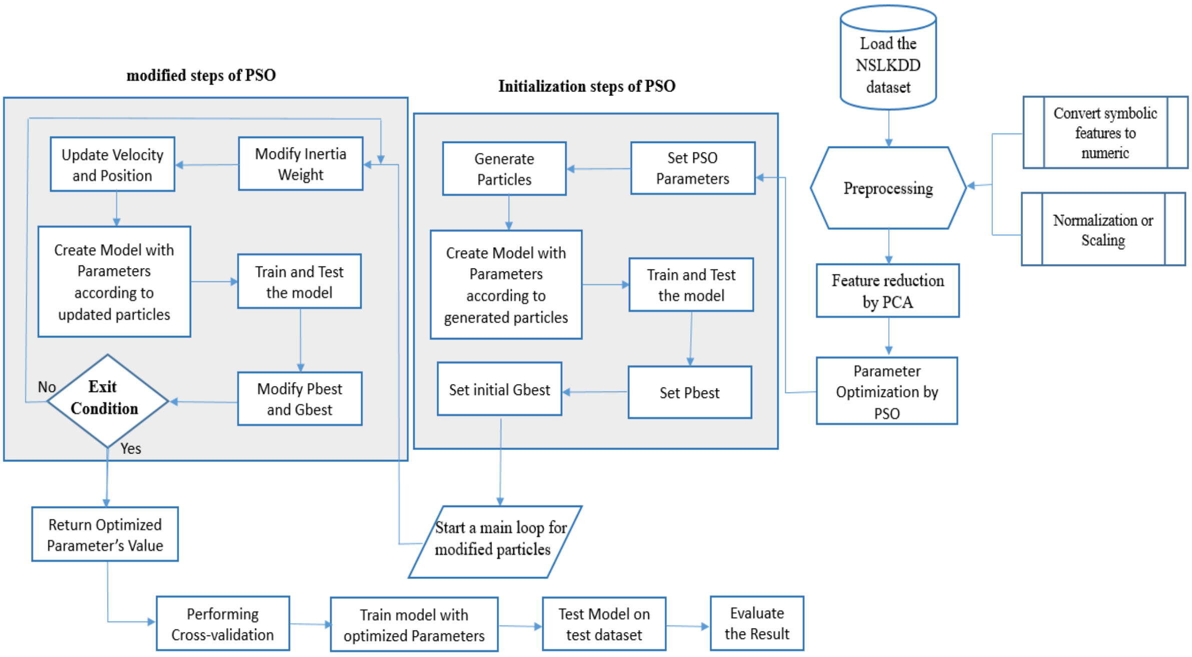

modified steps of PSO
Initialization steps of PSO
Load the NSLKDD dataset
Convert symbolic features to numeric
Normalization or Scaling
Preprocessing
Feature reduction by PCA
Parameter Optimization by PSO
Generate Particles
Set PSO Parameters
Create Model with Parameters according to generated particles
Train and Test the model
Set initial Gbest
Set Pbest
Update Velocity and Position
Modify Inertia Weight
Create Model with Parameters according to updated particles
Train and Test the model
No
Exit Condition
Yes
Modify Pbest and Gbest
Return Optimized Parameter's Value
Start a main loop for modified particles
Performing Cross-validation
Train model with optimized Parameters
Test Model on test dataset
Evaluate the Result

Figure 3.7 Semantic figure of parameter optimization by PSO

As seen in the Figure 3.7, data preprocessing and feature reduction are same with the system used the KNN, SVM and RF with default parameters. After these steps, optimization for the ML with PSO is started in two different phase as initialization and modification.

In initialization step, the PSO algorithm parameters were set. Number of swarm was selected as 10 and 20 particles, maximum iteration which also determined as termination criteria was 30, C_1 and C_2 were set as 0.7 and 1, respectively. Inertia weights were set as 0.9 and 0.4 for maximum and minimum, respectively. Maximum speed was different for each parameter and it was set 25% of each parameters ranges. PSO particles were generated randomly in swarm search area and theirs initial velocities are 0. In all of the classifiers, the accuracy rate is the objective function. So that PSO run for each of the classifier, separately and gave the best parameters values for the each of the classifier. Then, PSO-optimized classifiers have been trained and tested on datasets.

In modification step, inertia weights, velocities and positions were calculated again and PSO was found the best solution for the classifier. First of all, inertia weight was updated as Eq.3.8 given below.

$$w = w_{max} - \frac{iteration}{max - iteration}\ (w_{max} - w_{min}) \tag{3.8}$$

Then, all particles velocities and positions were updated according to the ranges of optimization parameters. The range of k parameters is (1-100) for KNN, for Cost and Gamma are $[2^{-1},2^{3}]$ and $[2^{-6},2]$ for SVM, respectively. In RF classifier, n_estimators was optimized in range (1-100) where default value is 10 in Python. As it mentioned before it was used as a default value for the experiments when classifying the attacks using RF algorithm with default values. Min_sample_leaf was optimized in range (1-10) where default value for this parameter is 1 and finally random_state optimization range was (0-100) as its default value is 0. The same rule is applied for velocity limitation. According to the best parameter values, optimized classifiers have been trained and tested on datasets. If the current particles position fitness value (*pbest*) is better than previous one, current position was set as *pbest* for the particle. Like it, if current *gbes*t is better than all gbests obtained by swarm in each of the iteration, it was set as *global gbest*. When the 30 iteration is done the algorithm return a set of values for the parameters in form *global gbest*. These parameter values help the classifier works well (Table 3.12).

After PSO finds the optimal values for the parameters, the classifiers use those parameter's values to build a model to classify normal network traffic from attack traffic. Here the attacks are classified into 5 categories of Normal, DoS, Probe, R2L and U2R categories. The classifiers (KNN, SVM and RF) with optimized parameters use NSL-KDD train dataset (KDDTrain) for training the model and the NSL-KDD test dataset. (KDDTrain+_20% for known attack testing and KDDTest+ for unknown attack testing) to test the model. Here 5-fold cross-validation method is used to validate the classifiers models. 80% of the dataset is used for training and 20% of the validation is a measure of fit. The goal of validation is to estimate the expected level of fit of a model to a data set that is independent of the data that were used to train the model.

The performance evaluation metrics for the algorithms are detection rate, accuracy rate, time usage and resource usage.

For the classifiers, the optimized parameters found by PSO are given in Table 3.13 below:

Table 3.12 Parameters Optimization Ranges

Classifier	Optimization Ranges for the parameters
KNN	(1-100) for k
SVM	$[2^{-1},2^{3}]$ for Cost and $[2^{-6},2]$for Gamma
RF	(1-100) for n_estimators (1-10) for min_sample_leaf (0-100) for random-state

Table 3.13 Best parameter's values found by PSO

PSO with 10 particles					
KNN	SVM		RF		
K	Cost	Gamma	N_estimators	Min_sample_leaf	Random_state
3	6.5	1.8	73	1	42
PSO with 20 particles					
KNN	SVM		RF		
K	Cost	Gamma	N_estimators	Min_sample_leaf	Random_state
3	6.5	1.85	70	1	38

In "Experimental Result" analysis section, it can be seen how these optimized parameters values enhance the algorithms accuracy and detection rates.

3.6.3.2 Parameter Optimization with ABC

Like PSO, also ABC was used to optimize the same parameters of classifiers. Figure 3.8 demonstrates the ABC steps for optimizing the KNN, SVM and RF parameters (Table 3.14).

As it is seen in the Figure 3.8, data preprocessing and feature reduction are same with the system used the KNN, SVM and RF with default parameters. After these steps, optimization for the ML with ABC is started in two different phase as initialization and modification.

In initialization step, the ABC algorithm parameters were set. Number of swarm was selected as 10 and 20 bees and maximum iteration which also determined as termination criteria was 30. The employed bee percentage was 50% (as used in the literature (Bansal, Sharma, & JAdon, 2013). In all of the classifiers, the accuracy rate is the objective function. So that ABC run for each of the classifier, separately and gave the best parameters values for the each of the classifier. Then, ABC-optimized classifiers have been trained and tested on datasets like PSO-optimized. Unlike PSO, there are three different steps for the bees.

1) Employed bee step: In this phase, employed bees modify the current solution based on the information of individual experiences and the fitness value of the new solution. If the

fitness value of the new food source is higher than that of the old food source (solution), the bee updated her position with new one and discards the old one.

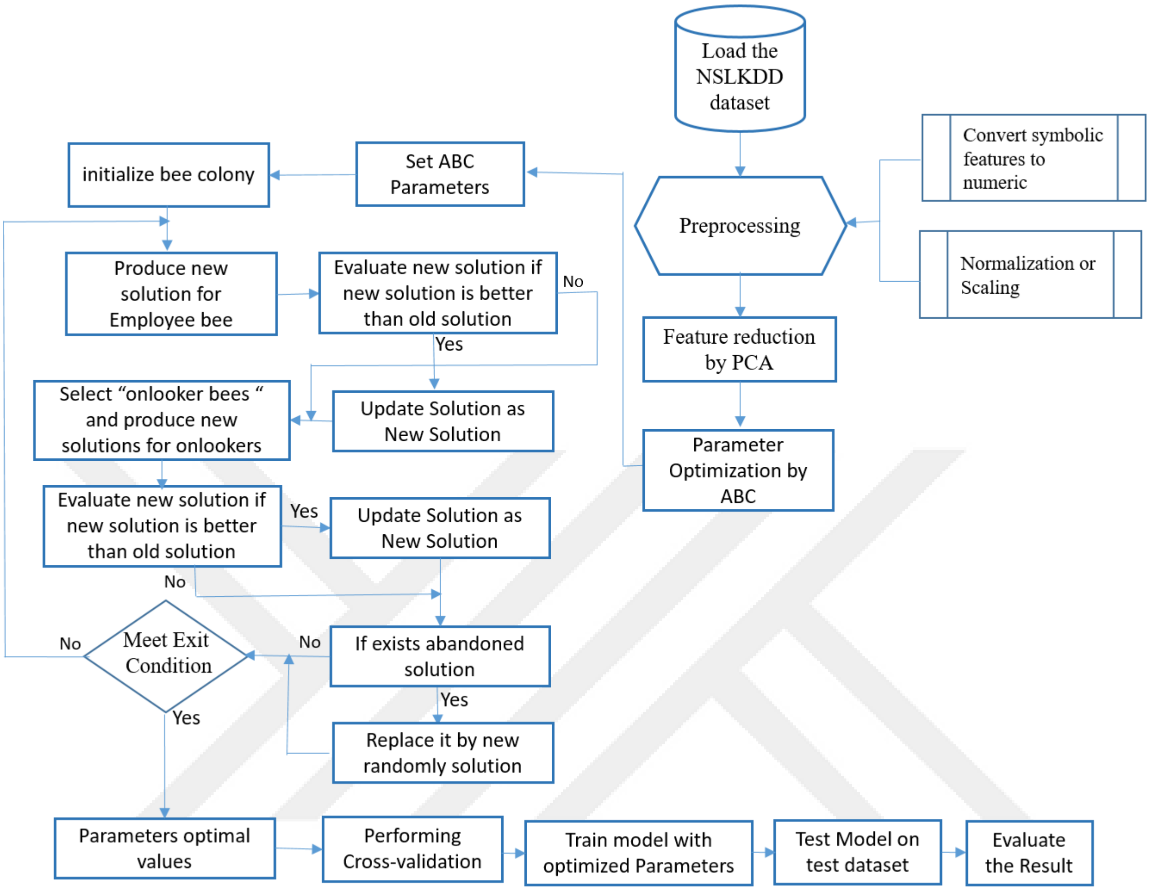

Load the NSLKDD dataset
Convert symbolic features to numeric
Preprocessing
Normalization or Scaling
Feature reduction by PCA
Parameter Optimization by ABC
Set ABC Parameters
initialize bee colony
Produce new solution for Employee bee
Evaluate new solution if new solution is better than old solution
No
Yes
Update Solution as New Solution
Select "onlooker bees " and produce new solutions for onlookers
Evaluate new solution if new solution is better than old solution
Yes
Update Solution as New Solution
No
Meet Exit Condition
No
No
If exists abandoned solution
Yes
Replace it by new randomly solution
Yes
Parameters optimal values
Performing Cross-validation
Train model with optimized Parameters
Test Model on test dataset
Evaluate the Result

Figure 3.8 Semantic figure of parameter optimization by ABC

2) Onlooker bee step: In this phase, all the employed bees share the fitness information of the updated best solutions and their position information with the onlooker bees in the hive. The onlooker bees analyses the available information and select a solution using Equation 3.7. As like employee bee, onlooker bee produces a modification in the position in its memory and checks the fitness of the candidate solution. If the fitness is better than the previous one, the bee memorizes the new position and forgets the old one; if it is not better it uses the old one. This new solution contains optimal values for the parameters that are optimizing. The classifiers (KNN, SVM and RF) use the new parameter values and get trained and tested on NSLKDD dataset.

3) Scout bee step: If the position of solution is not updated for a predetermined number of cycles, then the food source is assumed to be abandoned and scout bees phase starts. The predetermined number of cycles is calculated based on colony size and number of optimized

parameters. The bees that are associating with the abundant food source is scout bees and the food source (solution) replaced by the randomly chosen food origin within the parameters optimization ranges. Then the new solution is acting as optimized parameters values. The classifiers use these parameter values when they are get trained and tested on NSLKDD dataset.

The above processes repeated till termination criteria where the maximum iteration cycle is met. After the algorithm reached the maximum cycle then it pass the best value found for the parameters to the algorithms. The next step after cross validation is to train the algorithms by train dataset using ABC generated values for optimized parameters and default values for fixed parameters. The performance evaluation metrics for the algorithms are detection rate, accuracy rate, time usage and resource usage. Table 3.15 shows the algorithms parameters values found by ABC algorithm.

Table 3.14 Parameters Optimization Range in ABC

Classifier	Range for the parameters
KNN	(1-100) for k
SVM	$[2^{-1},2^{3}]$ for Cost and $[2^{-6},2]$ for Gamma
RF	(1-100) for n_estimators (1-10) for min_sample_leaf (0-100) for random-state

Table 3.15 Best parameter’s values found by ABC

ABC with 10 bees					
KNN	SVM		RF		
K	Cost	Gamma	N_estimator	Min_sample_leaf	Random_state
3	6.04	1.33	34	1	97
ABC with 20 bees					
KNN	SVM		RF		
K	Cost	Gamma	N_estimator	Min_sample_leaf	Random_state
3	5.34	1.50	33	1	97

ABC and PSO found different optimal value for the ML algorithms parameters because the function we have optimize has different local minima in each optimization algorithms.

4. EXPERIMENTAL RESULTS

The aim of this thesis is to design an INID using the ML algorithms such as KNN, SVM and RF. To achieve this goal, NSL-KDD original dataset was used for training and testing of the algorithms. Since dataset is noisy and contains redundant information, the dataset was preprocessed and then by applying PCA method. The features of the dataset were reduced from 41 features to 26 features with PCA. Our experiments were made with 22, 23, 24, 25, 26, 27, 28 and 29 features, but 26 features provide the best performance as below in Table 4.1. To improve the performance of KNN, SVM and RF, some critical parameters of these algorithms were optimized using ABC and PSO metaheuristic algorithms. All the of experiments of the thesis are multiclass classification. For testing the algorithms a known and an unknown attack datasets were used. Following shows the result of different experiments.

Table 4.1 Accuracy rate using different number of features by PCA

Classifiers	**Number of features**							
	22	**23**	**24**	**25**	**26**	**27**	**28**	**29**
KNN	0.777	0.777	0.778	0.776	0.78*	0.779	0.778	0.779
SVM	0.759	0.753	0.750	0.759	0.76*	0.75	0.755	0.752
RF	0.749	0.73	0.733	0.73	0.75*	0.749	0.733	0.749

*: Best result

4.1 Algorithms Results with Default Parameters

In the first experiment KNN, SVM and RF algorithms are used to build an anomaly detection model with default parameters. Table 4.2 illustrates the algorithms default parameter values.

Table 4.2 Default Parameters values

Classifiers	**Default Parameter Values**
KNN	K = 5

SVM	Cost = 1 (Float) and
	Gamma = 1 / number_of_features

RF	Min_sample_leaf= 1 (int)
	N_estimators= 10 (int)
	Random_state= 0 (int)

The model used the classifier with default parameter is trained on NSLKDD training dataset and tested on the NSLKDD testing datasets both known and unknown. The known test dataset results of the experiments are shown in Table 4.3. Accuracy Rate is the ratio of correctly classified instances and the total number of instances and as well as detection rate is defined as the ratio between the number of correctly detected attacks and the total number of attacks. Accuracy Rate given in the table with other performance criteria is shown the overall system accuracy. Besides, Accuracy Rates for every intrusion type are given with confusion matrices with average accuracy.

Table 4.3 Intrusion detection results using ML with default Parameters (Known Attacks)

Classifiers	CV-Score	Train Time (Sec)	Test Time (Sec)	Memory Usage (RAM)	Resource Usage (CPU)	Overall Detection Rate	Overall Accuracy Rate
KNN	0.9978	0.44	2.25	0.25 GB	42.7	0.997	0.998
SVM	0.990	79.5	10.23	0.27 GB	26	0.992	0.990
RF	0.995	6.33	0.038	0.25 GB	31.10	0.994	0.996

Table 4.5 shows the confusion matrices for the algorithms for known testing dataset.

As it is mentioned before the performance of the algorithms were evaluated based on computational cost, classification accuracy and detection rate. If computational cost was considered, all the algorithms perform very well in term of memory usage and CPU usage, but RF consumes fewer memory and time than SVM but CPU usage of RF is a bit higher than SVM algorithm. In terms of accuracy and detection rate, KNN performs better than other algorithms. It is not very clear according to the Table 4.3; but the confusion matrices are checked, it can be seen that KNN classifies network data better than SVM and RF. For example, how accurate these three algorithms categorized the normal data, it can be seen that among 13449 normal data records KNN identify 13434 normal data record accurately while SVM identifies 13336 and RF detect 13428. So, it can be said that in this experiment KNN performs better than SVM and RF.

Next step is to examine the performance of the algorithms on test dataset with the unknown attack. This test dataset contained some attacks that the classifier model has not seen before, based on the pattern it learned from the training, the model will identify those attacks. The unknown test dataset results of the experiments are shown in Tables 4.4.

Table 4.4 Intrusion detection results using ML with default Parameters (Unknown Attacks)

Classifiers	CV-Score	Train Time (Sec)	Test Time (Sec)	Memory Usage (RAM)	Resource Usage (CPU)	Overall Detection Rate	Overall Accuracy Rate
KNN	0.997	0.36	3.34	0.25 GB	36.7	0.72	0.78
SVM	0.991	81.27	9.6	0.26 GB	26.4	0.66	0.76
RF	0.996	5.70	0.030	0.24 GB	30.20	0.66	0.75

Here is in Table 4.4, it can be seen that KNN perform better than SVM and RF regarding classification accuracy, detection rate and train time. While RF consumes less time than KNN for testing, SVM regarding resource consumption does not perform very well but it accuracy and detection rate are better than RF.

Table 4.5 Confusion Matrices for ML algorithms on known test dataset

	Confusion Matrix for KNN						Confusion Matrix for SVM						Confusion Matrix for RF				
	Actual						Actual						Actual				
	Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R
Predicted	13434	7	8	6	1		13336	50	20	16	3		13428	14	16	14	3
	6	2275	1	0	1		70	2227	8	1	0		15	2273	5	0	1
	5	4	9225	3	0		27	10	9205	6	1		2	2	9213	2	0
	4	3	0	199	3		16	2	1	186	3		4	0	0	192	3
	0	0	0	1	6		0	0	0	0	4		0	0	0	1	4
Accuracy	99.91	99.38	99.90	95.21	54.54		99.15	97.29	99.68	88.99	36.36		99.84	99.3	99.7	91.86	36.36
Average Accuracy	89.78						84.29						85.42				

Table 4.5 demonstrates the confusion matrices of the algorithms on known test dataset. From the confusion matrices, we can figure out that the KNN algorithm performs better than RF and SVM in network anomaly detection. Most algorithms struggling with detecting R2L and U2R attacks as they are minority attacks and there is a few samples existed in the train dataset. Lower detection rates for R2L and U2R are due to fewer data samples of these attacks in the training dataset. Since DoS and Probe attacks happen quite often, system is successful in identifying these two attacks and it is worth to implement. If it is looked at the previous studies, in all those contexts these minority attacks detection is not very successful too due the same reason mentioned here. In (Voldan, 2016) from R2L was detected 0 and U2R was detected 57% using SVM. In (Chih and Chai, 2010) accuracy rate for detecting R2L and U2R attacks are 85% and 40% respectively using KNN algorithm and 78%, 60% accuracy rate using SVM algorithm. The dataset for their experiments were known attack test dataset.

If the confusion matrices of the algorithms on unknown dataset in Table 4.6 are checked, it is noticed that the algorithms perform very well in detection Normal, Probe and DoS attacks. For example from 9711 normal data record, the KNN algorithm detects 9446, SVM identifies 9419 and RF detects 9432 normal data records correctly. It is also same for Probe and DoS. But for minority attacks, the case is different. For example from 2754 R2L attacks in the dataset; KNN identifies only 490, SVM detects 198 and RF classifies 224 attacks.

Same as detecting unknown network attacks, KNN performs better in identifying unknown attacks. It can see in the Table 4.4; detection rate of KNN is 72% while SVM is 66% and RF is 66. As well as the accuracy rate for KNN is 78%, while SVM accuracy rate is 76% and RF accuracy is 75%. In terms of computational cost also KNN consume the least amount of resource and time.

Table 4.6 Confusion Matrices for ML algorithms on unknown test dataset

	Confusion Matrix for KNN						Confusion Matrix for SVM						Confusion Matrix for RF				
	Actual						Actual						Actual				
	Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R
Predicted	9446	318	1451	1720	28		9419	749	1149	2267	97		9432	782	1621	2027	101
	204	1783	145	301	133		209	1352	123	111	91		221	1453	94	243	65
	52	258	5862	221	2		78	320	6186	178	5		58	186	5743	260	7
	8	62	0	490	16		5	0	0	198	5		0	0	0	224	27
	1	0	0	22	21		0	0	0	0	2		0	0	0	0	0
Accuracy	97.27	73.64	78.6	17.79	10.5		96.99	55.84	82.94	7.18	1		97.12	60.01	77	8.13	0
Average Accuracy	55.56						48.79						48.45				

4.2 Algorithms Results with Optimized Parameters

The performance of the machine learning algorithms strongly depends on the selection of the appropriate parameters. Based on this theorem, two metaheuristic algorithms that are PSO and ABC were used to select the best values for the parameters of the algorithms for intrusion detection system. For this aim; K parameter for KNN, Cost and Gamma parameters for SVM and N_estimators, Min_sample_leaf and Random_state for RF algorithm were optimized in the thesis. After the PSO and ABC algorithms find the most convenient values for the parameters, the accuracies of the algorithms were tested based on both known and unknown attack test datasets.

Table 4.7 illustrates optimized parameter's values found by PSO. PSO swarm is set as 10 and 20 particles.

Table 4.7 Best parameter's values found by PSO

PSO with 10 particles					
KNN	SVM		RF		
K	Cost	Gamma	N_estimators	Min_sample_leaf	Random_state
3	6.5	1.8	73	1	42
PSO with 20 particles					
KNN	SVM		RF		
K	Cost	Gamma	N_estimators	Min_sample_leaf	Random_state
3	6.5	1.85	70	1	38

In the thesis, PSO was used with 10 particles and 20 particles. They are written as "PSO-10" and "PSO-20" respectively. The known test dataset results of the classifiers optimized by "PSO-10" are shown in Table 4.8.

Table 4.8 Intrusion detection results using ML with PSO-10 Optimized Parameters (Known Attacks)

Algorithms	CV-Score	Train Time (Sec)	Test Time (Sec)	Memory Usage (RAM)	Resource Usage (CPU)	Overall Detection Rate	Overall Accuracy Rate
KNN+PSO-10	0.998	0.83	0.46	0.21 GB	30.8	1.000	0.999
SVM+PSO-10	0.994	51.7	5.65	0.30 GB	29.2	0.995	0.994
RF + PSO-10	0.998	49.5	0.24	0.21 GB	27.4	0.999	0.999

Also, the unknown test dataset results of the classifiers optimized by PSO-10 are shown in Table 4.9.

Table 4.9 Intrusion detection results using ML with PSO-10 Optimized Parameters (Unknown Attacks)

Algorithms	CV-Score	Train Time (Sec)	Test Time (Sec)	Memory Usage (RAM)	Resource Usage (CPU)	Overall Detection Rate	Overall Accuracy Rate
KNN+PSO-10	0.998	0.66	2.25	0.21 GB	25.4	0.749	0.795
SVM+PSO-10	0.994	53.8	6.78	0.25 GB	25.2	0.683	0.770
RF + PSO-10	0.998	11.2	0.041	0.20 GB	53.2	0.696	0.76

The known and unknown test dataset results of the classifiers optimized by "PSO-20" are shown in Table 4.10 and 4.11, respectively

.

Table 4.10 Intrusion detection results using ML with PSO-20 Optimized Parameters (Known Attacks)

Algorithms	CV-Score	Train Time (Sec)	Test Time (Sec)	Memory Usage (RAM)	Resource Usage (CPU)	Overall Detection Rate	Overall Accuracy Rate
KNN+PSO-20	0.998	0.77	0.54	0.27 GB	21.1	1.000	0.999
SVM+PSO-20	0.993	54.4	5.65	0.30 GB	37.2	0.995	0.994
RF + PSO-20	0.998	57.7	0.25	0.11 GB	37.4	0.999	0.999

Table 4.11 Intrusion detection results using ML with PSO-20 Optimized Parameters (Unknown Attacks)

Algorithms	CV-Score	Train Time (Sec)	Test Time (Sec)	Memory Usage (RAM)	Resource Usage (CPU)	Overall Detection Rate	Overall Accuracy Rate
KNN+PSO-20	0.998	1.23	0.54	0.21 GB	51.1	0.749	0.795
SVM+PSO-20	0.994	51.9	4.33	0.29 GB	27.9	0.686	0.765
RF + PSO-20	0.998	52.2	0.038	0.20 GB	39.6	0.705	0.773

As seen in the Tables 4.8, 4.9, 4.10 and 4.11, the results of optimized algorithms shows that they perform better than the algorithms with default parameters values.

After PSO, ABC used to optimize the ML algorithms. Table 4.12 illustrates optimized parameter's values found by ABC.

Table 4.12 Best parameter's values found by ABC

<table>
<tr><th colspan="6">ABC with 10 bees</th></tr>
<tr><td>KNN</td><td colspan="2">SVM</td><td colspan="3">RF</td></tr>
<tr><td>K</td><td>Cost</td><td>Gamma</td><td>N_estimator</td><td>Min_sample_leaf</td><td>Random_state</td></tr>
<tr><td>3</td><td>6.04</td><td>1.33</td><td>34</td><td>1</td><td>97</td></tr>
<tr><td colspan="6">ABC with 20 bees</td></tr>
<tr><td>KNN</td><td colspan="2">SVM</td><td colspan="3">RF</td></tr>
<tr><td>K</td><td>Cost</td><td>Gamma</td><td>N_estimator</td><td>Min_sample_leaf</td><td>Random_state</td></tr>
<tr><td>3</td><td>5.34</td><td>1.50</td><td>33</td><td>1</td><td>97</td></tr>
</table>

In the thesis, like PSO, ABC was used with 10 and 20 bees for comparison. They are written as “ABC-10” and “ABC-20”, respectively. Tables 4.13-4.16 below show the result of our experiments on test dataset.

Table 4.13 Intrusion detection results using ML with ABC-10 Optimized Parameters (Known Attacks)

Algorithms	CV-Score	Train Time (Sec)	Test Time (Sec)	Memory Usage (RAM)	Resource Usage (CPU)	Overall Detection Rate	Overall Accuracy Rate
KNN+ABC10	0.998	0.67	0.57	0.20 GB	21.00	1.00	0.999
SVM+ABC-10	0.993	52.31	4.70	0.22 GB	22.00	0.995	0.994
RF+ ABC-10	0.998	34.12	0.34	0.21 GB	29.8	0.999	0.999

Table 4.14 Intrusion detection results using ML with ABC-10 Optimized Parameters (Unknown Attacks)

Algorithms	CV-Score	Train Time (Sec)	Test Time (Sec)	Memory Usage (RAM)	Resource Usage (CPU)	Overall Detection Rate	Overall Accuracy Rate
KNN+ABC-10	0.998	0.68	1.42	0.20 GB	22.5	0.751	0.795
SVM+ABC-10	0.994	65.86	5.71	0.23 GB	17.4	0.71	0.77
RF+ ABC-10	0.998	22.21	0.15	0.20 GB	24.20	0.709	0.779

Table 4.15 Intrusion detection results using ML with ABC-20 Optimized Parameters (Known Attacks)

Algorithms	CV-Score	Train Time (Sec)	Test Time (Sec)	Memory Usage (RAM)	Resource Usage (CPU)	Overall Detection Rate	Overall Accuracy Rate
KNN+ABC-20	0.998	0.69	0.60	0.21 GB	28.1	1.00	0.999
SVM+ABC-20	0.993	45.8	4.33	0.29 GB	27.9	0.994	0.995
RF + ABC-20	0.998	52.2	0.31	0.30 GB	42.8	0.999	0.999

Table 4.16 Intrusion detection results using ML with ABC-20 Optimized Parameters (Unknown Attacks)

Algorithms	CV-Score	Train Time (Sec)	Test Time (Sec)	Memory Usage (RAM)	Resource Usage (CPU)	Overall Detection Rate	Overall Accuracy Rate
KNN+ABC-20	0.998	0.62	2.87	0.20 GB	25.4	0.751	0.7955
SVM+ABC-20	0.994	110	16.2	0.30GB	13.2	0.684	0.764
RF + ABC-20	0.998	20.43	0.13	0.20 GB	25.9	0.709	0.778

In terms of detecting known network attacks, both optimized version of the algorithms perform very well, there is only a very little difference between ABC version of the algorithms and PSO version of the algorithm. Whether regarding detecting unknown network attacks the case is different KNN+PSO-10 detection rate is 74.9% and accuracy rate is 79.4%. KNN+PSO-20 detection rate is 74.9% and accuracy rate is 79.5%. Resource (RAM and CPU) utilization in PSO with 20 particles is higher than PSO-10. KNN+ABC-10 and KNN+PSO-20 have the similar detection rate and the same accuracy rate of 79.5%. Resource usage in KNN+ABC-20 is higher than KNN+ABC-10. When it comes to compare the performances of KNN based on PSO and ABC, the resource was used by KNN+ABC is lesser than KNN+PSO. In conclusion KNN+ABC-10 shows the better performance in detecting network attack data. In case of SVM, SVM+PSO-10 detection rate is 68% and accuracy rate is 77% while SVM+PSO-20 detection rate has been improved to 68.6% and accuracy rate dropped to 76.5%. SVM+ABC-10 detection rate raised to 71% and accuracy rate is 77% and SVM+ABC-20 detection rate dropped to 68.4% and accuracy rate decreased to 76.4%. In term of time and resource usage SVM+ABC-20 has the highest time and RAM utilization and least CPU utilization in SVM case. To conclude SVM+PSO-10 has the highest accuracy rate and SVM+ABC-10 shows the highest performance in SVM. In term of RF, RF+ABC-10 with detection rate of 70.9% and accuracy rate of 77.9% performs similar as RF+ABC-20 whose detection rate is same while the accuracy rate is 77.8%. RF+PSO-20 with detection rate of 70.5% and accuracy rate 77.3% performs better than RF+PSO-10 where its and detection rate is 69.6% and accuracy rate is 76%. Overall RF+ABC-10 performs better than other optimized version of RF algorithm. For known test dataset, while Tables 4.17 and 4.21 demonstrate the confusion matrices of the optimized ML algorithms by PSO and ABC with 10 particles, Tables 4.19 and 4.23 demonstrate with 20 particles. Besides, Tables 4.18, 4.20, 4.22 and 4.24 demonstrate the results by PSO and ABC with 10 and 20 particles on unknown test dataset. Confusion matrices tables give a clear picture of how the optimized algorithms classify the test instances. As seen that by optimizing the algorithm parameters the detection of minority attacks also have been improved. From this experiment, it can be figured out that selecting the appropriate values for algorithms parameters affect the algorithm performance positively.

We must note that the Accuracy Rate written in the tables are the Overall Accuracy Rate of the detection system which is calculated based on number of correctly classified data per total number of data in the dataset. On the other hand, the accuracy calculated in the confusion matrix tables are the individual accuracy of each class.

Table 4.17 Confusion Matrices for ML algorithms optimized by PSO-10 on known test dataset

	Confusion Matrix for KNN+PSO-10						Confusion Matrix for SVM+PSO-10						Confusion Matrix for RF+PSO-10				
	Actual						Actual						Actual				
	Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R
Predicted	13448	0	0	0	0		13387	32	9	15	2		13447	0	0	1	1
	0	2289	0	0	0		37	2255	1	1	1		1	2289	0	0	0
	0	0	9234	0	0		19	2	9223	3	0		0	0	9234	0	0
	0	0	0	209	0		6	0	1	189	1		0	0	0	208	0
	1	0	0	0	11		0	0	0	1	7		1	0	0	0	10
Accuracy	99.99	100	100	100	100		99.53	98.51	99.88	90.43	63.63		99.98	100	100	99.52	90.9
Average Accuracy	99.99						90.39						98.08				

Table 4.18 Confusion Matrices for ML algorithms optimized by PSO-10 on unknown test dataset

	Confusion Matrix for KNN+PSO-10					Confusion Matrix for SVM+PSO-10					Confusion Matrix for RF+PSO-10				
	Actual					Actual					Actual				
	Normal	Probe	DoS	R2L	U2R	Normal	Probe	DoS	R2L	U2R	Normal	Probe	DoS	R2L	U2R
Predicted	9465	367	1077	1730	42	9421	62	830	1344	23	9445	674	1297	1854	66
	183	1747	103	338	102	205	1960	126	716	140	208	1547	142	443	95
	58	256	6263	221	2	79	399	6408	189	8	56	200	6017	261	2
	5	50	0	417	13	6	0	94	498	4	2	0	2	104	12
	0	1	15	48	41	0	0	0	7	25	0	0	0	92	25
Accuracy	97.46	71.57	83.97	15.14	20.5	97.01	80.95	85.92	18.08	12.5	97.26	63.89	80.67	3.77	12.5
Average Accuracy	57.72					58.89					51.61				

Table 4.19 Confusion Matrices for ML algorithms optimized by PSO-20 on known test dataset

	Confusion Matrix for KNN+PSO-20						Confusion Matrix for SVM+PSO-20						Confusion Matrix for RF+PSO-20				
	Actual						Actual						Actual				
	Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R
Predicted	13448	0	0	0	0		13387	32	9	15	2		13447	0	0	1	1
	0	2289	0	0	0		37	2255	1	1	1		1	2289	0	0	0
	0	0	9234	0	0		19	2	9223	3	0		0	0	9234	0	0
	0	0	0	209	0		6	0	1	189	1		0	0	0	208	0
	1	0	0	0	11		0	0	0	1	7		1	0	0	0	10
Accuracy	99.99	100	100	100	100		99.53	98.51	99.88	90.43	63.63		99.98	100	100	99.52	90.9
Average Accuracy	99.99						90.39						98.08				

Table 4.20 Confusion Matrices for ML algorithms optimized by PSO-20 on unknown test dataset

	Confusion Matrix for KNN+PSO-20						Confusion Matrix for SVM+PSO-20						Confusion Matrix for RF+PSO-20				
	Actual						Actual						Actual				
	Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R
Predicted	9464	367	1077	1730	42		9420	696	1015	2236	70		9460	657	1116	1959	44
	183	1747	103	338	102		205	1380	229	143	96		197	1556	194	297	118
	59	256	6263	221	2		79	345	6214	126	9		51	208	6146	214	2
	5	50	0	417	13		6	0	0	238	10		3	0	2	240	12
	0	1	15	48	41		1	0	0	11	15		0	0	0	44	24
Accuracy	97.45	72.16	83.97	15.14	20.5		97	56.53	83.31	8.64	7.5		97.41	63.74	82.4	8.7	12
Average Accuracy	57.84						50.6						52.85				

Table 4.21 Confusion Matrices for ML algorithms optimized by ABC-10 on known test dataset

	Confusion Matrix for KNN+ ABC-10						Confusion Matrix for SVM+ ABC-10						Confusion Matrix for RF+ ABC-10				
	Actual						Actual						Actual				
	Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R
Predicted	13448	0	0	0	0		13388	32	7	16	1		13447	0	0	1	1
	0	2289	0	0	0		36	2254	1	0	1		0	2289	0	0	0
	0	0	9234	0	0		19	3	9225	4	0		0	0	9234	0	0
	0	0	0	209	0		6	0	1	188	2		1	0	0	208	0
	1	0	0	0	11		0	0	0	1	7		1	0	0	0	10
Accuracy	99.99	100	100	100	100		99.54	98.47	99.90	89.95	63.63		99.98	100	100	99.52	90.9
Average Accuracy	99.99						90.29						98.08				

Table 4.22 Confusion Matrices for ML algorithms optimized by ABC-10 on unknown test dataset

	Confusion Matrix for KNN+ ABC-10						Confusion Matrix for SVM+ ABC-10						Confusion Matrix for RF+ ABC-10				
	Actual						Actual						Actual				
	Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R
Predicted	9465	367	1057	1727	40		9451	445	1229	1973	56		9455	481	1223	1963	56
	183	1747	119	321	107		207	1767	135	266	108		203	1744	137	261	108
	58	254	6265	177	2		51	209	6092	250	2		50	196	6096	255	2
	5	52	2	419	13		2	0	237	237	20		3	0	2	246	10
	0	1	15	110	38		0	0	28	28	24		0	0	0	29	24
Accuracy	97.46	72.16	84	15.21	19		97.32	54.97	83.36	9.76	7		97.36	72.03	81.73	8.93	12
Average Accuracy	57.56						50.48						54.41				

Table 4.23 Confusion Matrices for ML algorithms optimized by ABC-20 on known test dataset

	Confusion Matrix for KNN+ABC-20						Confusion Matrix for SVM+ABC-20						Confusion Matrix for RF+ABC-20				
	Actual						Actual						Actual				
	Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R
Predicted	13448	0	0	0	0		13388	32	7	16	1		13447	0	0	1	1
	0	2289	0	0	0		36	2254	1	0	1		0	2289	0	0	0
	0	0	9234	0	0		19	3	9225	4	0		0	0	9234	0	0
	0	0	0	209	0		6	0	1	188	2		1	0	0	208	0
	1	0	0	0	11		0	0	0	1	7		1	0	0	0	10
Accuracy	99.99	100	100	100	100		99.54	98.47	99.90	89.95	63.63		99.98	100	100	99.52	90.9
Average Accuracy	99.99						90.3						98.08				

Table 4.24 Confusion Matrices for ML algorithms optimized by ABC-20 on unknown test dataset

	Confusion Matrix for KNN+ABC						Confusion Matrix for SVM+ABC						Confusion Matrix for RF+ABC				
	Actual						Actual						Actual				
	Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R		Normal	Probe	DoS	R2L	U2R
Predicted	9464	367	1057	1727	40		9406	57	919	2178	24		9449	517	1260	1979	49
	183	1747	119	321	107		227	1942	132	179	144		208	1730	102	251	115
	58	254	6265	177	2		73	422	6407	73	8		51	174	6094	234	2
	5	52	2	419	13		5	0	0	268	8		3	0	2	258	10
	1	1	15	110	38		0	0	0	56	16		0	0	0	32	24
Accuracy	97.45	72.16	84	15.21	19		96.85	80.21	85.9	9.73	8		97.3	71.45	81.71	9.39	12
Average Accuracy	57.56						56.13						54.37				

4.3 Overall Performance Analysis and Discussion

In the previous sections, the algorithms performance on both known and unknown attacks datasets were analyzed. When we optimized the parameters of the algorithms and found the appropriate values for the parameters, the performance of the algorithms are increased. In the following tables, it can be seen how the performance of the algorithms are improved by optimization. The algorithms performances on known attacks datasets are already good, but algorithms performances on unknown attacks datasets increased considerably.

From Table 4.25, it can be seen that on known attacks dataset, performance of the algorithms developed a bit. Following tables show the algorithms performances on known test dataset, as mentioned before; the algorithms performances are already excellent therefore the optimization results are not very outstanding.

Table 4.25 ML algorithms performance on known attacks dataset

Algorithms	CV-Score	Train Time (Sec)	Test Time (Sec)	Memory Usage	Resource Usage	Overall Detection Rate	Overall Accuracy Rate
KNN	0.9978	0.44	2.25	0.25 GB	42.7	0.997	0.998
KNN+PSO-10	0.998	0.83	0.46	0.21 GB	30.8	1.00	0.999
KNN+ABC-10	0.998	0.67	0.57	0.20 GB	21.00	1.00	0.999
KNN+PSO-20	0.998	0.77	0.54	0.27 GB	21.1	1.00	0.999
KNN+ABC-20	0.998	0.69	0.60	0.21 GB	28.1	1.00	0.999
SVM	0.990	79.5	10.23	0.27 GB	26	0.992	0.990
SVM+PSO-10	0.994	51.7	5.65	0.30 GB	29.2	0.995	0.994
SVM+ABC-10	0.993	52.31	4.70	0.22 GB	22.00	0.995	0.994
SVM+PSO-20	0.993	54.4	5.65	0.30 GB	37.2	0.995	0.994
SVM+ABC-20	0.993	45.8	4.33	0.29 GB	27.9	0.995	0.994
RF	0.995	6.33	0.038	0.25 GB	31.10	0.994	0.996
RF + PSO-10	0.998	49.5	0.24	0.21 GB	27.4	0.999	0.999
RF+ ABC-10	0.998	34.12	0.34	0.21 GB	29.8	0.999	0.999
RF + PSO-20	0.998	57.7	0.25	0.11 GB	37.4	0.999	0.999
RF + ABC-20	0.998	52.2	0.31	0.30 GB	42.8	0.999	0.999

From Table 4.26, it can be seen that how optimization improves the performance of ML algorithms on the unknown dataset. The accuracy of KNN has been improved from 78% to 79.5%, and its detection rate has been improved from 72% to 74.9% by PSO optimization, and the accuracy rate improved 79.5%, detection rate has been increased to 75.1% by ABC optimization.

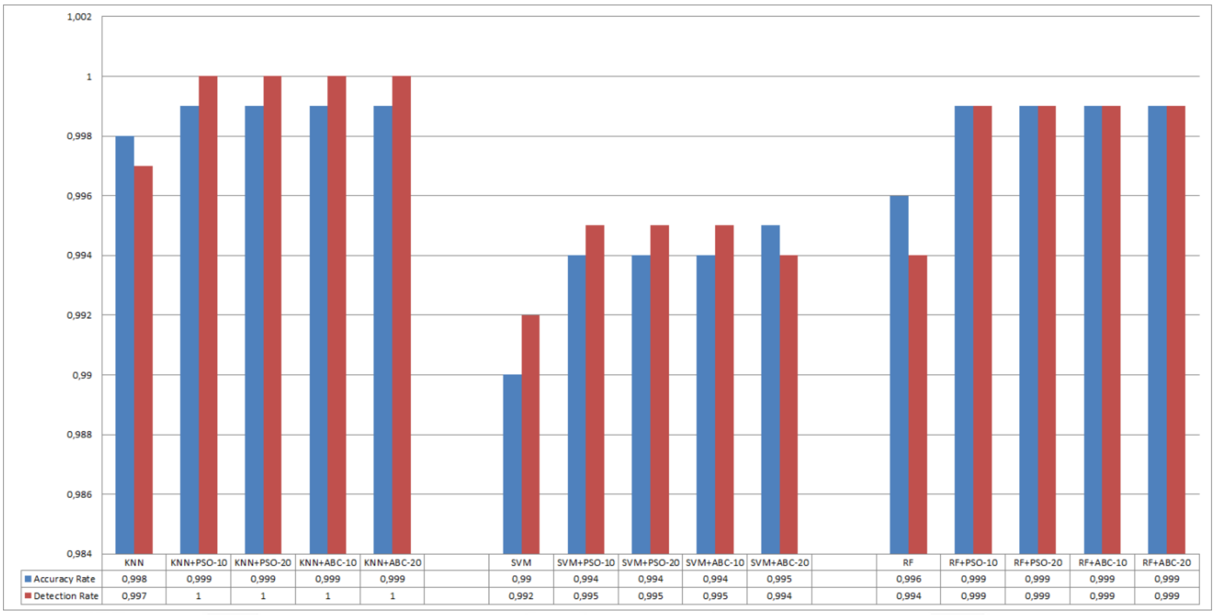
Table 4.26 ML algorithms performance on unknown attacks dataset

Algorithms	CV-Score	Train Time (Sec)	Test Time (Sec)	Memory Usage	Resource Usage	Overall Detection Rate	Overall Accuracy Rate
KNN	0.997	0.36	3.34	0.25 GB	36.7	0.72	0.78
KNN+PSO-10	0.998	0.66	2.25	0.21 GB	25.4	0.749	0.795
KNN+ABC-10	0.998	0.68	1.42	0.20 GB	22.5	0.751	0.795
KNN+PSO-20	0.998	1.23	0.54	0.21 GB	51.1	0.749	0.7954
KNN+ABC-20	0.998	0.62	2.87	0.20 GB	25.4	0.751	0.7955
SVM	0.991	81.27	9.6	0.26 GB	26.4	0.66	0.76
SVM+PSO-10	0.994	53.8	6.78	0.25 GB	25.2	0.683	0.770
SVM+ABC-10	0.994	65.86	5.71	0.23 GB	17.4	0.71	0.77
SVM+PSO-20	0.994	51.9	4.33	0.29 GB	27.9	0.686	0.765
SVM+ABC-20	0.994	110	16.2	0.30GB	13.2	0.684	0.765
RF	0.996	5.70	0.030	0.24 GB	30.20	0.66	0.75
RF + PSO-10	0.998	11.2	0.041	0.20 GB	53.2	0.696	0.76
RF + ABC-10	0.998	22.21	0.15	0.20 GB	24.20	0.709	0.779
RF + PSO-20	0.998	52.2	0.038	0.20 GB	39.6	0.705	0.773
RF + ABC-20	0.998	20.43	0.13	0.20 GB	25.9	0.709	0.778

The SVM performance also improved by optimization, there is an outstanding improvement in the accuracy rate, and also detection rate. PSO improved the accuracy rate and detection rate of the SVM, meanwhile decreased the computational cost. With ABC the accuracy rate has been improved, but the computational cost was increased. The detection rate improved from 66% to 68.6% by PSO and 68.7% by ABC optimization. Moreover the accuracy rate also improved from 76% to 77% by PSO and 76.5% by ABC optimization.

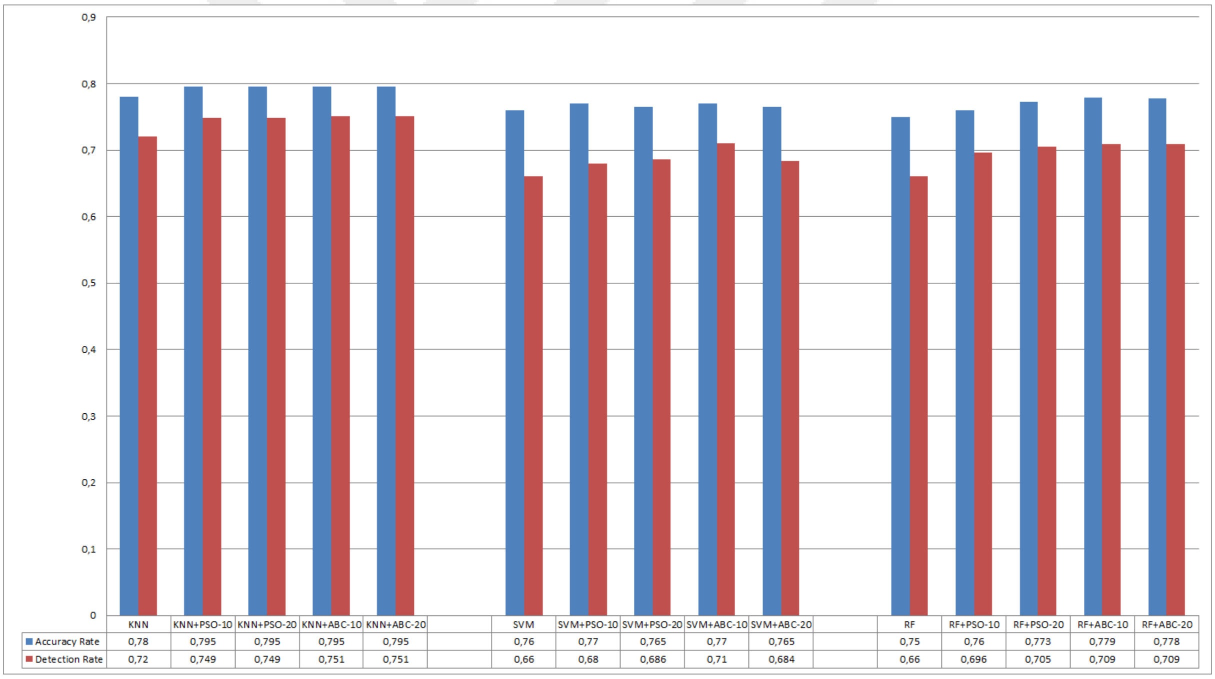
In RF classification, the ABC algorithm is performed better than PSO. ABC improved the accuracy rate of RF from 75% to 77.9%, furthermore the detection has been improved from 66% to 70.9%. In term of resource usage RF+ABC used less resources that RF+PSO and RF which is not very outstanding to be considered. RF+PSO-10 improved the accuracy rate from 75% to 76% and detection rate from 66% to 69.6%. RF+PSO-20 improved the accuracy rate from 75% to 77.3% and detection rate from 66% to 70.5%. In here the RF+ABC-20 performs better than RF and RF+PSO. RF+ABC-10 with accuracy rate of 77.9 and detection rate of 70.9 perform better than other RF-PSO with 10 and 20 particles, and also better than RF+ABC-20.

Figures 4.1 and 4.2 illustrates the ML algorithm performace comparisons. According to these figures, the KNN algorithm performs better than other algorithms in term of network anomaly detection.



	KNN	KNN+PSO-10	KNN+PSO-20	KNN+ABC-10	KNN+ABC-20		SVM	SVM+PSO-10	SVM+PSO-20	SVM+ABC-10	SVM+ABC-20		RF	RF+PSO-10	RF+PSO-20	RF+ABC-10	RF+ABC-20
🟦 Accuracy Rate	0,998	0,999	0,999	0,999	0,999		0,99	0,994	0,994	0,994	0,995		0,996	0,999	0,999	0,999	0,999
🟥 Detection Rate	0,997	1	1	1	1		0,992	0,995	0,995	0,995	0,994		0,994	0,999	0,999	0,999	0,999

Figure 4.1 Comparison of algorithms on known dataset



	KNN	KNN+PSO-10	KNN+PSO-20	KNN+ABC-10	KNN+ABC-20		SVM	SVM+PSO-10	SVM+PSO-20	SVM+ABC-10	SVM+ABC-20		RF	RF+PSO-10	RF+PSO-20	RF+ABC-10	RF+ABC-20
🟦 Accuracy Rate	0,78	0,795	0,795	0,795	0,795		0,76	0,77	0,765	0,77	0,765		0,75	0,76	0,773	0,779	0,778
🟥 Detection Rate	0,72	0,749	0,749	0,751	0,751		0,66	0,68	0,686	0,71	0,684		0,66	0,696	0,705	0,709	0,709

Figure 4.2 Comparison of algorithms on unknown dataset

5. CONCLUSION AND FUTURE WORK

The primary goal of this thesis was to implement network intrusion detection using machine learning and metaheuristics techniques. For this aim, we applied supervised ML algorithms such as KNN, SVM, and RF; then we used PSO and ABC for parameter optimization of these algorithms. The results obtained from using these algorithms are compared to see which of these algorithms perform better in network anomaly detection.

The network connections were simulated by using NSL-KDD benchmark network dataset sampling network traffic data. The dataset contains different records from several network attacks and regular data traffic. All the data records are labeled that makes it easy for the algorithms to identify attacks and normal data patterns. The algorithms are implemented using multi-class classification, and NSL-KDD used as training and testing set. Since the dataset had some redundant and unnecessary features, we used PCA to reduce the elements and makes the more feasible for training and testing.

Different types of experiments conducted in this thesis to find a solution for the problem statement. The performance metrics that shows which algorithm gives a better solution to the problem statement is classification accuracy, detection rate and computational cost. All the algorithms' performances were measured using these metrics. Based on these metrics, the more suitable algorithm for the anomaly detection in a network environment can be selected.

All the experimental results showed that optimized KNN algorithm has a better classification performance. The resource consumption of the RF is less than all other algorithms, but its classification performance is not better than SVM and KNN. Since KNN resource consumption and RF resource consumption is almost the same, so this is not a factor of comparison. The resource consumption of the SVM is more than KNN and RF. According to the unknown attacks results, KNN optimized by ABC improved the accuracy rate from 78% to 79.55% and detection rate from 72% to 75.1% and the PSO version of KNN accuracy rate is 79.54% and the detection rate is 74.9%, which are the best results among all the algorithms examined here. When experimental results are compared to the literature, our result is much better than what is achieved in the literatures. For example in (Voldan, 2016) Voldan achieved 92.47% accuracy for known attacks by KNN while our KNN result on the known dataset is 99.8% with default parameters. Furthermore, his classification performance for SVM was 69% while our was 99% for known data samples and 76% for unknown data samples with default parameters. Table 5.1 shows a comparison between our best results and the literatures for the known dataset.

Table 5.1 A comparison of our proposed method with the literatures (for known dataset)

With Default Parameters			
Method	Year	Dataset	Accuracy
RF (Farnaaz & Jabbar, 2016)	2016	NSL-KDD	RF = 99.8%
KNN and SVM (Voldan, 2016)	2016	NSL-KDD	KNN = 92.47 %, SVM = 69 %
RF and SVM (Roy et al., 2016)	2016	NSL-KDD	SVM = 99.1%, RF = 99.5

KNN, SVM and RF in this thesis (Best results found in the thesis)	2019	NSL-KDD	KNN = 99.8%,
			SVM = 99%,
			RF = 99.6%

With Optimized Parameters			
Method	Year	Dataset	Accuracy
SVM optimized by PSO (Wang, Hong, & Ren, 2009)	2009	KDD CUP'99	SVM (default parameters) = 82.6% SVM + PSO-30 = 99.8%
SVM Optimized by ABC (Wang et al., 2010)	2010	KDDCUP'99	SVM+ABC-20 = 92.7%
SVM Optimized by PSO and ABC (Enache & Patriciu, 2014).	2014	NSL-KDD	SVM+PSO-20 = 98.6%, SVM+ABC-20 = 98.8%
Proposed method in the thesis (Best results found in the thesis)	2019	NSL-KDD	KNN+PSO-10/20 = 99.9% KNN+ABC-10/20 = 99.9% SVM+PSO-10/20 = 99.4% SVM+ABC-10/20 = 99.4 RF+PSO-10/20 = 99.9% RF+ABC-10/20 = 99.9%

Note that accuracy results given in the Table 5.1 are evaluated on known attacks test datasets. For unknown attacks, Table 5.2 shows the best accuracy results of our method. Here, we could not compare our results with the literatures because, there is no studies has been done on the full unknown test dataset.

Table 5.2 Optimized ML algorithms best performance on unknown attacks datasets

KNN			**SVM**			**RF**	
KNN+PSO-20	79.54%		SVM+PSO-10	77%		RF+PSO-20	77.3%
KNN+ABC-20	79.55%		SVM+ABC-10	77%		RF+ABC-10	77.9%

For the future work, it can be worked on detecting minority attacks to improve their detection rate. Detecting minority attacks such as R2L and U2R are not an easy task and they need lots of experiments and algorithms implementation.This can be a subject for another study.

REFERENCES

Ab Wahab, M. N., Nefti-Meziani, S., & Atyabi, A., 2015, A comprehensive Review of Swarm Optimization Algorithms. PLOS ONE, 5(10), 1-36.

Aburromman, A., & Bin Ibne Reaz, A. M., 2016, A novel SVM-KNN-PSO ensemble method for intrusion detection system. Applied Soft Computing, 38, 360-372.

Ahmad, I. (2015). Feature Selection using Particle Swarm Optimization in Intrusion Detection Hinwani Publishing Corporation, 1-8.

Ahmed, A., lisitsa, A., & Dixon, C., 2011, A Misuse based Intrusion Detection System using Temporal Logic and Stream Processing. Paper presented at the 5th Internation Conference on Network Network and System Security (NSS),, Milan-Italy.

Aljarah, I., & Ludwig, S. A., 2013, MApReduce Intrusion Detection System based on a Particle Swarm Optimization Clustering Algorithm. Paper presented at the IEEE Congress on Evolutionary Computation (CEC), Cancun.

Alwas Hussain, N., 2014, Design of a Network Based Anomaly Detection System using VFDT Algorithm, Masters thesis, Mediterranean University, North Cyprus.

Bansal, J. C., Sharma, H., & JAdon, S. S., 2013, Artificial Bee Colony Algorithm: A Survey, International Journal of Advanced Intelligence Paradigm, 123-159.

Bhattacharyya, D. K., & Sharma, H., 2013, Network Anomaly Detection A Machine Learning Perspective: CRC Press.

Blum, C., & Roli, A., 2003, Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison, ACM Computing Surveys, 268-308.

Buczak, A. L., & Guven, E., 2016, a Survey of Data Mining and Machine Machine Learning Methods for Cyber Security Intrusion Detection, IEEE Communication Surveys and Tutorials, 1153-1175.

Butun, I., Morgera, S. D., & Sankar, R., 2013, A Survey of Intrusion Detection Systems in Wireless Sensor Networks. IEEE Communication Surveys and Tutorials, 266-282.

Chung, Y. Y., & Noorhaniza, W., 2012, A Hybrid Network Intrusion Detection System using Simplified Swar Optimization (SSO), Applied Soft Computing, 3014-3022.

Darigue, C., Jang, I. H., & Zeng, W., 2009, A new Data-Mining Based Approach for Network Intrusion Detection, Paper presented at the Seventh Annual Communication Networks and Services research Conferences, Moncton, Canada.

Data, K. C., 1999, Retrieved from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [Accessed in: Jan 2018].

Dhanabal, L., & Shantharajah, S., 2015, A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification, International Journal of Advanced Research in Computer and Communication Engineering, 447-452.

Enache, A. C., & Patriciu, V. V., 2014, Intrusions Detection Based on Support vector Machine Optimized with Swarm Intelligence. Paper presented at the 9th IEEE International Symposium on Applied Computational Intelligence and Informatics, Timisoara, Romania.

Eulogio, R., 2017, Random Forest Introduction. Retrieved from https://www.datascience.com/resources/notebooks/random-forest-intro [accessed in: June 2017].

Farnaaz, N., & Jabbar, M. A., 2016, Random Forest Modeling for Network Intrusion Detection System. Procedia Computer Science, 89, 213-217.

Ganapathy, S., Kulothungan, K., Muthuraj-Kumar, S., Vijayalakshmi, M., Yogesh, P., & Kannan, A., 2013, Intelligent Feature Selection and Classification Techniques for Intrusion Detection in Networks A Survey, EURASIP Journal on Wireless Communications and Networking, 913-921.

Hassan, M. A., Nasser, M., & Pal, B., 2014, Support Vector Machine and Random Forest Modeling for Intrusion Detection System, Journal of Intelligent Learning Systems and Applications, 45-52.

Ji, S. Y., Jeong, B. K., Choi, S., & Jeong, D. H., 2016, A Multi-level Intrusion Detection Method for Abnormal Network Behaviors, Journal of Network and Computer Applications, 9-17.

Karaboga, D., 2010, Artificial Bee Colony Algorithms. Retrieved from http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm [accessed in: Feb 2018].

Knipp, E., Browne, B., Weaver, W., Baumrucker, T. C., Chaffin, L., & Caesar, J., 2002, Managing Cisco Network Security. San-Francisco, USA.

Koehrsen, W., 2017, Random Forest Simple explanation. Retrieved from https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d [accessed in: April 2019].

Kolias, C., & Kambourakis, G., 2011, Swarm Intelligence in Intrusion Detection: A Survey, Computers and Security.

Kumbhar, P., & Mali, M., 2016, A Survey on Feature Selection Technique and Classification Algorithms for Efficient Text Classification, International Journal of Science and Research, 1267-1275.

lappas, T., & Pelechrinis, K., 2004, Data Mining Techique for Intrusion Detection System. UC Riverside, California, USA: Department of computer Science and Engineering.

Li, Z., & Xu, L., 2011, Anomaly Intrusion Detection Method Based in K-Means Clustering Algorithm with Particle Swarm Optimization, Paper presented at the IEEE Internation Conference on Information Technology Computer Engineering and Management Sciences (ICM), Nanjing-Jiangsu-China.

Malik, A. J., & Aslam Khan, F., 2013, A Hybrid Technique using Multi-objectve Particle Swarm Optimization and Random Forest for PROBE Attacks Detection in a Network, Paper presented at the IEEE International Conference in Systems, Man and Cybernetics.

Manning, C. D., Raghavan, P., & Schutze, H., 2009, Support Vector Machine, Introduction to Information Retrieval: Combridge University Press.

Martin, R., 1998, Suricata. Retrieved from https://suricata-ids.org/ [accessed in: June 2017].

Miniwatts, M. G., 2017, Internet Usage by World region Stats, Retrieved from https://www.internetworldstats.com/stats.htm [accessed in: Jan 2017].

Mukherjee, S., & Sharma, N., 2012, Intrusion Detection using Naive Bayes Classifier with Feature Reduction. Procedia Technology, 119-128.

Mulay, S. A., Devale, P., & Garje, G., 2010, Intrusion Detection System using Support Vector Machine and Decision Tree, Internation Journal of Computer Applications, 40-43.

Negnevitsky, M. (2005). Artificial Intelligence: A Guide to Intelligent System.

NG, A., 2017, Machine Learning. Retrieved from http://openclassroom.stanford.edu/main/folder/coursepage.php?course=machinelearning [accessed in: Oct 2017].

NLPCA., 2018, Principal Component Analysis. Retrieved from http://www.nlpca.org/pca_principal_component_analysis.html [accessed in: July 2017].

OISF., 2010, Snort. Retrieved from https://www.snort.org/ [accessed in: April 2019].

Peterson, L. E., 2009, K-Nearest Neighbors. Retrieved from http://www.scholarpedia.org/article/K-nearest_neighbor [accessed in: July 2017].

Python, A. p., 2017, Retrieved from https://anaconda.org/anaconda/python [accessed in: Sept 2017].

Revanthi, S., & Malathi, A., 2013, A Detailed Analysis on NSL-KDD Dataset using Various Machine Learning Algorithms, International Journal of Engineering Research Science and Technology (IJERT), 1848-1853.

Roughgarden, T., 2017, Algorithms. Retrieved from http://class.coursera.org/algo-004/lecture/preview [accessed in: July 2017].

Roy, S. S., Mittal, D., & Biba, M., 2016, Random Foest Support Vector Machine and Nearest Centriod Method for Classifying Network Intrusions, Computer Science Series, 9-17.

Statista., 2017, number of connected devices worldwide. Retrieved from https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ [accessed in: July 2017].

Tesfahun, A., & Bahaskari, D. L., 2013, Intrusion Detection using random Forest Classifier with SMOTE and Feature Reduction, Paper presented at the International Conference on Cloud and Ubiquitous Computing and Emerging Technologies (CUBE), Pune-India.

Tian, J., & Gu, H., 2010, Anomaly Detection Combining one-class SVMs and Particle Swarm Optimization Agorithms, Springer Link, 303-310.

Voldan, H. H., 2016, Anomaly Detection using Machine Learning Techniques, Master thesis, University of Oslo, Oslo, Norwey.

Wang, J., Hong, X., & Ren, R.-r. R., 2009, A Real-time Intrusion Detection System Based on PSO-SVM, Paper presented at the Proceedings of the 2009 International Workshop on Information Security and Application, Qingdao-China.

Wang, J., Li, T., & Ren, R.-r., 2010, a real time IDSs based on artificial bee colony-support vector machine algorithm, Paper presented at the third international workshop in advanced computational intelligence.

Wu, S. X., & Banzhaf, W., 2010, The use of Computational Intelligence in Intrusion Detection Systems: A review, Applied Soft Computing, 1-35.

Xiaohui, H., 2006, Particle Swarm Optimization. Retrieved from http://www.swarmintelligence.org/index.php [accessed in: Oct 2017].

Yan, X., 2011, Metaheuristic Optimization Algorithms. Retrieved from http://www.scholarpedia.org/article/Metaheuristic_Optimization [accessed in: Dec 2017].

Zhang, J., Zulkernine, M., & Haque, A., 2008, Random Forest based Intrusion Detection Systems. IEEE Transactions on System, Man and Cybertics, Applications and reviews, 649-659.

RESUME

PERSONAL INFORMATION

Name and Surname	**:**	Tahira Khorram
Nationality	**:**	Afghanistan
Birth place and date	**:**	Ghazni, 12/05/1992
Telephone	**:**	05365457158
Fax	**:**	----
e-mail	**:**	tahirakhorram92@gmail.com

EDUCATION

Degree		**City, Province**	**Completion Year**
School	:	**Fatimia Aska Davod**	**2009**
University	:	**Kabul University**	**2013**
Master	:	**Selcuk universitesi**	**2019**
PHD	:		

WORK EXPERIENCES

Year	**Corporation**	**Position**
2011-2013	**KU IT Center**	**Help Desk**
2013-2014	**ADRAS**	**IT Officer**

PROFESSION: Network and Cyber Security

FOREIGN LANGUAGES: Persian, English

OTHER SPECIFICATIONS

PUBLICATIONS

Khorram, T., Baykan, N.A., 2018, “Feature Selection in Intrusion Detection System using metaheuristic algorithms”, International Journal of Advance Research, Ideas and Innovations in Technology (IJARIIT), pp.704-710