



**PEOPLES DEMOCRATIC REPUBLIC OF ALGERIA**  
**Ministry of Higher Education and Scientific Research**  
**Mohamed Khider University– BISKRA**  
**Faculty of Exact Sciences and Sciences of Nature and Life**  
**Computer Science Department**

**Order N°: RTIC05 /M2/2022**

**Memoir**

Presented to obtain the diploma of academic master in

**Computer Science**

Path: **Information and Communication Networks and Technologies (RTIC)**

---

**Intrusion Detection based on Machine  
Learning techniques for Software Defined  
Networks**

---

**By:**

**LEHAT NARIMEN**

Sustained on June/27/2022 in front of the juries:

ALOUI IMEN

MCB

President

AYAD SOHEYB

MCA

Rapporteur

ZERARKA NOUR ELHOUDA

MAB

Examinator

Academical year: 2021-2022

# Abstract

As an emerged network paradigm that was developed to reduce network complexity, Software-defined networks (SDN) became widely implemented in different data centers' network environments. Nevertheless, having vulnerabilities makes it prone to different attacks especially DoS and DDoS which tend to target the controller the most to have full access to the whole network which remains a true challenge for manufacturers to solve.

Deploying and performing the intrusion detection systems and techniques for monitoring the malicious activities relies on the quality of the dataset. Therefore, the proposed models based on machine learning for this project are trained on the newly generated InSDN dataset to predict and detect the DoS/DDoS attacks that can occur in the different SDN platform elements.

According to the obtained results, the ANN model that was trained on a specific set of selected features performed better than the Support Vector Machine and Random Forest classifiers.

**Key words:** Intrusion Detection System (IDS); DoS/DDoS attacks, Software Defined Network (SDN), Security, Machine Learning.

# الملخص

كـنـمـوـذـج شـبـكـة نـاشـئـت تم تـطـوـيره لتـقـلـيل تـعـقـيد الشـبـكـة، أـصـبـحـت الشـبـكـات المـعـرـفـة بـالـبـرـمـجـيـات (SDN) مـطـبـقـة عـلـى نـطـاق وـاسـع فـي بـيـنـات شـبـكـات مـرـاكـز البـيـنـات المـخـتـلـفـة. مـع ذـلـك، فـإن وـجـود ثـغـرات أـمـنـيـة يـجـعـلـها عـرـضـة لـهـجـمـات مـخـتـلـفـة خـاصـةً DDoS و DoS الـتـي تـمـيـل إـلـى اسـتـهـدـاف وـحـدة التـحـكـم أكـثـر مـن غـيـرـها للـوـصـول الكـامـل إـلـى الشـبـكـة بـالـكـامـل وـالـتـي لا تـزـال تـمـثـل تـحـديًا حـقـيـقـيًا لـلـمـصـنـعـين لـحـلـها.

يـعـتـمـد نـشـر و تـنـفـيـذ أنـظـمـة و تـقـنـيـات كـشـف التـسـلـل لـمـرـاقـبـة الأـنـشـطـة الضـارـة عـلـى جـودـة مـجـمـوعـة البـيـنـات. لـذـلـك، يـتـم تـدـرـيـب النـمـاـذـج المـقـتـرـحـة القـائـمـة عـلـى التـعـلـم الـآلـي لـهـذا المـشـرـوع عـلـى مـجـمـوعـة بـيـنـات InSDN الـتـي تم إنـشـاؤها حـديـثًا لـلـتـنـبـؤ و اكتـشـاف هـجـمـات DoS / DDoS الـتـي يـمـكـن أن تـحـدث فـي عـنـاصـر مـنـصـة SDN المـخـتـلـفـة.

و فـقـًا لـلـنـتـائـج الـتـي تم الحـصـول عـلـيـها، كان أداء نـمـوـذـج ANN الـذي تم تـدـرـيـبه عـلـى مـجـمـوعـة فرـعـيـة مـحـدـدة مـن المـيـزات المـخـتـارـة أفضـل مـن مـصـنـفـات آلة المـتـجـهات الداعـمـة و مـصـنـفـات الغـابـة العـشـوائـيـة.

**الكلمات الأساسية:** نظام كشف التسلل (IDS)؛ هجمات DoS / DDoS، الشبكة المعرفة بالبرمجيات (SDN)، الأمان، التعلم الآلي.

# Acknowledgement

First, I would like to thank the Almighty God for granting me life and good health as well as supreme protection and guidance throughout my studies.

I am thankful to my supervisor, Dr. Soheyb AYAD, for his informative and useful guidance and suggestions throughout the journey of realizing this project.

I would like to express my gratitude to my friends (**Bouaziz Wafa, Hachouf Chaima, Khadraoui Ikhlas**) for their moral and intellectual support. Knowing they have my back was enough to give me the strength and courage to move on.

Last but not least, a big thank you to my parents for encouraging and supporting me in the construction of this dissertation. their uncountable love and the sacrifices they had to take just to make me the person I am, is something I will always remain grateful for.

Without them, I would not be here.

*Lehat Narimen*

# Table of Contents

List of Abbreviations .....	I
General Introduction.....	III
<b>Chapter 1: Generalities</b> .....	<b>1</b>
1.1. Introduction .....	2
1.2. Software Defined Networking (SDN).....	2
1.2.1. History .....	2
1.2.2. SDN architecture.....	4
1.2.3. Application of SDN.....	7
1.3. Security in SDN.....	9
1.3.1. Security analysis of SDN architecture .....	9
1.3.2. Security attacks and threats to SDN architecture .....	10
1.4. Intrusion Detection System (IDS).....	14
1.4.1. IDS Approaches .....	14
1.4.2. Classification of IDS .....	15
1.5. DOS/DDOS Attacks .....	15
1.5.1. Denial-of-Service (DoS).....	15
1.5.2. Distributed-Denial-of-Service (DDoS).....	16
1.5.3. DDoS Defense mechanisms.....	18
1.6. Conclusion.....	19
<b>Chapter 2: Machine Learning in Networking</b> .....	<b>20</b>
2.1. Introduction .....	21
2.2. Machine Learning .....	21
2.2.1. Machine Learning techniques .....	22
2.2.2. Classification in Machine Learning .....	23
2.3. Feature Selection .....	28

2.3.1.	Feature selection methods.....	28
2.4.	Related work.....	30
2.4.1.	InSDN: A Novel SDN Intrusion Dataset.....	30
2.4.2.	Collaborative detection and mitigation of DDoS in Software Defined Network.....	31
2.4.3.	An evaluation of machine learning methods for classifying Bot traffic in Software Defined Networks	32
2.4.4.	A comparison between the related works .....	33
2.5.	Conclusion.....	34
<b>Chapter 3: ML based Models for Intrusion Detection in SDN .....</b>		<b>35</b>
3.1.	Introduction .....	36
3.2.	General architecture .....	36
3.3.	Objectif.....	36
3.4.	Modeling process .....	37
3.5.	Data preparation and preprocessing .....	38
3.5.1.	Data description .....	38
3.5.2.	Data preparation .....	40
3.5.3.	Data Preprocessing .....	41
3.6.	Feature Selection .....	42
3.7.	Processing .....	42
3.7.1.	RF Modeling .....	43
3.7.2.	ANN Modeling .....	43
3.7.3.	SVM Modeling .....	44
3.8.	Performance Evaluation .....	44
3.8.1.	Confusion Matrix (CM).....	44
3.8.2.	Accuracy.....	45
3.8.4.	Recall.....	46
3.8.5.	F1-score .....	46

3.9.	Model's evaluation and comparison .....	47
3.10.	Conclusion.....	47
<b>Chapter 4: Experimental results and Discussion .....</b>		<b>48</b>
4.1.	Introduction .....	49
4.2.	Environments and development tools .....	49
4.3.	Environment description .....	50
4.4.	Dataset analysis .....	51
4.4.1.	Overview on the dataset .....	51
4.4.2.	Dataset preprocessing.....	51
4.5.	Model training and evaluation.....	59
4.6.	Comparison with related work models .....	70
4.7.	Conclusion.....	71
General Conclusion .....		72
<u>BIBLIOGRAPHY</u> .....		73
APPENDIX A: Source Code .....		78
APPENDIX B: Additional results of feature selection tests .....		83

## Table of Figures

Figure 3. 1: General architecture	36
Figure 3. 2: Detailed architecture	37
Figure 3. 3: Project realization process	38
Figure 3. 4: a sample of InSDN dataset's features [12]	39
Figure 3. 5: Imbalanced dataset histogram	40
Figure 3. 6: RF modeling process	43
Figure 3. 7: ANN modeling process	44
Figure 3. 8: Confusion matrix for binary classification [2]	45
Figure 2. 1: Machine Learning in AI	22
Figure 2. 2: SVM Classifier [9][10]	24
Figure 2. 3: Random Forest Classifier [12]	25
Figure 2. 4: Structure of a neural network [16]	27
Figure 2. 5: Model of the artificial neuron [17]	27
Figure 2. 6: Filter feature selection flowchart [19]	29
Figure 2. 7: Wrapper feature selection flowchart [19]	29
Figure 2. 8: Embedded feature selection flowchart [19]	29
Figure 2. 9: Features ranking [23]	32
Figure 3. 1: General architecture	36
Figure 3. 2: Detailed architecture	37
Figure 3. 3: Project realization process	38
Figure 3. 4: a sample of InSDN dataset's features [12]	39
Figure 3. 5: Imbalanced dataset histogram	40
Figure 3. 6: RF modeling process	43
Figure 3. 7: ANN modeling process	44
Figure 3. 8: Confusion matrix for binary classification [2]	45
Figure 4. 1: Work Environment	50
Figure 4. 2: a csv file dataset	51
Figure 4. 3 : Visualizing Data Classes	54
Figure 4. 4: Results of feature standardization	56
Figure 4. 5 : SVM Evaluation Results	60
Figure 4. 6: SVM Confusion Matrix	61
Figure 4. 7 : RF Evaluation Results	62
Figure 4. 8 : RF Confusion Matrix	62
Figure 4. 9 : ANN Evaluation Results	63



Figure 4. 10 : ANN Confusion Matrix .....	64
Figure 4. 11: Specific_SVM Evaluation Results .....	65
Figure 4. 12: Specific_SVM Confusion Matrix .....	66
Figure 4. 13: Specific_RF Evaluation Results.....	67
Figure 4. 14: Specific_RF Confusion Matrix .....	67
Figure 4. 15: Specific_ANN Evaluation Results .....	68
Figure 4. 16: Specific_ANN Confusion Matrix .....	68
Figure 4. 17: Specific features models' evaluation metrics comparison .....	69

## List of Tables

Table 1. 1: Software-defined networking (SDN) versus traditional networks .....	6
Table 2. 1: Metrics performance for the fully-featured version of the dataset [12].....	31
Table 2. 2: Metrics performance for the SDN specific-featured version of the dataset [12] .....	31
Table 2. 3: Evaluation of V-NKDE classifier performance for all dataset [22] .....	32
Table 2. 4: Classification results of ML models [23] .....	33
Table 2. 5: Comparison between related works.....	34
Table 4. 1: Development tools.....	50
Table 4. 2: Class distribution results and histogram before oversampling .....	57
Table 4. 3: class distribution results and histogram after oversampling ( SMOTE) .....	57
Table 4. 4 : Selected Subset Features.....	59
Table 4. 5 : SVM Hyperparameters .....	60
Table 4. 7 : RF Hyperparameters.....	61
Table 4. 8 : ANN Hyperparameters .....	63
Table 4. 9 : Performance metrics for Fully-Featured dataset models (70:30 VS 80:20) .....	65
Table 4. 10: Comparison between specific models' performance metrics .....	69
Table 4. 11: Comparison between all the models .....	70
Table 4. 12: Comparison with related works models.....	71

## Table of Listings

Listing 4. 1: Preprocessing and Plot Libraries.....	52
Listing 4. 2: Evaluation Libraries .....	52
Listing 4. 3 : Supervised Model Libraries .....	53
Listing 4. 4 Keras Libraries .....	53
Listing 4. 5: Feature Selection Libraries .....	53
Listing 4. 6 : Importing Dataset .....	53
Listing 4. 7 : Features and Label Separation .....	54
Listing 4. 8: Results of missing values check.....	54

Listing 4. 9 : Results of data splitting .....	55
Listing 4. 10: Feature Standardization .....	55
Listing 4. 11: Forward selection method.....	58

# List of Abbreviations

<b>SDN</b>	Software-defined network
<b>DoS</b>	Denial of Service
<b>DDoS</b>	Distributed Denial of Service
<b>IDS</b>	Intrusion Detection System
<b>ForCES</b>	Forwarding and Control Element Separation
<b>ONF</b>	Forwarding and Control Element Separation
<b>API</b>	Application Programming Interface
<b>WAN</b>	Wide Area Network
<b>VM</b>	Virtual Machine
<b>OSI</b>	Open Systems Interaction
<b>IPS</b>	intrusion prevention system
<b>BYOD</b>	Bring Your Own Device
<b>MITM</b>	Man-In-The-Middle
<b>HIDs</b>	Host Intrusion Detection system
<b>NIDs</b>	Network Intrusion Detection system
<b>SYN/ACK</b>	Synchronization / Acknowledgment
<b>ICMP</b>	Internet Control Message Protocol
<b>UDP</b>	User Datagram Protocol
<b>TCP/IP</b>	Transmission Control Protocol/Internet Protocol
<b>HTTP</b>	HyperText Transfer Protocol
<b>ML</b>	Machine Learning
<b>SVM</b>	Support Vector Machine
<b>RF</b>	Random Forest
<b>ANN</b>	Artificial Neuron Network

<b>OOB</b>	Out Of Bag
<b>AI</b>	Artificial Intelligence
<b>FS</b>	Feature Selection
<b>DL</b>	Deep Learning
<b>V-NKDE</b>	Voting- Naive bayes- K nearest neighbors, Decision tree and Extra trees
<b>SMOTE</b>	Synthetic Minority Oversampling Technique-Oversampling

# General Introduction

A Software-defined network (SDN) is a new paradigm that came to light in recent years to solve the conventional network's problems and limits. Unlike the traditional network that implements the network traffic and configures traffic policies on each device independently such as routing, switching and quality of service, SDN separates the control and data plane so the management of the network is carried out by the central controller that has the ability to control and apply network policies to the whole network from a single point. Decoupling the data and control planes gave SDN the ability to make the network more flexible and easier to manage because of the centralized controller as a key benefit. This flexible nature helps in enhancing the security measurements like threat detection and prevention, as it accelerates innovation research compared with conventional networks.

However, despite all the benefits and advantages SDN can provide, like any other network, it is prone to several security issues and exposed to specific attacks. the most dangerous attack that can affect SDN is the DoS/DDoS attacks that can be exploited by attackers to perform malicious tasks, in which the users are denied to access the network services, especially if the target was the SDN controller that will expose the whole network to critical threats. thus, deploying IDS techniques is considered an essential part in order to detect malicious intrusions in SDN network traffic.

Therefore, with the recent advances in the field of Artificial Intelligence and machine learning, we are experiencing more research on SDN security using several machine learning techniques for DoS/DDOS attack detection and prevention.

This project aims to propose and develop models based on machine learning techniques for DoS/DDoS intrusion detection using a newly generated dataset called InSDN.

This dissertation is organized into four chapters, the topics of which are given as follows:

- **Chapter 1:** concerns generalities related to this subject. it starts with a general view of the SDN and how it differs from the traditional networks. followed by discussing the security part, IDS and the DoS/DDoS attacks in SDN.

- **Chapter 2:** presents machine learning, its techniques in general and an overview of the selected models for this work. In the end, some research works that were done on the same topic are discussed.
- **Chapter 3:** explains the followed steps in order to realize the proposed models
- **Chapter 4:** provides the used tools and work environment and discusses the obtained results after evaluating the models.

# **Chapter 1: Generalities**

---

## 1.1. Introduction

Software-Defined Network (SDN) as a new networking paradigm was developed to help overcome the flexibility and scalability limitation of traditional networks and reduce complexity by managing the network centrally. SDN recently has been widely implemented and became a hot topic in the networking community. In spite of that, SDN technology introduces many vulnerabilities and threats that make a real challenge for developers to address it, especially the Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks that made deploying Intrusion Detection System (IDS) for monitoring malicious activities an important and a crucial part of the network.

## 1.2. Software Defined Networking (SDN)

Software Defined Network is a term of the programmable networks paradigm. It is a new networking technology that was designed to make a network more flexible and easier to manage. SDN decouples the control plane (which deals only with the routing and forwarding decisions of networking elements such as routers, switches...) from the data plane (orchestrates the network traffic in accordance with the established configuration in the control plane), in which results the un complication of both the administration and management [1].

### 1.2.1. History

Traditional computer networks have always been known for their complexity and the difficulty to manage the large number of devices its built from, and the huge problems it suffers from such as scalability, classification of data and routing traffic, time consuming, decentralized network control... and so on. Addressing these problems gave the desire to provide a user-controlled management of forwarding in network node, in which was the driver behind the concept of SDN that has been evolving since 1996 [2]. The history of this approach is divided into 3 stages, each has its own contributions:



### 1.2.1.1. Active Networks

Active networking came at a time when the Internet was seeing much more diverse applications and increasing use (mid-1900's to early 2000's) and was the first attempt to make networks programmable. It proposed two programming models: the capsule of model and the programmable router/switch model. The intellectual contributions of active networking to SDN included [3]:

- The notion of programmable functions in the network.
- Network virtualization.
- The ability to have a packet demultiplex into software programs.
- And although it was never realized, active networking did offer a vision of a unified architecture for middlebox orchestration.

### 1.2.1.2. Separating control and data planes

The idea to separate control and data planes appeared in the early 2000's (2001-2007), it developed open interfaces between the control and data planes, thus, two innovations came to light:

- Open interface between the control and data planes (Forwarding and Control Element Separation (ForCES)...).
- Logically centralized control of the network.

Control and data plane separation offered two important intellectual contributions to SDN The first was the notion of logically centralized control using an open interface to routers and switches. The second was technologies and algorithms for achieving distributed state management across a distributed set of network controllers [3].

### 1.2.1.3. OpenFlow

OpenFlow is a communication protocol standard that is used by the controller to manipulate data plane operations, it was created by a group of researchers of Stanford in the mid-2000s. OpenFlow was initially adopted in campuses and then in data centers and now there is more deployments of open flow in a variety of different networks. It offered several intellectual contributions [3]:

- Generalizing the network devices and functions that control data plan to support.
- The vision of a network operating system with three layers: a data plan with an open API, a layer from managing state, and a third control logic layer that affected the data plans based on the state of the network.
- Developed new distributed state management techniques.

### 1.2.2. SDN architecture

SDN basically is composed of SDN devices (include components in which deal with the incoming traffic), SDN controllers and applications. The SDN controller programs the network devices and presents an abstraction of the underlying network infrastructure to the SDN applications. The controller allows an SDN application to define traffic flows and paths, in terms of common characteristics of packets, on the network devices to satisfy its needs and to respond to dynamic requirements by users and traffic/network conditions. The Open Networking Foundation (ONF) defines a high-level architecture for SDN with three main layers as shown in Figure (1.1).[4]

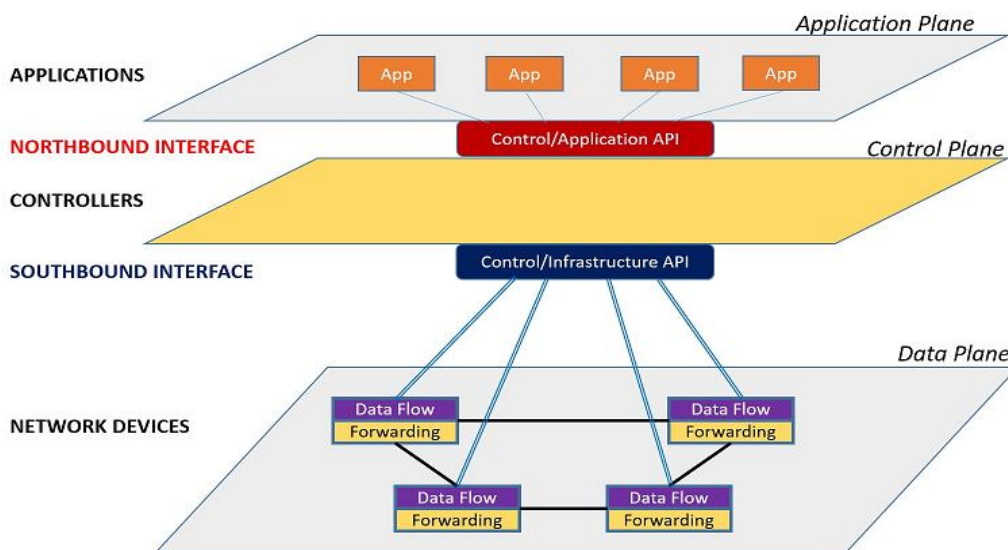


Figure 1. 1: SDN architecture [4]

### 1.2.2.1. Infrastructure Layer

Infrastructure or data layer is the bottom layer, it consists of various networking equipment that form underlying networks to forward traffic such as router, physical/virtual switches, access points etc. The SDN devices are composed of API to communicate with the controller that is positioning in the control layer to manage underlying physical networks. The packet processing function decides on actions to be taken, based on the results of the evaluating incoming packets relative to flow entries in the flow tables.

### 1.2.2.2. Control Layer

An SDN control plane comprises a set of software-based SDN controller(s) to provide control functionality in order to supervise the network forwarding behavior through an open interface. It has interfaces to enable communication among controllers in a control plane, between controllers and network devices through a southbound, and also between controllers and application through a northbound. As mentioned in [5], a controller consists of two main components:

- Functional component, in which the controllers can include more than one like coordinator, Virtualize and so on, to manage the controller behavior.
- control logic that maps networking requirements of applications into instructions for network element resources.

### 1.2.2.3. Application Layer

This layer consists of one or more end-user applications (security, virtualization etc.) that interact with controller(s) to utilize an abstract view of the network for their internal decision-making process [5]. the controller allows the applications to affect the behavior of underlying infrastructure by:

- flows configuration.
- traffic loads balancing across multiple paths.
- reacting to changes in networking topology (link failures, the addition of new devices and paths...).
- redirecting traffic for purposes of inspection.

#### 1.2.2.4. SDN vs Traditional Networks

Conventionally, Networking was made to connect the hardware equipment like routers and switched backed with basic software programs in which used to configure all the connected devices in a network. However, with the technology's development and the wide use of networks, the SDN showed up with many differences on several levels. Alongside the evident variation in the architecture that is shown in Figure 1.2, SDN is a software-based unlike the Traditional Networks that are typically hardware-based, which made room for more differences in many characteristics that are shown in Table 1.1 that was made based on our studies.

<i>Characteristics</i>	<i>SDN</i>	<i>Traditional Networks</i>
<i>Centralized control</i>	centralized control	distributed control
<i>Configuration</i>	automatic	static/manual
<i>Global network view</i>	Central view at controller	difficult
<i>Time required for update/error handling</i>	Quite easy because of central controller(s)	Sometimes it takes months
<i>Programmability</i>	Programmable	Non programmable
<i>Flexibility</i>	More	Less
<i>Implementation</i>	Easy	Hard
<i>Maintenance Cost</i>	low	High
<i>Authenticity, integrity and consistency of controller(s)</i>	important	Not important
<i>Network Management</i>	Easy with the help of the controller(s)	Difficult because changes are implemented separately at each device

Table 1. 1: Software-defined networking (SDN) versus traditional networks

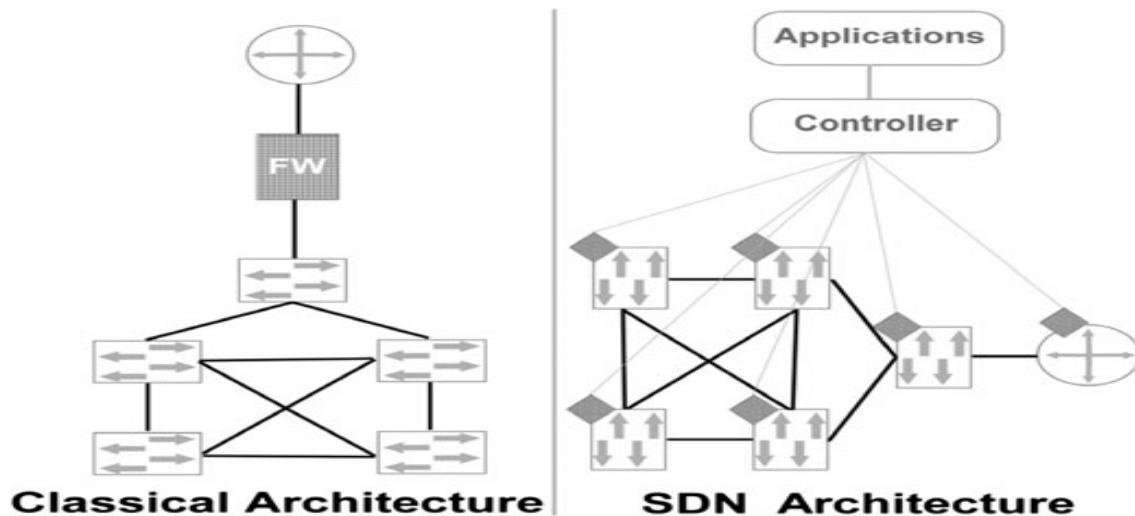


Figure 1. 2: SDN VS Classical Architecture [1]

### 1.2.3. Application of SDN

The use of SDN has been increasing recently, that many companies (Google, Facebook, Microsoft...) have invested in it with their both data centers and WANs (wide area networks). the applications of SDN are classified according to [6] into domains:

#### 1.2.3.1. Data Center Networks domain

Some of the data center networks domain include:

- **Network virtualization** is considered as one of the major applications of SDN in datacenters, for which it is used to realize multi-tenant networks and stretched/extended Networks. Dynamically SDN is applied to create location-agnostic networks and separated topologically equivalent networks across a datacenter, with dynamic reallocation of resources and VM mobility. In this case SDN has many advantages, it helped to improve the recovery time in disasters, offered a better utilization of datacenter resources, faster turnaround times and so on.
- **Service Insertion or Chaining** means creating dynamic chains of L4-7 services (application services running within those OSI layers that provide data storage,

manipulation, and communication services) to accommodate self-service L4-7 service selection or policy based L4-7, for example turning on DDoS protection in response to attacks, self-service firewall, IPS (intrusion prevention system) services in hosting environments. As [7] claimed the use of SDN reduced the provisioning time from weeks to months, as the improvement of agility and self-service provided new revenue and service opportunities with lower costs.

- **Tap Aggregation** that provides total traffic visibility and troubleshooting on any port into network and facilitates optimal performance and security. therefore, using SDN reduces the cost (saves 50-100k per 24 to 48 switches) and leads to less overhead in initial deployment, reducing need to run extra cables from NPBs or every switch.

### 1.2.3.2. Service Provider and Transport Networks domain

The programmatic Operator-Network interfaces provided by SDN allows addressing operator requirements without changing the lower-lever aspects of the network. this flexibility is a result of decoupling the control and data plane so the distribution doesn't need to mimic the distribution of data plane. SDN has many advantages on many use-cases:

- Dynamic WAN reroute: Savings lot of money from unnecessary investment in 10 Gbps or 100 Gbps L4-7 firewalls, load-balancers, IPS/IDS that process unnecessary traffic.
- Dynamic WAN interconnects: Ability to instantly connect and providing ability to enable self-service by reducing the operational expense in creating cross organization interconnects.
- Bandwidth on Demand: reduce operational expense, and increase the agility saving long periods of manual provisioning.

### 1.2.3.3. Campus/Enterprise/Home Networks domain

SDN has known various use-cases with this type of networks, therefore, many applications are explored using SDN such as video streaming, BYOD and network virtualization, application aware routing etc.

### 1.3. Security in SDN

Like with any other network or platform, the software-defined networks require security against every threat, to which they might be exposed.

#### 1.3.1. Security analysis of SDN architecture

The Software-Defined-Networking (SDN) concept moves traditional networking from hardware to software with the benefits of automating and simplifying network operation and administration and improving the network performance. Therefore, because of its design nature, SDN has security advantages, of between it:

- The effective monitoring of abnormal traffic: it is easier to notice any abnormal behavior caused by any attacker because the controller allows to perceive the entire network traffic simultaneously.
- Timely dealing with vulnerabilities, i.e., once new threats are detected, new analyzing software can be programmed to immediately deal with the vulnerability, instead of spending time in waiting for updates.

Unfortunately, SDN technology introduces new vulnerabilities and threat vectors that are inherent to its novel architecture. In fact, SDN also have a lot of security flaws, the most important of which is the "vulnerable controller" that is a very high-value target, where most functions (network information collection, configuration, routing...) are concentrated. in other word, once the controller is controlled, the entire network is controlled. Furthermore, the open nature of programmable interfaces made SDN more susceptible to security threats and causes many risks in view of the fact that it exposes the controller's vulnerabilities and caused interfaces abuse (e.g., embedding malicious code). As the separation of the planes in SDN offered more points to attack (switches, controllers, applications and links between them) as shown in Figure 1.3.[8][9][10]

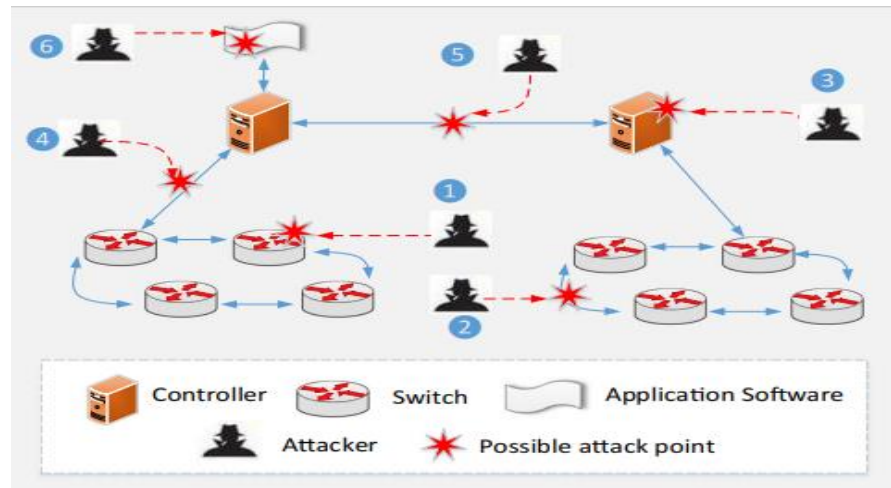


Figure 1. 3: Possible attack points in SDN architecture [8]

### 1.3.2. Security attacks and threats to SDN architecture

The SDN architecture has its unique requirements of security and lacking these requirements make the architecture vulnerable to various attacks and threats [11] that can occur through various points in the network. Each layer and interface in SDN's architecture is sensitive to certain attacks that might compromise the network components in the same layer or target elements in other layers. The virtualized behavior of the SDN makes the network susceptible to new attacks, which are different from those found in the conventional network [12]. The threats are classified into groups according to the layers/points at which it occurs.

#### 1.3.2.1. Infrastructure Layer Security Threats

The Data Plane layer contains a large number of interconnected switches that are responsible for forwarding packets. It is considered as the direct entry point of network access for end-device users. The switch can get attacked by attacking the link to its port. Identifying the possible security threats in this layer is very important, the main attacks are the Man-In-The-Middle (MITM) attacks that occur between the switch and the controller, and DoS attacks to overflow both OpenFlow switch's function modules (Flow Table and Flow Buffer).[8]



#### **1.3.2.1.1. Man-In-The-Middle Attacks (MITM)**

A typical network attack that consists on inserting an agent node between the destination and source nodes to control the communication process. This attack aims to intercepting and tampering the communication data, without being detected by both sides. Specific MITM attacks methods include DNS Spoofing in which the attacker provides fake data, Session Hijacking etc. [13] In SDN, this type of attack takes control of the network packet forwarding by intercepting it with the forwarding rules issued to the switch, and since the switch and controller can be indirectly connected, all switches and hosts connected to them (in a direct way) on the communication path are susceptible to be converted to agent nodes. Achieving MITM attack allows implementing further attacks like Black-hole attacks.[14][15]

#### **1.3.2.1.2. Denial-of-Service (DoS) Attacks**

DoS attack allows the attacker to generate numerous fake packets destined to unknown network devices in a short time period. This can paralyze the legal traffic and prevent forwarding correctly, because the overflow of irregular traffic in the Flow Table will fill up the switch's limited Flow Table storage capacity, hence, it won't be able to insert new rules. Flow Buffer (in which the packets are buffered while searching for a rule or inserting new one before getting forwarded out) as well is another target of DoS attacks. Similarly, to the "Flow Table", flow buffer has a limited storage capacity. Hences, flooding large packets (belong to different flow than the switch normally encounters) lead to its saturation. Thus, there will be no space for the new legitimate packet and get dropped.[8]

#### **1.3.2.2. Control Layer Security Threats**

In SDN the control plane (i.e., the OpenFlow controllers) is a centralized decision- making entity that have a great impact on the whole network. If the controller is hacked the whole network will get effected. As it has direct impact on the forwarding level because the controller is responsible on the forwarding rules (if the switch didn't receive any forward rules from the

controller, it cannot forward packets). Being the main part in the network made it the controller hackers main focus, more exposed to various attacks and faces a lot of challenges.

### 1.3.2.2.1. DoS/DDoS attacks on the controller

DoS and distributed DoS attacks are the most threatening security challenges for the SDN controller. It is an attempt to make a network resource unavailable to legitimate users [11] by exhausting memory resources in both the Control-Plane and Data-Plane [14].

as shown in Figure 1.5, an attacker implements DoS\DDoS out of producing a large amount of flooding traffic in a short period of time to the SDN-Enabled network using its own host or controlling other distributed hosts, that will be mixed with the normal traffic and it will be hard to differentiate between the two types. In this case, the controller will struggle in dealing with the huge amount of Packet\_In messages generated in short time by the flooding traffic. This will lead to consuming all the resources as a try to control the situation and process the normal traffic. At the same time, the bandwidth between the controller and the switches may be fully occupied this will seriously reduce the performance of the whole network.[12]

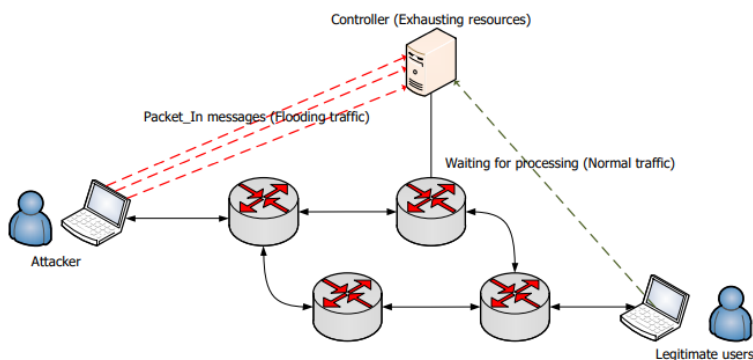


Figure 1. 4:DoS/DDoS Attacks on the Controller [8]

### 1.3.2.2.2. Threats from Applications

The threats to the controller can also come from the applications that run on the controller. For each type of application that has different functional requirements, the network needs to specify a specific security policy. For example, load balancing applications may need to have access to network packet statistics, and intrusion detection applications (IDS) may need to check the header field of packets.[11]

### **1.3.2.2.3. Threats on distributed multi-controllers**

SDN were designed at first as a single controller architecture, but got upgraded to be a distributed control architecture (in form of clusters) as a solution to address the lack of scalability and reliability and process the management of a huge number and variety of devices that cannot be managed by a single SDN controller. This divided the network into different subnetworks, where each individual controller controls specific number of switches, and the controllers can communicate to manage the whole network collaboratively. However, this dividing makes information aggregation and maintaining different privacy rules in each sub-network a challenge. In this situation, an application that passes over multiple network control domains will face numerous security problems, such as authentication, authorization and privacy issues... In addition to the hidden inconsistent configuration threats, that are resulted from the switch-over of the master controller and the coexistence of multiple controllers in a single network domain.[11][8]

### **1.3.2.3. Application Layer Security Threats**

Attackers in the application layer can seize network resources, damage the configuration and steal information, and more of the same, all through the insertion of spyware or malware programs into the application, so in this way, it can damage the network and influence its reliability and availability. Even though OpenFlow deployed flow-based security detection algorithms for security applications, these applications weren't compelling nor mandatory [16]. Besides that, the absence of agreed-upon development environments or network programming models/paradigms led to developing applications with different programming languages, which could cause interoperability inconsistency and security policy conflict. therefore, every malicious application should be stopped early as soon as possible. Some of the security threats to and countermeasures of the application layer are described below.

#### **1.3.2.1.1. Illegal Access**

As known, the applications running on the controller are flexible and extensible and have special rights to control the network behavior and access the resources. Therefore, the lack of a standardized security mechanism for SDN applications causes serious security threats [8] and makes the application layer vulnerable to illegal access. [14]

### **1.3.2.1.2. Security Rules and Configuration Conflicts**

The difference in programming languages of each application is considered as an obstruction to the cooperation of the various applications used in the application layer which cause Security rules and configuration conflict.[15]

## **1.4. Intrusion Detection System (IDS)**

Intrusion Detection System (IDS) is a mechanism or technology that was built to monitor a network or system for intrusions, detect unauthorized access or malicious activities. IDS' main role in a network is to help computer systems prepare to deal with network attacks and take suitable treatments by reporting the detected intrusion. According to [17] Intrusion detection functions include:

- Monitoring and analyzing both user and system activities.
- Analyzing system configurations and vulnerabilities.
- Assessing system and file integrity.
- Ability to recognize patterns typical of attacks.

### **1.4.1. IDS Approaches**

There are two basic categories of intrusion detection techniques: anomaly detection and signature-based detection.

#### **1.4.1.1. Signature based Detection**

This technique is based on a database that consists of attacks descriptions or signatures to be used in the intrusion detection process by comparing the received packet's signature with the ones in the database. This type is widely used in commercial products due to its high detection rate and low false alarms, very effective against the known attacks, unlike the 0-day attacks that he fails to discover. As new attacks are discovered, developers must model and add them to the signature database.[18]

### **1.4.1.2. Anomaly Based Detection**

This type of detection depends on the description and classification of the network to the normal and anomalous, as this classification is based on rules or heuristics rather than patterns or signatures. The implementation of this system needs to know the normal behavior of the network. Unlike the previous type, this type can detect all kinds of intrusion whether it was known or zero-day attack.[17]

## **1.4.2. Classification of IDS**

IDS can be classified based on the serving component as either host-based, network-based:

### **1.4.2.1. Host based IDS (HIDS)**

This type is placed on one device such as a server or workstation, where the data is analyzed locally to the machine and are collecting this data from different sources. HIDS can use both anomaly and misuse detection systems.

### **1.4.2.2. Network based IDS (NIDS)**

NIDS are deployed at strategic points in network infrastructure. The NIDS can capture and analyze data to detect known attacks by comparing patterns or signatures of the database or detection of illegal activities by scanning traffic for anomalous activity, it usually detects attacks such as worms, scans, DoS attacks, botnets, and other types of attacks. NIDS is also referred to as “packet-sniffers” Because it captures the packets passing through the communication mediums.[18]

## **1.5. DOS/DDOS Attacks**

### **1.5.1. Denial-of-Service (DoS)**

Denial-of-Service (DoS) attacks are considered a cyberattack that exploits the internet to target critical Web services. This type of attack is intended to prevent legitimate users from accessing a specific network resource or degrade normal services by sending huge unwanted traffic to the victim (machines or networks) to exhaust services and connection capacity or the bandwidth.

The increasing flow of DoS attacks make servers and network devices on the internet at greater risk, it becomes more like a body with no brain. DoS attacks can cost an organization both time and money while their resources and services are inaccessible.[19]

The DoS attacks have two general methods: flooding services or crashing services. the flood attacks are a result of receiving the system too much traffic for the server to buffer which causes them to slow down and eventually stop. Other different DoS attacks exploit vulnerabilities that damage and crash the targeted system or service. Inputs in these attacks are sent to take advantage of bugs in the target that crash the system so it can be used or accessed.

### **1.5.2. Distributed-Denial-of-Service (DDoS)**

A distributed denial-of-service (DDoS) attack uses multiple hosts to attack against a system, unlike the DoS attack in which a single source performs the attack. The attackers developed specialized malware which they spread to as many vulnerable computers as possible. Malware can spread via compromised websites, email attachments or through an organization's network. Any users tricked into running such malware will unintentionally turn their computer into a bot and provide an access point for attackers to their computer. Once a computer turns into a bot, it connects to the attackers' command and control servers, and it begins to accept orders from these centralized machines. The orders from the command-and-control servers include directions for launching an attack from the bots malware to a particular target using selected attack methods. DDoS allows for exponentially more requests to be sent to the target, therefore increasing the attack power, as the true source of the attack is harder to identify. The motivations behind DDoS attacks can be financially driven, pursuit of crippling a business, competitor, activism, political, or even just for fun.[20][21]

A botnet is an army of bots that usually consists of the infected computers forming a network (known also as zombies). Anytime the botnet owners want to launch an attack, they send messages to their botnets, command and control servers with instructions to perform an attack on a particular target. Any infected machines will comply by launching a coordinated well time distributed attack, known as a DDoS attack. Figure 1.5 show us the difference between DoS and DDoS attacks that was explained above.

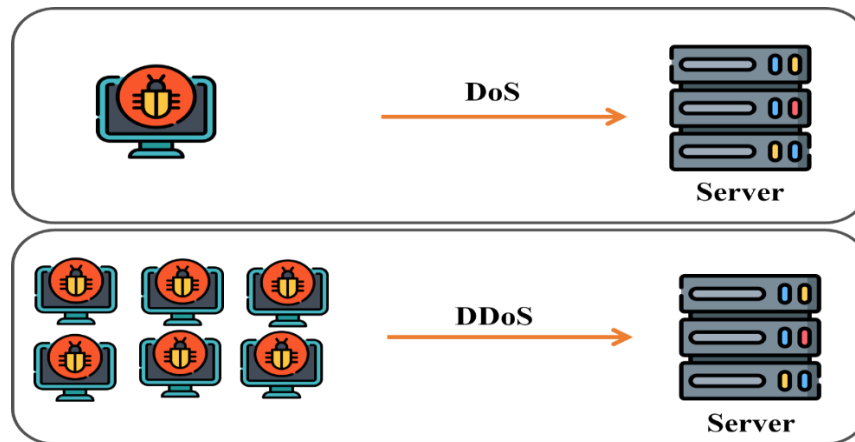


Figure 1. 5: DoS versus DDoS attack

### 1.5.2.1. DDoS attacks Classification

According to [20], the DDoS attacks is classified into two categories: flooding attacks and logical attacks. Flooding attacks creates avalanche of transmitting packets at the victim side which makes the target machine incapable of handling request from the legitimate users. In Logical or software attack, a small number of malformed packets are designed to exploit known software bugs on the target system.

#### 1.5.2.1.1. Flooding Attack Types

The flooding attack types include:

- **SYN Flooding Attack:** the attacker uses spoofed IP addresses to send requests to a server. The server responds by sending the SYN/ACK signal waiting for the ACK signal from its client. But this time no reply comes since the IP is spoofed and the real client is unaware of the ACK signal that the server is expecting. This leaves the half open connections on the server side thus consuming its resources. Therefore, creating thousands and thousands of requests like this can force the server to crash or hang.
- **ICMP Attack:** the attacker sends forged ICMP echo packets to broadcast addresses of vulnerable networks. All the systems on this network reply to the victim with ICMP ECHO replies. This rapidly exhausts the bandwidth available to the target, effectively denying its services to legitimate users.

- **UDP Flooding Attack:** is possible when an attacker sends a UDP packet to a random port on the victim system. The victim system will look for the application waiting on that port. When it realizes that there is no application that is waiting on the port, it generates an ICMP packet of destination unreachable to the forged source address. If flood of UDP packets is sent to the victim machine, the system will surely go down.

#### 1.5.2.1.2. Logic Attack Types

Logic attack types include:

- **Ping of Death:** the target system is pinged with a data packet that exceeds the maximum bytes allowed by TCP/IP.
- **Teardrop Attack:** the attacker sends two fragments (of a packet) that cannot be reassembled properly making use of a bug in the TCP/IP fragmentation re-assembly code of various operating systems by manipulating the offset value of packet and cause reboot or halt the victim system.
- **Land Attack:** An attacker sends a forged packet with the same source and destination IP address. Whenever victim system replies to that packet it actually sends that packet to itself, thus creating an infinite loop between the target system and target system itself thus causing the system to crash or reboot.

#### 1.5.3. DDoS Defense mechanisms

As mentioned in [20], the DDoS defense mechanisms can be classified into prevention, detection, response, mitigation and tolerance. The attack prevention methods try to stop every well know signature-based and broadcast-based attack from being launched, which keep all the devices updated to any new security fixes. It consists of many approaches including anti-DDoS HTTP Throttling, firewall and packets filtering. An attack detection aims to detect an ongoing attack as soon as possible without misclassifying and disrupting legitimate traffic. DDoS detection can be Signature-based or anomaly-based detection. Normally after detecting an attack, the traffic must be blocked from its source, but in this case, the source can't be identified due to using spoofed IP addresses thus making it difficult to trace back. Therefore, mitigation and tolerance aim to reduce



the effect of these attacks on the victim devices by using load balance, better queue management, traffic control scheduling etc.

## **1.6. Conclusion**

In this chapter, we highlighted an overview of some generalities that include Software-defined network and its architecture, difference with traditional network and the security challenges it faces. We also presented an overview that defines the Intrusion Detection System (IDS) and both denial of service and distributed denial of services attacks. The next chapter will focus on the machine learning in networking and some related works analysis study.

# **Chapter 2: Machine Learning in Networking**

---

## 2.1. Introduction

Machine learning technology has been rapidly developed in recent years and used in various fields to solve many problems. Therefore, some studies have begun to introduce machine learning methods into SDN, as a new network architecture that enables to control and define the network through software programming, seeking to solve several problems that cannot be solved using the traditional methods easily and improve the efficiency of network management.

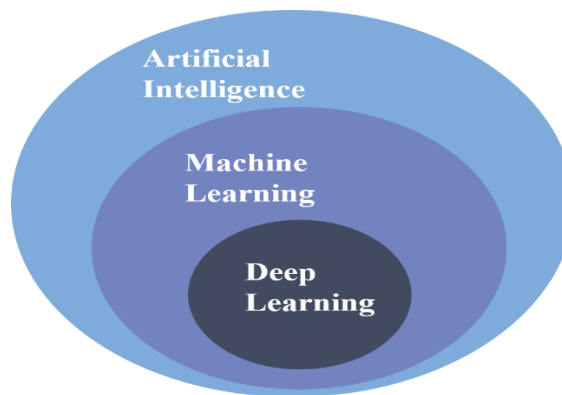
## 2.2. Machine Learning

Machine Learning (ML) is a branch of Artificial Intelligence (AI) as shown in Figure 2.1. It is considered as the capability of AI systems to learn by extracting models from processed data. The “learning” concept is about the ability of the developed algorithms to generalize different behaviors by using the information from the training examples. Using a computer science lexicon, Tom Mitchell presented it as “A computer program is said to learn from experience (E) with respect to some class of tasks (T) and performance measure (P), if its performance at tasks in T, as measured by P, improves with experience E” [22]. Unlike the traditional programming that relies on hard-coded rules and uses data to run the program on the computer to produce the output, in machine learning algorithms, the output of the execution depends on the training phase of the software using the data. Machine Learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data, which mean that the same algorithm can produce different outputs depending on the training data used.

The entire process of machine learning revolves around two main processes: training and prediction. The training process relies on the dataset, in which the model learns the best parameters to minimize the error rate. The dataset is gathered properly according to the achieved outcome and divided into three sets to improve the generalization capabilities of the model: the training set, which the algorithms use to learn the best parameters, the validation set that the model hyperparameters are tuned on and to choose the optimal, finally the test set is used to evaluate the model's final performance. Prediction is the final step of the machine learning process. this is the stage where we consider the model to be ready for practical applications. The challenge for the

model remains whether it can outperform or at least match human judgment in different relevant scenarios as it draws its own conclusion on the basis of its data sets and training.

Machine Learning uses a number of theories and techniques from Data Science: classification, categorization, clustering, trend analysis, anomaly detection, visualization and decision making, these ML techniques and AI are being increasingly used in various functions such as: image processing, healthcare, data mining, video games, robotics, text analysis and so on.



*Figure 2. 1: Machine Learning in AI*

### **2.2.1. Machine Learning techniques**

There are typically two different types of machine learning techniques: supervised Learning and unsupervised Learning, both are typically used for different kinds of machine learning tasks.

#### **2.2.1.1. Supervised Learning**

It uses labeled data, called training data, to build a predictive model to predict the label of unlabeled data. It seeks to create a model that can make predictions about the response values for a new dataset. If larger training datasets are used, it is possible to generate models with superior predictive capacity and, consequently, obtain good results on new sets.[23]

#### **2.2.1.2. Unsupervised learning**

In unsupervised learning, there is no output (unlabeled) associated with the inputs; even the model tries to extract the relationships from the data [24]. the algorithms learn the structure

and representations from the unlabeled inputs. The goal is to model the fundamental structure or distribution in the data to predict unknown data [25]. Unsupervised learning is used as classifying the set of similar patterns into clusters, dimensionality reduction, and anomaly detection from the data [24]. When new data has been given to the trained model, the model puts it in one of the clusters [26].

### **2.2.2. Classification in Machine Learning**

Classification is a supervised learning approach that predicts the outcome of a class type, in other words, it is the process of classifying a set of data into classes, where it can be structured or unstructured data.

The classification model uses the attributes of any type of entity to predict the class of entities. The attributes of the selected entity can be shape, dimensions, color... and so on. These data points can be used to predict the outcome of a class, meaning that the model learns that certain traits belong to certain classes or categories.

The classification model learns that these attributes belong to a particular categorical result in a supervised manner where you map the data points directly to a category label. The class label can be binary such as positive or negative, whether or not the disease is present, whether or not the customer is a returning customer, or whether the job applicant is a successful or unsuccessful one. Some of the main algorithms used in classification models include decision trees, naive Bayes, support vector machines, and neural networks. They all take different approaches to predict the outcome of a class.

#### **2.2.2.1. Support Vector Machine (SVM)**

Support vector machines (SVMs) are one of the most popular supervised machine learning models with related learning algorithms that examine data, distinguish patterns and intended for categorization, which was invented by Cortes and Vapnik in 1995 [27].

SVM is selected as a classification algorithm for its ability to simultaneously minimize the empirical classification error and maximize the geometric margin classification space. These properties reduce the structural risk of over-learning with limited samples. Therefore, it has been applied successfully to image recognition, text categorization, medical diagnosis, remote sensing, motion classification and so on as classification problems.[28]

Yet, SVM's principle is very simple, it separates several classes in the training set with a surface that maximizes the margin between them, i.e., it allows to maximize the model's generalization ability [29].

For a given training data, an optimal hyperplane, by the support vector which maximize the margin between classes, separates (i.e., classify) observations that belong to one class from another based-on linearly or nonlinearly separable patterns of information called features (Figure 2.2). That hyperplane can then be used to determine the most probable label for unseen data.

The features used to infer the hyperplane typically are not raw data, but derivative data from some kind of interpolation during the feature selection stage. Features are further referenced by coordinates based on their relationships to each other and form the support vectors. [30] [31]

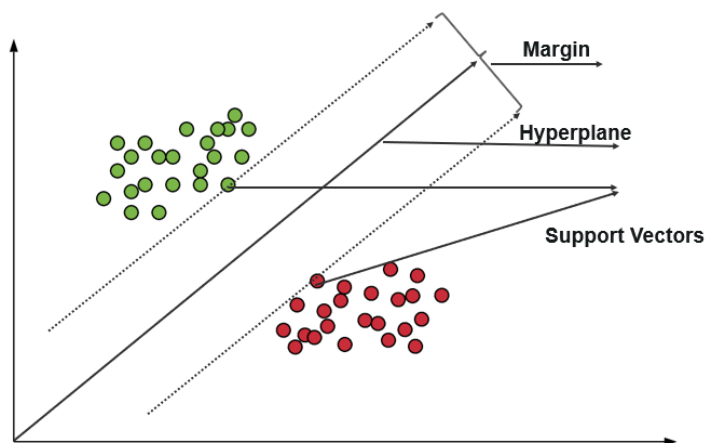


Figure 2. 2: SVM Classifier [30][31]

An SVM model is classified into two types linear classification where the data can be separated linearly as shown in Figure 2.1, and the other one is non-linear. In this case the support vector machine uses a kernel trick to transform into a higher dimensional space so it can be easier to segregate the data points. SVM kernels include linear kernel, polynomial, Radial Basis Function Kernel etc.

#### 2.2.2.2. Random Forest (RF)

Random forest is a popular supervised classification algorithm that was first introduced by Breiman in 2001 [32]. As the name refers to, it is a collection of multiple trees that's known as

decision trees, and the term "Random" is because the algorithm is a forest of randomly created decision trees either by using random feature selection or bootstrap. Hence, the Random Forest algorithm is considered as an advancement to the existing decision trees for the overfitting problem it suffers from. Therefore, RF is considered to be faster and more accurate for complex dataset.

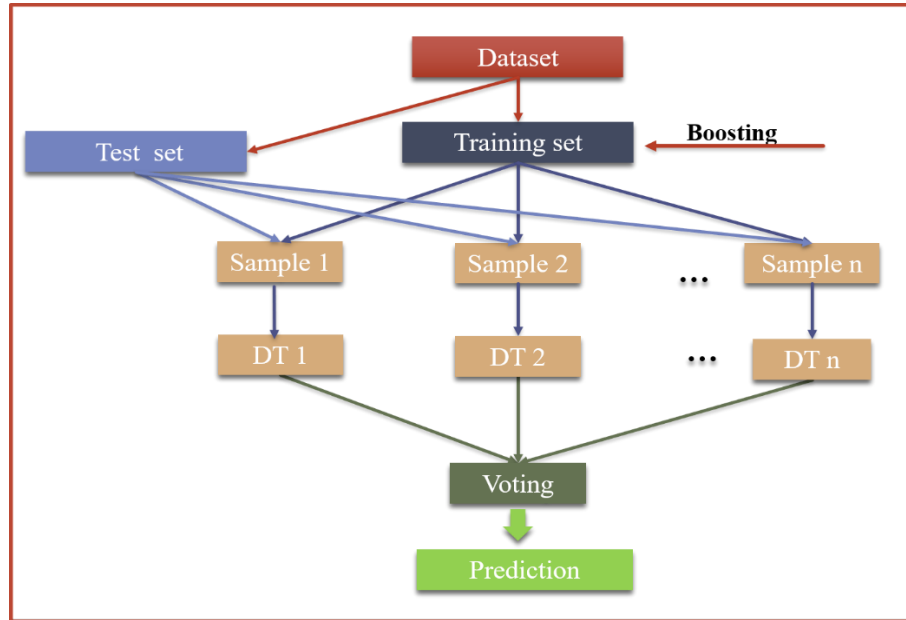


Figure 2. 3: Random Forest Classifier [12]

Random Forest classification algorithm is based on the concept of ensemble learning. As explained in Figure 2.3, it creates multiple trees, each trained on a bootstrapped sample of the original training set, and the results from each tree are aggregated to give a prediction for each observation as the predicted class is calculated based on the majority voting of the trees. [33]

the discrimination function is defined as:

$$H(x) = \operatorname{argmax}_y \sum_{i=1}^k I(h_i(X, \theta_k) = Y) \quad (2.1)$$

In which:  $x$  is the input vector,  $I()$  is the indicator function,  $h(X, \theta_k)$  is a classifier (single decision tree) where  $\theta_k$  represents a random vector for the  $k$ th tree,  $Y$  is the output variable and "argmax<sub>y</sub>" denotes the output value when maximizing  $\sum_{i=1}^k I(h_i(X, \theta_k) = Y)$ . [34]

In the training process, the RF algorithm works on a randomly selected subset of input features or predictive variables, which reduces the generalization error, to determine node split.

To make the trees grow from different training data subsets, the RF algorithm uses bootstrap aggregation or the bagging technique that resample randomly the original dataset with replacement and each subset selected contains a certain proportion of the training dataset, in which it increases the diversity of trees. the absent samples in the training subset (i.e., unselected elements by bootstrapping process that are one-third of the samples) are included in another subset called OOB (out-of-bag), in which is formed for every tree in the ensemble, can be used to evaluate the model performance. [35]

The number of variables is the only adjustable parameter. setting the default value to the square route of the total number of inputs (i.e., limiting the number of used variables for a split) decrease the correlation between the trees and reduce the computational complexity of the algorithm. [36]

### **2.2.2.3. Artificial Neural Networks (ANN)**

Artificial Neural Networks are a supervised machine learning model that was initially derived from the principle of the operation of the human neural system and inspired by the interconnection between them. It was first proposed in the middle of the 20th century but the main revolution was in the 21st century in which the advances in computing made it possible to train complex networks.

Based on multiple input signals, each unit or neuron performs a simple action to produce a single output. the interconnection between the neurons is known as a network, as shown in Figure 2.4, ANNs follow usually a layer-based structure that consists of 3 or more neuron layers, the first is called the “Input layer”, the last is the “Output Layer” and the second one that’s in between is known as the hidden layer, in which, the neurons of each layer are connected by an axon to each neuron of the next layer except the neurons of the last one. [37]



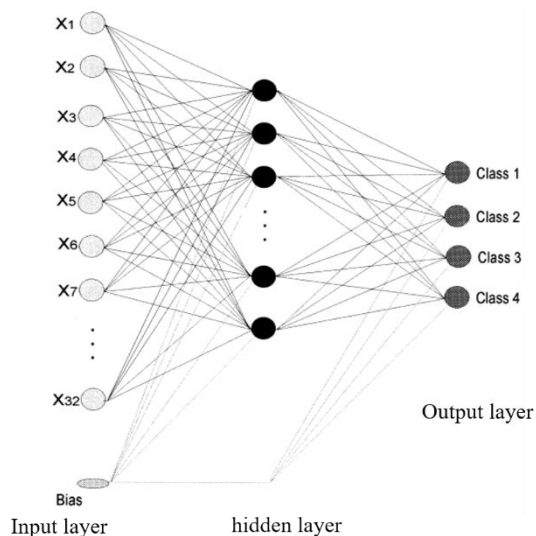


Figure 2. 4: Structure of a neural network [16]

ANNs are one of the most used ML techniques for its ability to model complex non-linear systems. The parameters of ANN are the values of the used models to compute the Output. Hyper-parameters are the parameters that change the learning of the process and the fitting capabilities, its huge quantity to choose from is considered as the main trouble when using ANNs, though it is not a part of the resulting model. [37]

The operation performed in ANN is mathematically defined by the equation:

$$y_i = f_i(z) \quad (2.2)$$

In which  $(i)$  is the neuron, and the  $(z)$  is considered as the argument  $(\sum_{\forall j} w_{i,j}x + b_i)$ .

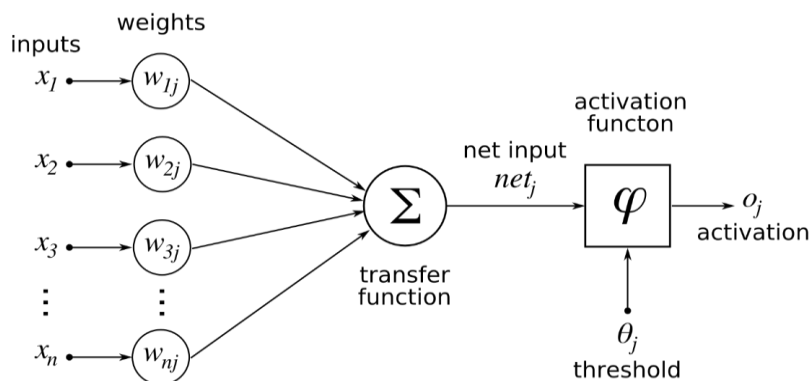


Figure 2. 5: Model of the artificial neuron [38]

As explained in Figure 2.5, for each neuron there will be multiple combined inputs  $x_i$ , each of it has weight assigned to it, in which it is weighted by the parameters  $w_{i,j}$  that are modified during learning process. The weight of each input into the neuron gives it strength, in other word, the input influences the output of the neuron if it has great weight, as it doesn't influence it in the opposite case. The influence of the whole neuron is controlled by the modifiable parameter  $b_i$ .

The activity of neurons is usually determined via the activation function  $f_i(z)$ , in which its usage depends on the type of the network and the layer. There are several different used activation functions, one of which is the Sigmoid function that's considered as one of the oldest and most commonly used non-linear functions that is defined by:  $f_i(z) = \frac{1}{1+e^{-z}}$ . [37]

## 2.3. Feature Selection

Feature selection (FS) is the process of selecting the most important and appropriate sample of features to build an efficient ML model. Employing it aims to narrow down the ML model's set of features to the most relevant ones for a better evaluation metrics rate, all by eliminating the redundant or irrelevant input variables. This process has many advantages include [39]:

- simpler models.
- shorter training time and decreased computational cost.
- model's variance reduction.
- increasing performance of the ML model.

### 2.3.1. Feature selection methods

There are three main feature selection methods [40][41]:

#### 2.3.1.1. Filter method

Filter method selects features based on their statistical score in various tests for their correlation with the outcome variable. Some of the existed filter techniques are Information Gain, Chi-Square Test, Fisher's Score and so on. Figure 2.6 shows the flowchart of filter method.

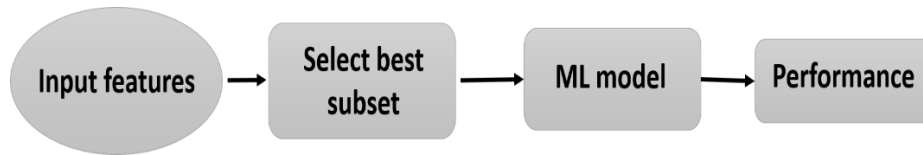


Figure 2. 6: Filter feature selection flowchart [40]

### 2.3.1.2. Wrapper method

Wrapper method generates feature subset using ML algorithms as a part of the feature evaluation function. It splits the data into subsets and use it to train the ML models. Based on the calculated inference and the model's output, it decides whether the feature will be added or removed from the subset. Figure 2.7 shows a detailed flowchart. The wrapper method has several techniques include: forward selection, backwards elimination, genetic algorithm etc.

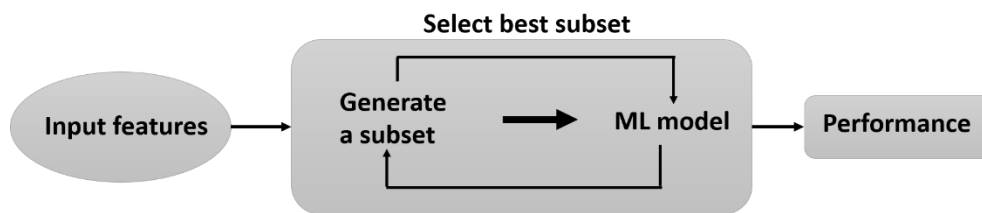


Figure 2. 7: Wrapper feature selection flowchart [40]

### 2.3.1.3. Embedded method

The embedded method is a result of combining the qualities of both the filter and wrapper method to generate the best features subset. Figure 2.8 is detailed flowchart of this method.

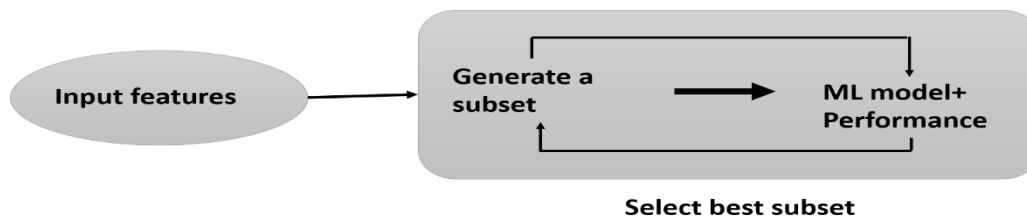


Figure 2. 8: Embedded feature selection flowchart [40]

## 2.4. Related work

In the recent years, researchers started to be more interested with network security, and with the significant increasing of the ML and DL techniques that the last decade has witnessed, many studies have been done on publicly datasets (InSDN dataset) as a try to detect and secure SDN against DoS/DDoS attacks.

### 2.4.1. InSDN: A Novel SDN Intrusion Dataset

Elsayed et al in the article [12], generated a new significant and comprehensive SDN dataset that includes diverse attack categories that can be found in all SDN elements, the attack classes are DoS, DDoS, Prob, web attacks, brute force attack, malware and exploitation. further, they evaluated the quality and demonstrated the use of the proposed InSDN dataset with popular ML techniques for IDSs by performing an experimental evaluation.

Specifically, 8 common supervised learning algorithms have been employed: two SVM based methods: linear kernel (lin-SVM) and a radial basis function kernel (rbf-SVM), three tree-based algorithms: single Decision Tree, RF and adaptive boosting. besides to the K-Nearest Neighbor (KNN) classifier, Naive Bayes and a Multi-Layer Perceptron model.

The performance of all algorithms is tested with two versions of the proposed dataset, a specific-featured version of 48 features that was selected using the same method (information gain) as [42] and a fully-featured version, and trained using the cross-validation technique with  $K = 5$ , where the training and test data are split into 80% and 20%. The results for DoS and DDoS attacks are shown in Table 2.1 and Table 2.2.

<i>Attack Name</i>	<i>Algorithm</i>	<i>Metrics</i>			
		<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Training Time</b>
<b>DoS</b>	<b>RF</b>	0.999876	0.999835	0.999586	61.695
	<b>Rbf-SVM</b>	0.987335	0.997009	0.992148	852.874
	<b>Lin-SVM</b>	0.981379	0.990146	0.990146	202.16
<b>DDoS</b>	<b>RF</b>	0.999992	0.999951	0.999971	41.503
	<b>Rbf-SVM</b>	0.999680	0.999885	0.999783	2825.451
	<b>Lin-SVM</b>	0.999639	0.999967	0.999803	209.679

<i>Merged</i>	<b>RF</b>	0.999931	0.999942	0.999936	231.436
	<b>Rbf-SVM</b>	0.997692	0.999786	0.998738	17161.164
	<b>Lin-SVM</b>	0.999857	0.999898	0.999529	12161.164

Table 2. 1: Metrics performance for the fully-featured version of the dataset [12]

<i>Attack Name</i>	<i>Algorithm</i>	<i>Metrics</i>			
		<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Training Time</b>
<b>DoS</b>	<b>RF</b>	0.999732	0.999567	0.999649	70.569
	<b>Rbf-SVM</b>	0.826327	0.995008	0.902856	2271.213
	<b>Lin-SVM</b>	0.817605	0.994204	0.897298	789.381
<b>DDoS</b>	<b>RF</b>	0.999732	0.999567	0.999649	70.569
	<b>Rbf-SVM</b>	0.999631	0.999672	0.999651	2521.642
	<b>Lin-SVM</b>	0.999557	0.999688	0.999623	113.666
<i>Merged</i>	<b>RF</b>	0.994269	0.996918	0.995592	226.151
	<b>Rbf-SVM</b>	0.963110	0.997622	0.980062	25837.86
	<b>Lin-SVM</b>	0.972192	0.995326	0.991338	152475.72

Table 2. 2: Metrics performance for the SDN specific-featured version of the dataset [12]

#### 2.4.2. Collaborative detection and mitigation of DDoS in Software Defined Network

Tayfour and Marsono in the work [43] proposed a collaborative DDoS detection and mitigation method that consists of a machine learning ensemble method called V-NKDE (Voting -Naive Bayes, K Nearest Neighbors, Decision Tree, and Extra Trees) multiple classifiers to improve DDoS detection accuracy. The performance of the proposed classifier is tested on several datasets including the InSDN dataset. The results and evaluation metrics are shown below in Table 2.3.

<i>Dataset</i>	<i>Metrics</i>			
	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>
<i>InSDN</i>	99.84	99.80	99.80	99.80
<i>CICIDS2017</i>	99.67	99.70	99.70	99.70
<i>NSL-KDD</i>	99.77	99.80	99.80	99.80
<i>UNSW-NB15</i>	98.09	98.10	98.10	98.10

Table 2. 3: Evaluation of V-NKDE classifier performance for all dataset [22]

### 2.4.3. An evaluation of machine learning methods for classifying Bot traffic in Software Defined Networks

Staden and Brown in the article [44] made a comparison of various machine learning algorithms: KNN, SVM, RF, Neural network, logistic regression and multilayer Perceptron, for identifying malicious bot activity on a network. The ML models were tested against the InSDN dataset by extracting the top 36, 18 and 8 features from the original 80 features by using the SELECTKBEST method as shown in Figure 2.9. the dataset was split to 30% test set and a 70% training set. The Table 2.4 represents the classification results the evaluation metrics.

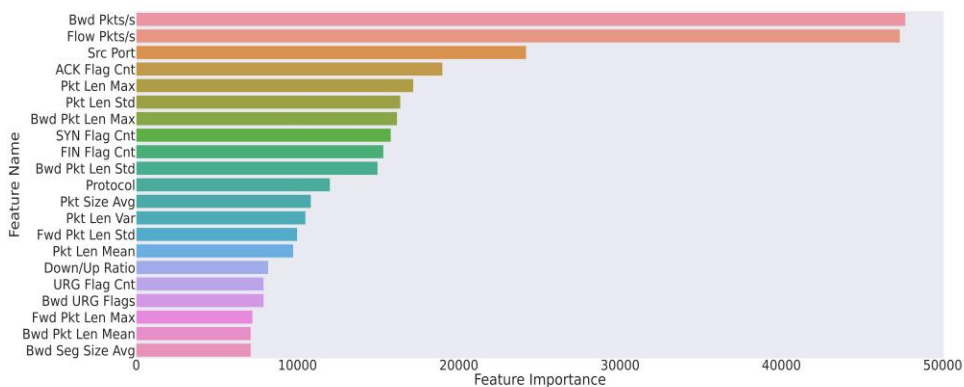


Figure 2. 9: Features ranking [23]

<i>Features size</i>	<i>Metrics</i>	<i>ML Models</i>		
		<i>NN</i>	<i>RF</i>	<i>SVM</i>
<i>36 Features</i>	<i>Precision</i>	77.77	93.53	82.74
	<i>Recall</i>	95.95	92.91	98.82
	<i>F1-Score</i>	80.50	93.18	88.12
<i>18 Features</i>	<i>Precision</i>	72.68	93.59	71.99
	<i>Recall</i>	91.32	91.57	97.33
	<i>F1-Score</i>	78.31	92.44	79.89
<i>8 Features</i>	<i>Precision</i>	59.33	92.36	67.68
	<i>Recall</i>	79.20	90.89	96.85
	<i>F1-Score</i>	64.87	91.52	76.42

Table 2. 4: Classification results of ML models [44]

#### 2.4.4. A comparison between the related works

Table 2.5 shows the comparison between the previous related works with respect to the following metrics: the machine learning methods and model used, datasets and number of features used, the Precision, Recall and F1-score obtained by the applied machine learning algorithms.

<i>Article</i>	<i>ML method</i>	<i>Dataset used</i>	<i>Number of features</i>	<i>Pre-processing</i>	<i>Feature selection</i>	<i>Evaluation metrics</i>
<i>[12]</i>	RF	InSDN 2020 dataset	48 specific features	Yes	Information gain	Table2.1 and Table2.2
	SVM (lin/ rbf)		Fully- featured version			
	V-NKDE	InSDN2020	48 features	Yes	Information gain	
		CICIDS2017	5 features			
		NSL-KDD	41 features			

[43]		UNSW-NB15	48 features			Table 2.3
[44]	Neural Network	InSDN 2020 dataset	Top 36 features	Yes	SELECTKBEST method	Table 2.4
	RF		Top 18 features			
	SVM		Top 8 features			

Table 2. 5: Comparison between related works

After analyzing the previous various related works that address similar issues, it shows that:

- All works trained various machine learning models on the same dataset as ours after going through the preprocessing step but with different features subsets.
- [21] and [43] used the same feature selection method (Information gain that belongs to the filter methods) but [44] used the SELECTKBEST method unlike our chosen feature selected method (Forward Feature Selection).

## 2.5. Conclusion

In this chapter, we presented an overview on the machine learning focusing on the supervised classification approach and 3 of its most known models. Next, we gave a simple glance on the feature selection process and its different methods. At the end, we presented a summary of related works that treated similar issues and problem of detection DoS/DDoS attacks in SDN. The next chapter will introduce and discuss our contributions.



# **Chapter 3: ML based Models for Intrusion Detection in SDN**

---

### 3.1. Introduction

After taking a sufficient look on the machine learning in general and its techniques, this chapter, will present the process methodology and the different steps that we followed to achieve our objective in this work.

### 3.2. General architecture

The general architecture as shown in Figure 3.1 consists of three principal parts:

- Attacks: it indicates the source of the attacks that can infect the SDN datacenter.
- Datacenter infrastructure: an infrastructure that includes SDN datacenter and the legitimate users that would interact with it and at the same time might be an attacker as well.
- ML based models: its essential work is predicting and detecting the DoS/DDoS attacks.

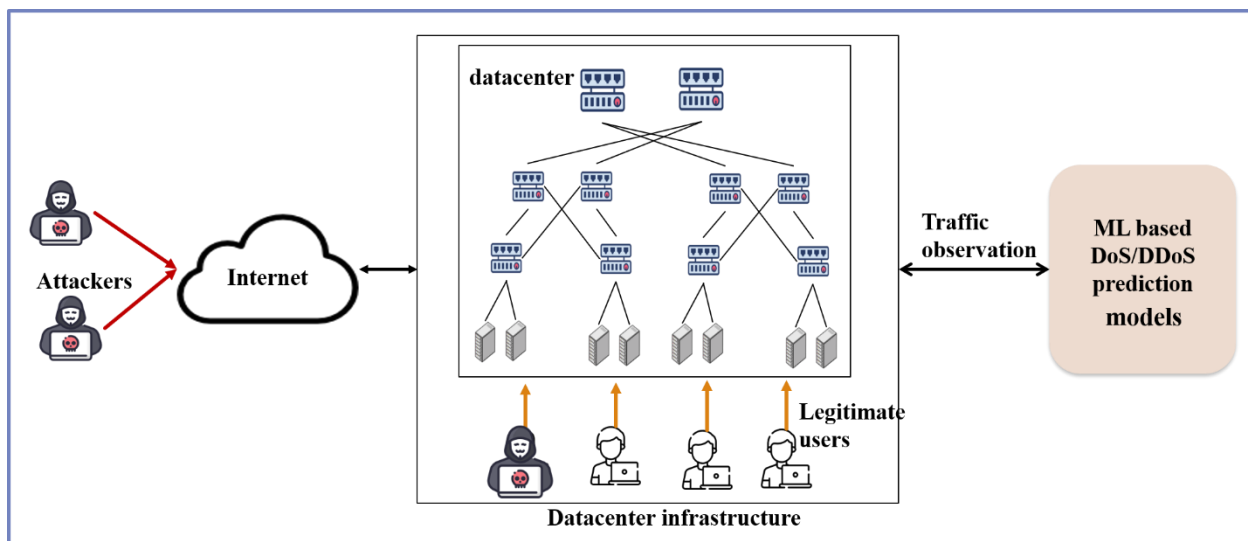


Figure 3. 1: General architecture

### 3.3. Objectif

We aim in our project to develop an efficient model based on machine learning techniques (supervised learning) to predict DoS/DDoS attacks in SDN using the forward selection method for features selection from the InSDN dataset. As shown in the detailed

architecture Figure 3.2 the purpose is to classify the incoming traffic using ML methods into normal or attack (DoS/DDoS) traffic.

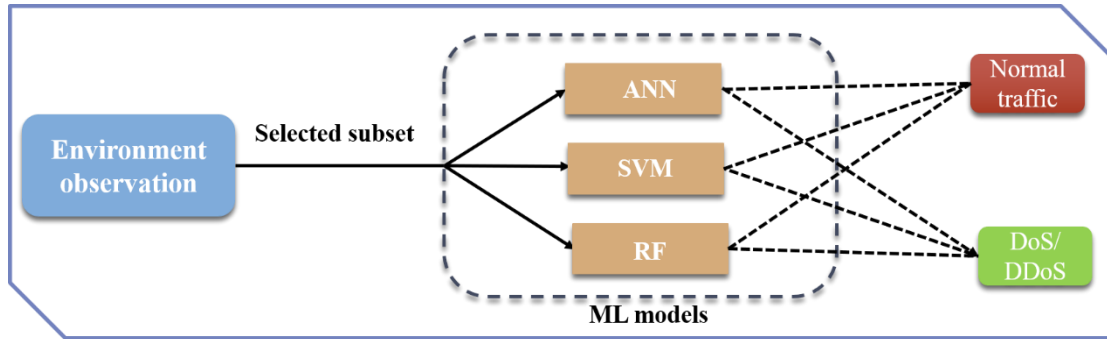


Figure 3. 2: Detailed architecture

### 3.4. Modeling process

The project realization process goes through the following stages and steps, as shown in Figure 3.3, that are applied for all the used algorithms to achieve the best possible performance and results:

- A. Data preparation and preprocessing.
- B. Processing
- C. Models' evaluation.
- D. Models Comparison
- E. Comparative study.

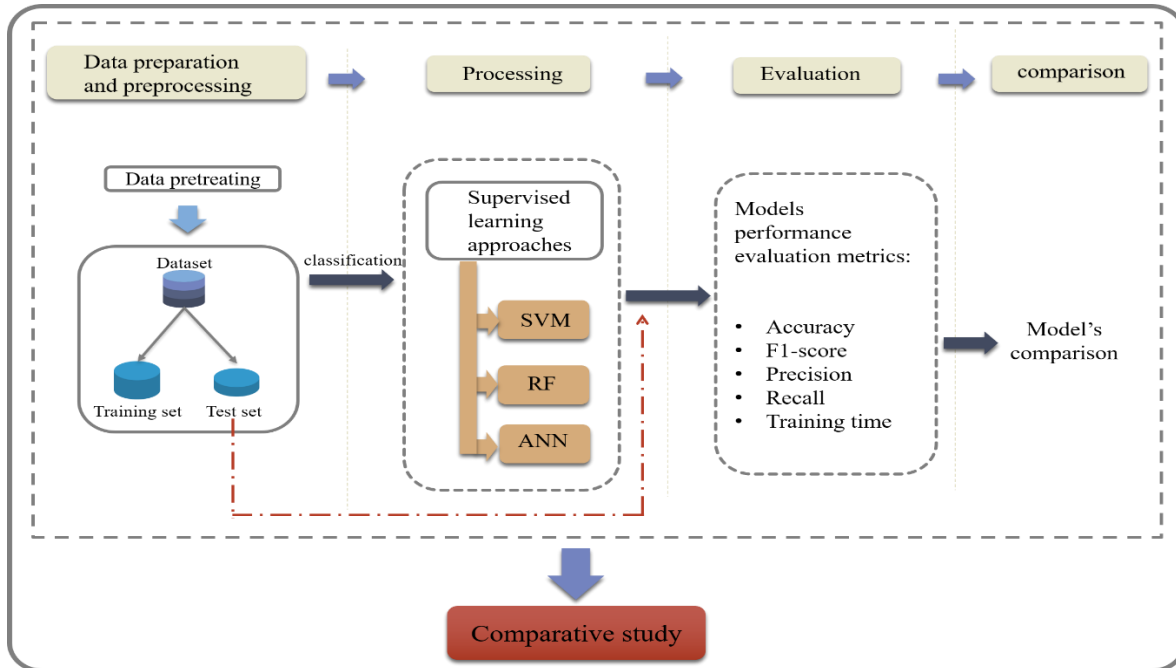


Figure 3. 3: Project realization process

### 3.5. Data preparation and preprocessing

Basically, our project realization process depends on studying the proposed the dataset (InSDN) in [12] and using it for the modeling process.

#### 3.5.1. Data description

InSDN dataset-2020 that was proposed in [12] is a particularly robust dataset for intrusion detection system assessment in SDN. It includes benign and various attack categories including DoS, DDoS, Probe, U2R... attacks. The dataset was divided into 3 groups based on the traffic type and target machines:

- Normal traffic (68424 instances).
- OVS machine' attacks traffic (100884 DoS/DDoS instances).
- Metasploitable-2 server's attack traffic (74674 DoS/DDoS instances).

The InSDN dataset is constructed of more than 80 features (shown in Figure 3.4) that was extracted using the CICFlowMeter tool especially, for its time-based features consideration, and been divided into eight groups as the following:

- Network identifiers attributes: common basic information related to source and destination.
- Packet-based attributes: information about packets.
- Bytes-based attributes: information related to the bytes.
- Interarrival time attributes: interarrival time related information (both backward and forward directions).
- Flow timers' attributes: the information related to the time of each flow (active and inactive).
- Flag attributes: the information related to the flags like SYN Flag, RST Flag, Push flag, etc.
- Flow descriptors attributes: the traffic flow information (the number of packets...).
- Sub flow descriptors attributes: the information related to sub flows (the number of packet and bytes in forwarding and backward directions).

Network-identifiers attributes			
Feature	Feature name	Description	Type
F1	Flow-id	ID of the flow	C
F2	Src-IP	Source IP address	C
F3	Src-Port	Source port number	C
F4	Dst-IP	Destination IP address	C
F5	Dst-Port	Destination port number	C
F6	Protocol-Type	Type of protocol, e.g., tcp, udp, etc.	D
F7	Timestamp	Timestamp	C
Byte-based attributes			
F8	Fwd-Header-Len	Total bytes used for headers in the forward direction	C
F9	Bwd-Header-Len	Total bytes used for headers in the backward direction	C
Packet-based attributes			
F10	Tot-Fwd-Pkts	Total packets in the forward direction	C
F11	Tot-Bwd-Pkts	Total packets in the backward direction	C
F12	TotLen-Fwd-Pkts	Total size of packet in forward direction	C
F13	TotLen-Bwd-Pkts	Total size of packet in backward direction	C
F14	Fwd-Pkt-Len (Min, Mean, Max, Std)	Min, Mean, Max, and standard deviation of the size of packet in forward direction	C
F15	Bwd-Pkt-Len (Min, Mean, Max, Std)	Min, Mean, Max, and standard deviation of the size of packet in backward direction	C
F16	Pkt-Len (Min, Mean, Max, Var, Std)	Min, Mean, Max, Var and standard deviation of the length of a packet	C
F17	Pkt-Size-Avg	Average size of packet	C
Interarrival Times attributes			
F18	Duration	Duration of the flow in Microsecond	C
F19	Flow-IAT (Min, Mean, Max, Std)	Min, Mean, Max, and standard deviation of the time between two packets sent in the flow	C
F20	Fwd-IAT (Tot, Min, Mean, Max, Std)	Tot, Min, Mean, Max, and standard deviation of the time between two packets sent in the forward direction	C
F21	Bwd-IAT (Tot, Min, Mean, Max, Std)	Tot, Min, Mean, Max, and standard deviation of the Time between two packets sent in the backward direction	C
Flow timers attributes			
F22	Active-Time (Min, Mean, Max, Std)	Min, Mean, Max, Standard deviation of the time flow was active before becoming idle	C
F23	Idle (Min, Mean, Max, Std)	Min, Mean, Max, Standard deviation time flow was idle before becoming active	C
Flag-based attributes			
F24	Fwd-PSH-Flags	Number of times the PSH flag was set in packets travelling in the forward direction	D
F25	Bwd-PSH-Flags	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)	D
F26	Fwd-URG-Flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)	D
F27	Bwd-URG-Flags	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)	D
F28	FIN-Flag-Cnt	Number of packets with FIN	D
F29	SYN-Flag-Cnt	Number of packets with SYN	D
F30	RST-Flag-Cnt	Number of packets with RST	D

Figure 3. 4: a sample of InSDN dataset's features [12]

### 3.5.2. Data preparation

As a first step of preparing the InSDN dataset that contains several attack types, we had to get rid of the unwanted types and kept only DOS/DDoS attacks because that's what our project aims to detect. Figure 3.5 shows us the large gap between the samples of the categories, as we can clearly see that the “malicious/attack” traffic category makes up the majority compared to the “normal” traffic category which defines the case of imbalanced dataset.

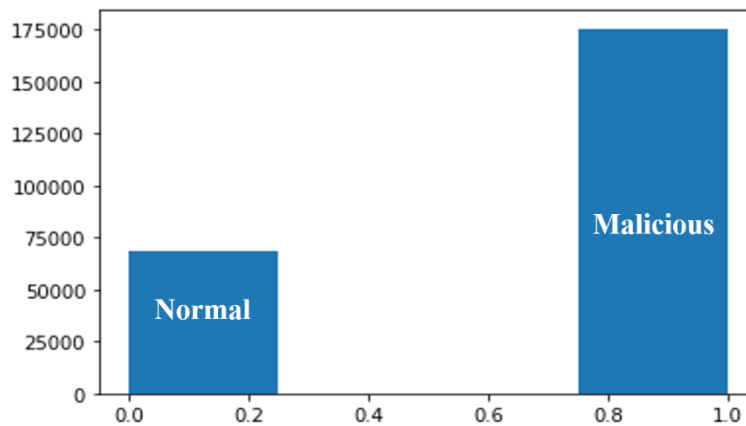


Figure 3. 5: Imbalanced dataset histogram

An imbalanced dataset is a term that usually refers to a classification problem that is defined as the non-uniform distribution of classes which generally contains two classes: majority class and minority class.

The imbalanced dataset leads to the build of biased and inaccurate models therefore balancing it is an important step for a more accurate result. There are several ways and algorithms that are widely used to deal with imbalanced class distribution:

- Oversampling
- Undersampling
- Generating synthetic data (ex: SMOTE method).

### 3.5.3. Data Preprocessing

Raw data is known that's often incomplete, missing certain values, inconsistent and subjected to numerous errors. Preparing and fitting such a dataset into a machine learning model is known as "**Data preprocessing**". It is one of the most important steps to go through in processing and dealing with a brute dataset, and indeed is a proven method to solve dataset problems.

In our project, we as well went through data preprocessing steps that are explained below:

#### **Step1: Importing libraries and dataset**

Before starting the preprocessing, importing libraries and datasets is a must. As it's known, libraries are extra helpful in storing frequently used routines which helps to simply access the dataset that can be imported in several ways depending on its format (.xlsx, .csv...).

#### **Step2: Missing values verification**

“Missing values” is when some participants or values have no stored data due to incomplete data entry or lost files.... It might cause performance reduction, therefore properly handling the missing values is an essential step of data preprocessing.

#### **Step3: Splitting data**

Any machine learning algorithm needs to be tested for various metrics. Therefore, the dataset should be divided into a training set and a testing set. this means that the model will be trained on the training set to teach it the present behaviors in data and evaluated on a different set (test set) that it has not yet encountered to assess whether it has generalized well from the data that it has already seen. There are numerous ratio ways to split the dataset. In our project, to construct the training and evaluation set, we chose both 70:30 and 80:20 ratios, in order to compare and choose the best one.

#### **Step4: Feature standardization**

A crucial part of the data preprocessing stage. it is a method that transforms the features in a dataset and normalizes the variables within a specific range to compare them for common patterns. [45]

There are methods to do the feature scaling step including standardization and normalization. Since there is no hard rule to tell when and what is best method to use, we can always fit the model, normalized and standardized data and compare the performance for better results.

### **3.6. Feature Selection**

The main goal behind using the feature selection step is to select the features that would yield minimum classification error. As mentioned in Chapter 2 section 2.3, among the several feature selection techniques and methods, we chose the Forward selection for its ability to select the best features and optimize the models' performance.

Forward selection is an iterative method that uses the searching technique in order to select the features. The whole process consists of having an empty set of features, starts adding selected features by an evaluation function, that, at each iteration, selects the best feature that would create the best performing model until it meets certain criteria.

### **3.7. Processing**

This work uses three of the common supervised learning algorithms: Support Vector Machines, Random Forest and Artificial Neural Network for classifying our dataset. Choosing these three relies on the nature of the used dataset and the addressed problem whereas:

- The type of classification in our case is a binary classification, therefore the common ML methods used for it include ours.
- We chose random forest over decision tree because it is generally a better model if the goal is for prediction, as it gives better accuracy and reduces the chances of overfitting.

The modeling of the used classification techniques is explained below:



### 3.7.1. RF Modeling

Figure 3.6 illustrates the modelling of a random forest, that takes specific features as inputs to build multiple decision trees based on it, each tree represents a possible occurrence or response. Finally, it will contribute to the decision and predicts the outcome as if there is an attack or not with a majority vote.

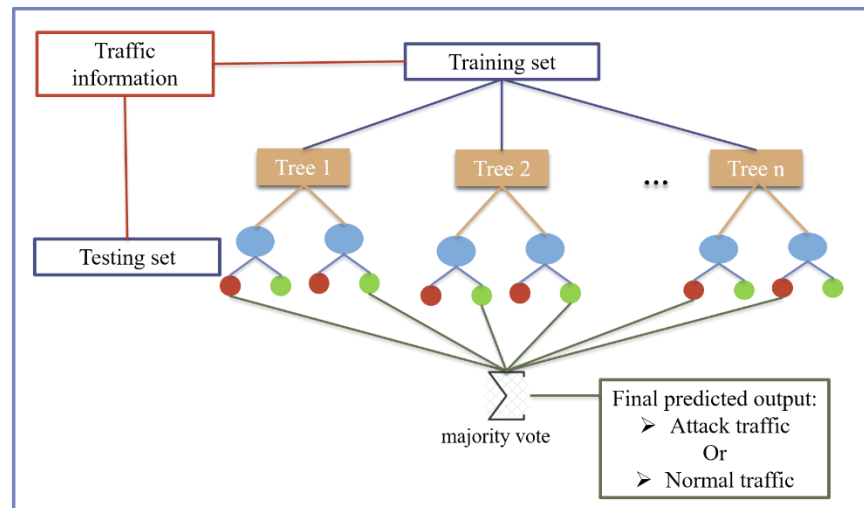


Figure 3. 6: RF modeling process

### 3.7.2. ANN Modeling

The artificial neural network receives a subset of selected features as inputs so it can determine an output that represents the appropriate class which is either 0 or 1. The whole process can be summarized in few steps as shown in Figure 3.7:

- The received inputs get to be multiplied by the assigned weight  $w$ .
- adding the multiplied values to form the weighted sum.
- applying a relevant activation function on the weighted sum of the inputs and their perspective weights.
- the enable function maps the input to the corresponding output.

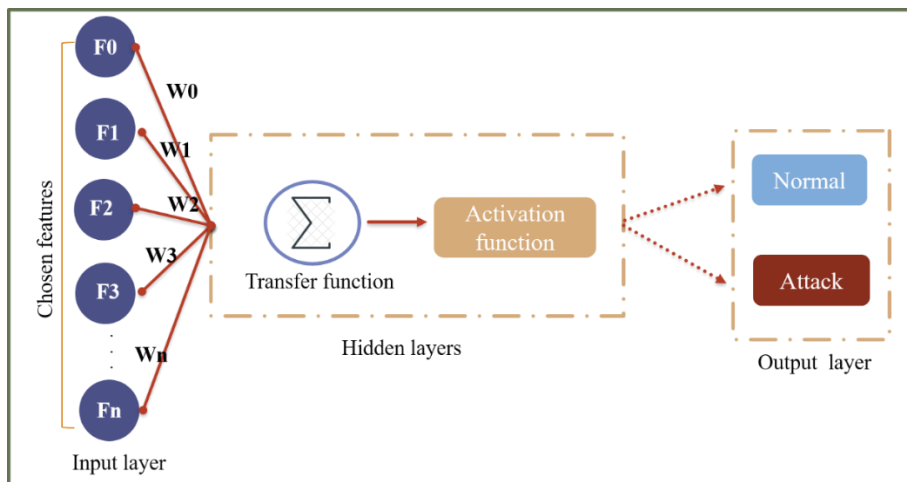


Figure 3. 7: ANN modeling process

### 3.7.3. SVM Modeling

After feeding the SVM classifier a subset of selected features, it will start learning what a DoS/DDOS attack traffic looks like and how it is different from normal traffic. the process consists of taking all the data points into consideration to find the optimal hyperplane (decision boundary) that decides properly both out classes (normal/ attack) and maximizing the margins as possible to avoid data misclassification.

## 3.8. Performance Evaluation

Evaluating the ML algorithms after applying them is an essential step to make sure it is working properly using the performance evaluation metrics. in our work, we relied on using the most important performance metrics to evaluate our models and do a comparative analysis such as precision, recall, f1-score along with accuracy. it's known that these metrics are commonly used for intrusion detection systems.

### 3.8.1. Confusion Matrix (CM)

A confusion matrix is a performance measurement for classification problems and is extremely useful in measuring some metrics like Recall, Precision.... It is an  $n*n$  matrix where  $n$  is the number of targeted classes (in our study  $n = 2$  because we have two classes either 1 or 0, i.e., a binary classification). It defines the number of correct and incorrect predictions a classifier makes. As shown in Figure 3.8 that illustrates an example of a CM,

the rows represent the predicted values and the columns represent the actual values that are:

- **True Positive (TP):** is the number of positive outcomes that are positively predicted, in this case the number of attacks that are classified as an attack.
- **True Negative (TN):** defines the false outcomes that were positively predicted, i.e., the number of normal traffic that are detected as normal.
- **False Positive is (FP):** the number of positive outcomes that are predicted negatively, i.e., the number of normal traffic that are detected as attack.
- **False Negative (FN):** is the number of negative outcomes that are predicted negatively. i.e., the number of attacks that are detected as normal.

		Actual Values	
		Positive	Negative
Predicted Values	Positive	TP	FP
	Negative	FN	TN

Figure 3. 8: Confusion matrix for binary classification [46]

### 3.8.2. Accuracy

The most used metric to evaluate the performance of a classification problem's model. it defines the ratio of the correctly predicted data from the total number of made predictions for a dataset. It is a simple metric for model evaluation and suits the balanced data unlike unbalanced one. It is defined as [47]:

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}} \quad (3.1)$$

Or can be defined using positives and negatives for binary classification:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.2)$$

### 3.8.3. Precision

It defines the ratio of the actual positive observations that were predicted positively, and the more predictive false positives there are the lower "precision" gets. it is considered a useful metric for skewed and unbalanced datasets. It is defined as [47]:

$$Precision = \frac{TP}{TP + FP} \quad (3.3)$$

### 3.8.4. Recall

known also as Sensitivity or True Positive Rate (TPR), defines the ratio of the correctly identified positive instances. as more false negatives are predicted, the recall gets lower. it helps to measure the model's ability in detecting positive samples. It is defined as [47]:

$$Recall = \frac{TP}{TP + FN} \quad (3.4)$$

### 3.8.5. F1-score

It is a combination between Recall and precision. it defines the weighted average of the two previous metrics. it has a range of [0,1]. The higher the F1- score, the better model's performance we get. It is defined as [47]:

$$F1 = \frac{2 \cdot precision \cdot Recall}{Precision + Recall} \quad (3.5)$$

### **3.9. Model's evaluation and comparison**

After preprocessing the dataset, building the ML models, and training and testing them comes the evaluation step in which we analyze and discuss the obtained results of each model to compare them and choose the best one for predicting the DoS/DDoS attacks. Then again, we compare our results with the related works' results that were mentioned in the Chapter2 section (2.2).

### **3.10. Conclusion**

This chapter presented the general and detailed structure to which we explained the different phases of the project<sup>s</sup> which summed up in the data analysis, used ML models and the evaluation techniques. The next chapter will be about the implementation of what we had in this one followed by discussing the obtained results.

# **Chapter 4: Experimental results and Discussion**





---




## 4.1. Introduction

In this chapter, we will present and discuss the different experimental results of the proposed solution that we conceptually described in chapter 3. As an opening to this chapter, we'll present the environment and tools used for the implementation of this project, after that, we'll move right ahead to the description of our used dataset and the obtained results, then the performance of the classification methods will be evaluated using the chosen evaluation metrics. Finally, a comparative study between the different methods shall be presented.

## 4.2. Environments and development tools

Table 4.1 shows the different environments and tools we used to realize the project.

Tool	Description
	<b>Google Colab</b> is a cloud service, offered by Google (free), based on Jupyter Notebook and intended for training and research in machine learning. This platform makes it possible to train Machine Learning and Deep Learning models directly in the cloud. Without therefore having to install anything on our computer except a browser.
	<b>Python</b> is an interpreted, object-oriented, high-level programming language with dynamic semantics [48]. It is in high demand by a large community of developers and programmers. Python packages (libraries) encourage code modularity and reusability. It is used for web development, AI, machine learning, operating systems, mobile app development, video games and many more.
	<b>NumPy</b> is an extension of the Python programming language, intended to manipulate matrices or multidimensional arrays as well as mathematical functions operating on these arrays.
	<b>Matplotlib</b> is an open-source Python library for creating data visualizations.

	<p><b>Pandas</b> is a Python written library for data manipulation and analysis. it offers data structure and operations for manipulating numerical arrays.</p>
	<p><b>Keras</b> is a neural network API written in Python language. It is an open-source library, running on top of frameworks such as Theano and TensorFlow. Designed to be modular, fast and easy to use, Keras was created by Google engineer François Chollet. It offers a simple and intuitive way to create Deep Learning models.</p>
	<p><b>Scikit-learn</b> is a machine learning free python library. It notably includes functions for estimating classification algorithms. It is designed to harmonize with other free python libraries, including NumPy.</p>

*Table 4. 1: Development tools*

### 4.3. Environment description

In order to train our dataset, we used Google Colab which is a free cloud service provided by Google that supports free GPU, and it offers us a single 25GB NVIDIA Tesla K80 GPU that can be used for up to 12 hours continuously. (See Figure 4.1)



```

PFE-M2_increased RAM.ipynb
File Edit View Insert Runtime Tools Help Last edited on June 20
+ Code + Text
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
importing libraries
[ ] #plot library
import matplotlib.pyplot as plt
from matplotlib import pyplot
from itertools import cycle
from scipy import interp
import seaborn as sb
#dataset preprocessing
from sklearn.preprocessing import MinMaxScaler
from collections import Counter
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import SMOTE
#####
import numpy as np
import pandas as pd
from time import time
from sklearn.model_selection import train_test_split

```

*Figure 4. 1: Work Environment*



## 4.4. Dataset analysis

### 4.4.1. Overview on the dataset

The used dataset contains 84 columns and 343889 rows in total, all structured in a .CSV file. After we got rid of the unwanted attack types, and as we removed all the socket features such as source IP, destination IP and so on in order to avoid the overfitting problem, where these features can be changed from network to another, the final dataset now includes 77 columns and 243982 rows, where:

- Columns from 1 to 76 are attributes (features).
- The 77th column represents the target class (label) of the dataset which can be either 0 or 1, which represents normal and DoS/DDoS attack traffic respectively.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Protocol	Tot Fwd Pkts	Tot Bwd Pkts	TotLen	Fwd   Totten Bwd	Fwd Pkt Len	Fwd Pkt Len	Fwd Pkt Len	Fwd Pkt Len	Bwd Pkt Len	Bwd Pkt Len	Bwd Pkt Len	Bwd Pkt Len	Flow Bwts/s	Flow Pkts/s	Flow IAT Me	Flow I
2	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	234830.339	798.403194	1431.42857	2575
3	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	224952.199	764.818356	1494.28571	2445
4	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	211239.788	718.197325	1591.28571	2524
5	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	170322.114	579.080709	1973.57143	4715
6	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	160089.808	544.29174	2099.71429	455
7	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	148024.66	503.271263	2270.85714	4561
8	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	140335.182	477.127691	2395.28571	5986
9	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	131665.827	447.652622	2553	5925
10	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	119885.872	407.601773	2803.85714	6199
11	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	113671.498	386.47343	2957.14286	6209
12	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	355169.811	1207.54717	946.428571	2213
13	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	285454.325	970.520442	1177.57143	2597
14	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	254653.68	865.800866	1320	2590
15	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	237868.985	808.734331	1413.14286	2546
16	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	223350.736	759.373517	1505	2574
17	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	207002.727	703.791678	1623.85714	2554
18	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	269530.355	916.380298	1247.14286	2737
19	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	247606.019	841.839419	1357.57143	2755
20	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	226992.09	771.753811	1480.85714	283
21	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	161396.529	548.734481	2082.71429	4688
22	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	195952.698	666.222518	1715.42857	3922
23	6	2	6	17	2336	17	0	8.5	12.0208153	1448	0	389.333333	623.883216	182799.876	621.50404	1838.85714	3921

Figure 4. 2: a csv file dataset

### 4.4.2. Dataset preprocessing

#### A. Importing libraries and dataset

As a first step, we start importing the needed libraries and modules. We used a different set of libraries; each has a different purpose as presented in the Listings below:

- Libraries needed for preprocessing and visualization the dataset

```
#plot library
import matplotlib.pyplot as plt
from matplotlib import pyplot
from itertools import cycle
from scipy import interp
import seaborn as sb
#dataset preprocessing
from sklearn.preprocessing import MinMaxScaler
from collections import Counter
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import SMOTE
#####
import numpy as np
import pandas as pd
from time import time
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn import model_selection
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
```

*Listing 4. 1: Preprocessing and Plot Libraries*

- Necessary libraries for models' evaluation.

```
#evaluation library
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import mean_absolute_error
from sklearn.preprocessing import StandardScaler
```

*Listing 4. 2: Evaluation Libraries*

- The used libraries in building the ML models.

```
#supervised learning models library
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
```

*Listing 4. 3 : Supervised Model Libraries*

- Keras libraries

```
#keras library
from keras import Sequential
from keras.utils import np_utils
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
#importing activation functions
from keras.layers import LeakyReLU, PReLU, ELU
from keras.layers import Dropout
```

*Listing 4. 4 Keras Libraries*

- Libraries used for the feature selection.

```
#for feature selection
import joblib
import sys
sys.modules['sklearn.externals.joblib'] = joblib
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
```

*Listing 4. 5: Feature Selection Libraries*

## B. Importing dataset

Since the dataset used is in CSV format, we used “read\_csv ()” function of the Pandas library in order to import it, as shown in Listing 4.6.

```
dataset= pd.read_csv("/content/drive/MyDrive/DATASET/PFEdataset/InSDNmerged.csv")
print(dataset)
```

*Listing 4. 6 : Importing Dataset*

- **Separate features and labels**

In our dataset, the features take columns from 1 to 76 which are considered inputs during the phase of model training. the last column"77" contains the Labels that present

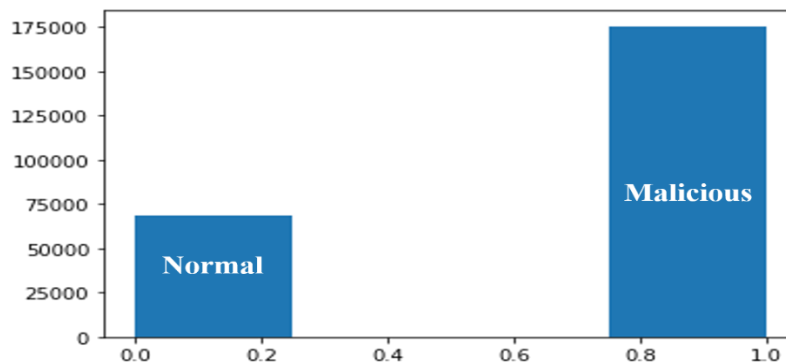
the final output category i.e.; the result of prediction. Listing 4.7 shows the separation step of labels and features.

```
#define label and features
X= dataset.drop('Label', axis=1)
Y= dataset['Label']
```

*Listing 4. 7 : Features and Label Separation*

### Data visualization:

Data visualization is used to help better understand the used dataset. here as shown in Figure 4.3 we visualize the distribution of the classes 0 and 1 which represents respectively normal and malicious traffic.



*Figure 4. 3 : Visualizing Data Classes*

### C. Missing values verification

As mentioned in the chapter 3 section 3.5.3, checking missing values is a very important step that should be done during data preprocessing. To perform this step, we used the "isna()" function, in which the result was false, i.e., there is no missing values in our dataset Listing 4.8.

```
#check missing data
dataset.isna().any().any()
```

False

*Listing 4. 8: Results of missing values check*

## D. Splitting data

Dataset splitting remains an essential step. after loading and preparing the dataset, it shall be split into test and training sets using the 80:20 that we chose after comparing the model's evaluation results with the 70:30 ratio, where:

- 80% is a training set that is used by the model to train on.
- 20% is a testing set, used for evaluating the models for predicting these new data.

`Train_test_split ()` is a Python module that's used for data splitting, as shown in Listing 4.9.

```
#splitting data into train and test sets 80:20
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20)
```

```
print("\n 80:20 ratio split")
print('\nx_train shape:', X_train.shape)
print('x_test shape:', X_test.shape)
print('y_train shape:', Y_train.shape)
print('y_test shape:', Y_test.shape)
```

```
80:20 ratio split

x_train shape: (195185, 76)
x_test shape: (48797, 76)
y_train shape: (195185,)
y_test shape: (48797,)
```

*Listing 4. 9 : Results of data splitting*

## E. Feature standardization

This step aims to standardize the variables of the dataset in a specific range. To apply it we used the `StandardScaler ()` method (See Listing 4.10), and the results of it shown in Figure 4.4.

```
#standardization

Standardisation = preprocessing.StandardScaler()
#80:20 ratio
X_train_stand = Standardisation.fit_transform(X_train)
X_test_stand= Standardisation.fit_transform(X_test)
```

*Listing 4. 10: Feature Standardization*

```

before standardization :
Protocol  Tot Fwd Pkts  Tot Bwd Pkts  TotLen Fwd Pkts  \
      6           2           4           0.0
      0           0           2           0.0
      0           0           2           0.0
      0           0           2           0.0
      0           0           2           0.0
      ...         ...         ...         ...
      6          21          26          1060.0
      0           0           2           0.0
     17           1           3           30.0
      0           0           2           0.0
      0           0           2           0.0

after standardization :
[[ 0.25498343 -0.00346485 -0.02494384 ... -0.10326572 -0.20600000
 -0.18796108]
 [-0.79282787 -0.00443449 -0.03977218 ... -0.10326572 -0.20600000
 -0.18796108]
 [-0.79282787 -0.00443449 -0.03977218 ... -0.10326572 -0.20600000
 -0.18796108]
 ...
 [ 2.17597082 -0.00394967 -0.03235801 ... -0.10326572 -0.20600000
 -0.18796108]
 [-0.79282787 -0.00443449 -0.03977218 ... -0.10326572 -0.20600000
 -0.18796108]
 [-0.79282787 -0.00443449 -0.03977218 ... -0.10326572 -0.20600000
 -0.18796108]

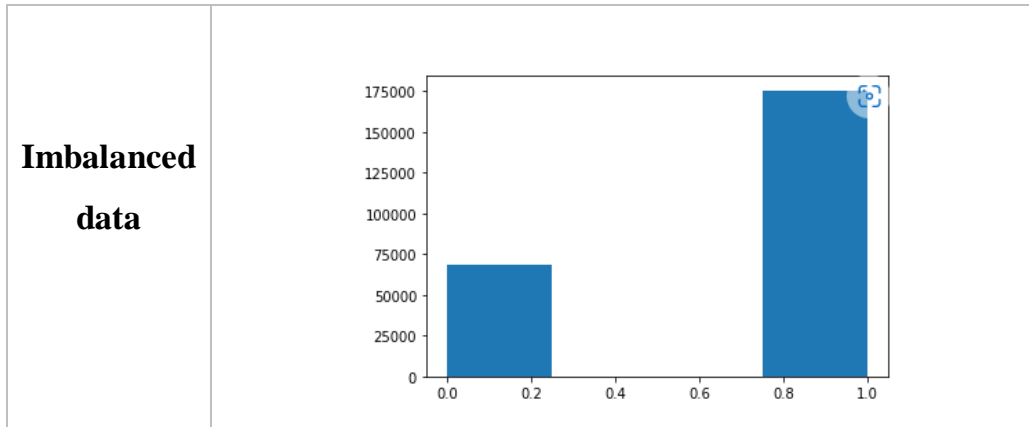
```

Figure 4. 4: Results of feature standardization

## F. Data preparation and pre-processing

Among the techniques cited in the previous chapter to deal with the unbalanced class distribution, we used the SMOTE technique (Synthetic Minority Oversampling Technique-Oversampling). And it gives us better performance in terms of recall and precision. Table 4.2 and 4.3 show the classes distribution before and after oversampling.

<b>Dataset shape</b>	<pre> xtrain set shape: (195185, 76) xtest set shape: (48797, 76) ytrain set shape: (195185,) ytrain set shape: (48797,) </pre>
<b>Class proportion</b>	<pre> number of label classes before balancing : Counter({1: 140319, 0: 54866}) </pre>



*Table 4. 2: Class distribution results and histogram before oversampling*

Dataset shape	<pre>xtrain set shape after balancing : (280638, 76) ytrain set shape after balancing : (280638,)</pre>
Class proportion	<pre>number of label classes after balancing : Counter({1: 140319, 0: 140319})</pre>
Imbalanced data	<p>A histogram with two bars of equal height. Both bars, representing class 0 and class 1, have a height of approximately 140,000. The x-axis is labeled from 0.0 to 1.0 in increments of 0.2. The y-axis is labeled from 0 to 140,000 in increments of 20,000.</p>

*Table 4. 3: class distribution results and histogram after oversampling ( SMOTE)*

## G. Hyperparameters choosing

Choosing a set of optimal hyperparameters remains a problem for learning algorithms. the objective is to find an optimal combination of hyperparameters that would improve and give better performance to the model. several techniques help to tune the hyperparameters, one of which is the GridSearchCV() that we chose in this work, it gives the best collection of hyperparameters among as set of given parameters. this method was applied on our models to determine their best hyperparameters for better performance in term of evaluation metrics. (See APPENDIX A sec 2)

## H. Feature Selection

Using the forward selection method (Listing 4.11), we got a subset of 55 features that would give us the best models performance as a result of comparing several subsets with different number of features. (See APPENDIX B)

```
feature_selector = SFS(RandomForestClassifier(),
    k_features=55,
    forward=True,
    floating=False,
    verbose=2,
    scoring='recall',
    cv=5)

features = feature_selector.fit(x_train, y_train)
```

*Listing 4. 11: Forward selection method*

Table 4.4 represents the obtained features after the feature selection step. Obviously, we got some common features with the related works' features like the Protocol, yet there are some features that wasn't used in the test before as a specific subset like Bwd Blk Rate Avg

No.	Attribute Name	No.	Attribute Name
01	Protocol	29	Fwd PSH Flags
02	Tot Fwd Pkts	30	Bwd PSH Flags
03	Tot Bwd Pkts	31	Fwd URG Flags
04	TotLen Fwd Pkts	32	Bwd URG Flags
05	TotLen Bwd Pkts	33	RST Flag Cnt



06	Fwd Pkt Len Max	34	PSH Flag Cnt
07	Fwd Pkt Len Min	35	ACK Flag Cnt
08	Fwd Pkt Len Mean	36	URG Flag Cnt
09	Fwd Pkt Len Std	37	CWE Flag Count
10	Bwd Pkt Len Max	38	ECE Flag Cnt
11	Bwd Pkt Len Min	39	Down/Up Ratio
12	Bwd Pkt Len Mean	40	Fwd Seg Size Avg
13	Bwd Pkt Len Std	41	Bwd Seg Size Avg
14	Flow Byts/s	42	Fwd Byts/b Avg
15	Flow Pkts/s	43	Fwd Pkts/b Avg
16	Flow IAT Mean	44	Fwd Blk Rate Avg
17	Flow IAT Std	45	Bwd Byts/b Avg
18	Flow IAT Max	46	Bwd Pkts/b Avg
19	Flow IAT Min	47	Bwd Blk Rate Avg
20	Fwd IAT Tot	48	Subflow Fwd Pkts
21	Fwd IAT Mean	49	Subflow Fwd Byts
22	Fwd IAT Std	50	Subflow Bwd Pkts
23	Fwd IAT Max	51	Subflow Bwd Byts
24	Fwd IAT Min	52	Init Fwd Win Byts
25	Bwd IAT Tot	53	Init Bwd Win Byts
26	Bwd IAT Mean	54	Fwd Act Data Pkts
27	Bwd IAT Std	55	Fwd Seg Size Min
28	Bwd IAT Max		

*Table 4.4 : Selected Subset Features*

## 4.5. Model training and evaluation

All the selected models were trained and tested on 2 sets of datasets, a fully featured dataset (brut) that contains all the features, and a specific features subset.

### A. Fully featured version of the dataset

#### ➤ Support vector machine

- Hyperparameters

Table 4.5 represents the used hyperparameters for training and testing the SVM model. (APPENDIX A sec2)

Hyperparameters	C	Gamma	Kernel
Values	10	0.0001	rbf

*Table 4.5 : SVM Hyperparameters*

- **Results**

The Figure 4.5 shows the results of training and evaluating the performance of SVM classifier. (See APPENDX A).

```

*****evaluate the 80:20 ratio*****

-----classification report-----

-->
      precision    recall  f1-score   support

     0       1.00      0.99      0.99     13501
     1       1.00      1.00      1.00     35296

 accuracy          1.00          1.00     48797
 macro avg         1.00      0.99      0.99     48797
weighted avg         1.00      1.00      1.00     48797

-----SVM evaluation metrics-----

--> accuracy : 0.9957989220648811

--> precision score is : 0.9953976903746788

--> Recall score is : 0.9988100634632819

--> f1 score is : 0.9971009573911446

--> training time: 582.5490443706512

```

*Figure 4.5 : SVM Evaluation Results*

- **Confused matrix** is presented in Figure 4.6 below:

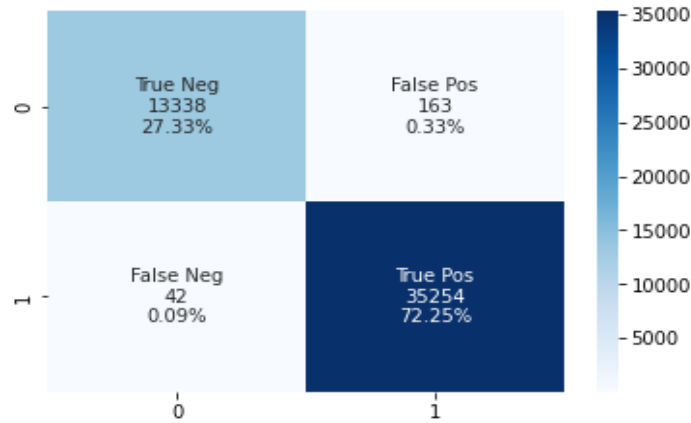


Figure 4. 6: SVM Confusion Matrix

- **Discussion**

According to the confusion matrix, class 1 which represents the malicious traffic was detected as an attack correctly with 72.25%, also 27.33% of the normal traffic class (0) are detected correctly as normal traffic, along with a low error rate, which considered good results.

- **Random forest**

- **Hyperparameters**

Table 4.7 represents the used hyperparameters for training and testing the random forest classifier. (APPENDIX A sec2)

Hyperparameters	Max_ depth	Max_ features	Max_ samples	Min_ samples_split	N_estimator
Values	7	auto	200	2	150

Table 4. 6 : RF Hyperparameters

- **Results and discussion**

The performance evaluation results of the RF classifier are shown in Figures 4.7 and 4.8 where they represent the evaluation metrics and confusion matrix respectively.

```

classification report:
      precision    recall  f1-score   support

     0       0.95      1.00      0.97     13562
     1       1.00      0.98      0.99     35235

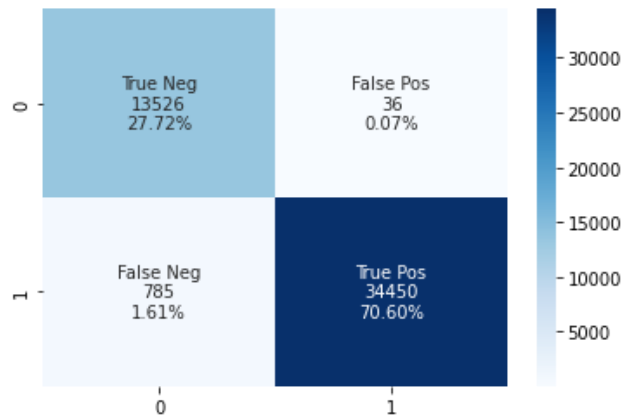
 accuracy          0.98     48797
 macro avg       0.97      0.99      0.98     48797
 weighted avg    0.98      0.98      0.98     48797

-----RF evaluation metrics-----

--> accuracy : 0.983175195196426
--> precision score is : 0.9989560981267761
--> Recall score is : 0.9777210160351922
--> f1-score is : 0.9882244947720199
--> training time: 4.627038478851318

```

*Figure 4.7 : RF Evaluation Results*



*Figure 4.8 : RF Confusion Matrix*

From the confusion matrix, we can see that class 1 was detected correctly with a rate of 70.10%, and class 0 got 27.61%, yet the error rate is a bit high.

➤ ANN

• **Hyperparameters**

Table 4.6 represents the used hyperparameters for training and testing the ANN classifier. (APPENDIX A sec2)

Hyperparameters	Hidden_layers	Max_iter	Activation	Solver
Values	8	150	relu	adam

*Table 4.7 : ANN Hyperparameters*

• **Results and discussion**

The performance evaluation results of the ANN classifier are shown in Figures 4.9 and 4.10 where they represent the evaluation metrics and confusion matrix respectively.

```

***** 80:20 rasion *****

----- Classification report -----

classification report:
      precision    recall  f1-score   support

   0           1.00     1.00     1.00     13501
   1           1.00     1.00     1.00     35296

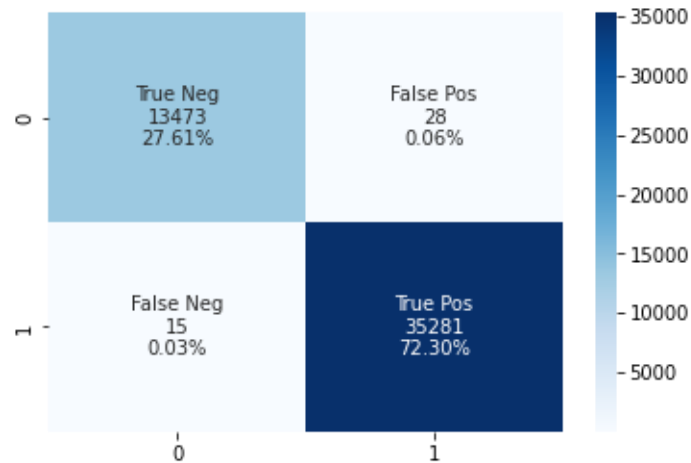
 accuracy          1.00     1.00     1.00     48797
 macro avg         1.00     1.00     1.00     48797
weighted avg         1.00     1.00     1.00     48797

----- ANN evaluation metrics -----

--> accuracy : 0.99911879828678
--> precision score is : 0.9992070010478915
--> Recall score is : 0.9995750226654578
--> f1-score is : 0.999390977976064
--> training time: 81.13095498085022

```

*Figure 4.9 : ANN Evaluation Results*



*Figure 4.10 : ANN Confusion Matrix*

According to the confusion matrix above, class 1 which represents the malicious traffic was detected as an attack correctly with 72.30%, also 27.61% of the normal traffic class (0) are detected correctly as normal traffic, along with a low error rate, which considered good results.

### ➤ Models Comparison

Table 4.7 shows the performance of the different classifiers using the fully-featured version of the dataset. we used the most commonly used performance metrics in intrusion detection systems, besides the accuracy score as well, though we won't focus on it that much compared to the others because it doesn't yield precise comparisons.

Clearly, the overall score metrics are good enough in detecting malicious traffic for both split ratios. yet there is a small variation between the classifiers. the recall and F1-score metrics for the SVM, RF and ANN algorithms of the 70:30 split are lower than the 80:20 split classifiers, which seem to have the best results of all. Yet in the same ratio, the classifiers are varied, and we can clearly recognize that SVM and RF have lower metric scores than the ANN. Furthermore, the performance and training time of SVM is a bit slower, unlike the other classifiers that are reasonable. Therefore, ANN is considered the best model in this case for detecting and predicting malicious attacks.

Split ratio	ML models	Evaluation Metrics				
		Accuracy	Precision	Recall	F1-score	Training time
80:20	SVM	0.9957	0.9953	0.9988	0.9971	582.549
	RF	0.9831	0.9989	0.9777	0.9882	4.6270
	ANN	0.9991	0.9992	0.9995	0.9993	81.130
70:30	SVM	0.9954	0.9950	0.9986	0.9968	440.22
	RF	0.9784	0.9998	0.9703	0.9848	4.0618
	ANN	0.9987	0.9991	0.9991	0.9991	79.0199

Table 4.8 : Performance metrics for Fully-Featured dataset models (70:30 VS 80:20)

## B. Specific features version of the dataset

### ➤ Support vector machine

The figure 4.11 shows the results of training and evaluating the performance of SVM model with a featured of the dataset (see APPENDX A).

```

-----classification report-----
-->
              precision    recall  f1-score   support

     0           1.00      0.99      0.99     13701
     1           0.99      1.00      1.00     35096

 accuracy          1.00      1.00      1.00     48797
 macro avg          1.00      0.99      0.99     48797
 weighted avg       1.00      1.00      1.00     48797

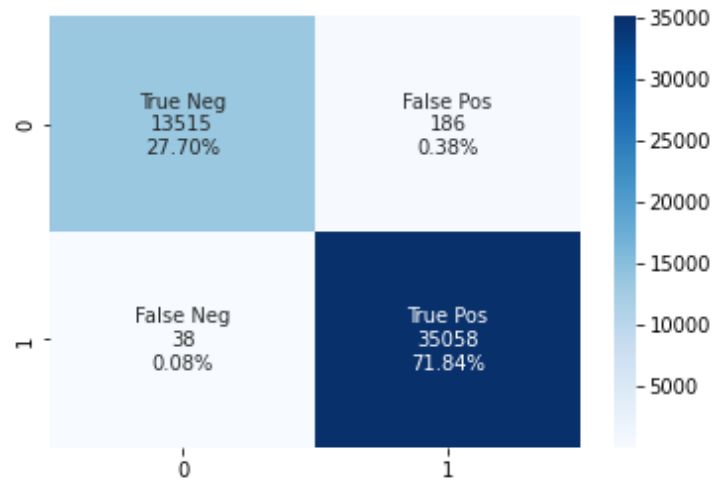
-----SVM evaluation metrics-----

--> accuracy : 0.9954095538660164
--> precision score is : 0.994722505958461
--> Recall score is : 0.9989172555276955
--> f1 score is : 0.9968154677281774
--> training time: 89.56371474266052

```

Figure 4.11: Spec\_SVM Evaluation Results

- **Confused matrix** is presented in figure 4.12 below:



*Figure 4. 12: Spec\_SVM Confusion Matrix*

- **Discussion**

**Class 0:** 13515 instances of this class got classified as normal traffic, and it is normal traffic scoring a 27.70% with a very low error rate.

**Class 1:** the rate of malicious attacks that were detected and classified correctly as attack, according to the confusion matrix, is 71.84%, with a low error rate.

### ➤ **Random Forest**

Figures 4.13 and 4.14 respectively represent the evaluation metrics and confusion matrix of the performance evaluation of the classifier RF on a specific featured version of the dataset.



```

-----classification report-----
-->
              precision    recall  f1-score   support

     0       0.94         0.99         0.97     13701
     1       1.00         0.98         0.99     35096

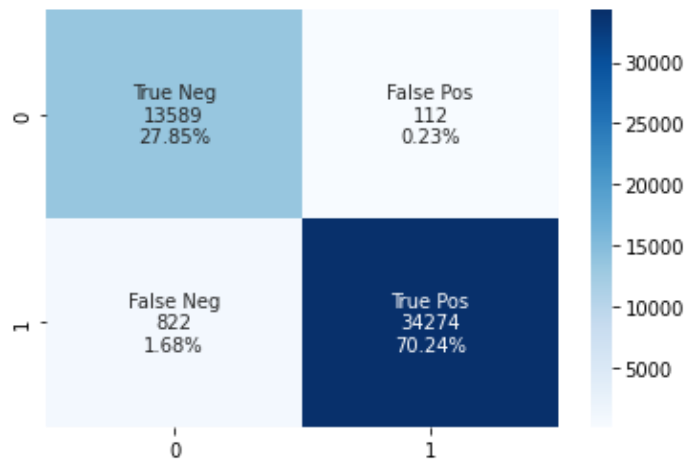
 accuracy         0.98         0.98         0.98     48797
 macro avg        0.97         0.98         0.98     48797
 weighted avg     0.98         0.98         0.98     48797

-----RF evaluation metrics-----

--> accuracy : 0.9808594790663361
--> precision score is : 0.9967428604664689
--> Recall score is : 0.9765785274675177
--> f1-score is : 0.9865576696122738
--> training time: 4.182465314865112

```

*Figure 4. 13: Spec\_RF Evaluation Results*



*Figure 4. 14: Spec\_RF Confusion Matrix*

- **Discussion**

**Class 0:** 13589 instances of this class got classified as a normal traffic, and it is normal traffic, with rate of 27.85% and a low error rate.

**Class 1:** the rate of malicious attacks that were detected and classified correctly as attack, according to the confusion matrix, is 70.24%, with a low error rate.

### ➤ ANN

The coming Figures 4.15 and 4.16 show the evaluation metrics and confusion matrix of the performance evaluation of the classifier RF on a specific featured version of the dataset, respectively.

```

classification report:
              precision    recall  f1-score   support

     0           1.00        1.00        1.00     13701
     1           1.00        1.00        1.00     35096

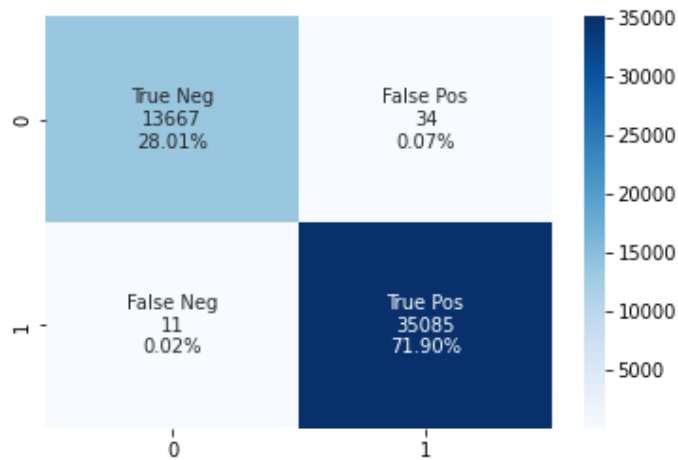
 accuracy          1.00          1.00          1.00     48797
 macro avg          1.00          1.00          1.00     48797
 weighted avg       1.00          1.00          1.00     48797

-----ANN evaluation metrics-----

--> accuracy : 0.9990778121605837
--> precision score is : 0.9990318630940517
--> Recall score is : 0.9996865739685434
--> f1-score is : 0.9993591113010041
--> training time: 99.66347360610962

```

*Figure 4. 15: Spec\_ANN Evaluation Results*



*Figure 4. 16: Spec\_ANN Confusion Matrix*

- **Discussion:**

**Class 0:** the 13667 instances of this class got classified as a normal traffic, and it is normal traffic. Got a rate of 28.01% and a very low error rate.

**Class 1:** the rate of malicious attacks that were detected and classified correctly as attack, according to the confusion matrix, is 71.90% representing 35085 instances, with a very low error rate.

➤ **Specific features models Comparison**

In this section, the models are compared to each other according to the chosen evaluation metrics. As shown in Table 4.8, the RF classifier tends to have the shortest training time, but the lowest metrics scores compared to the other classifiers as we can see in Figure 4.17, clearly ANN is the best model with the best scores in Recall, F1 score, Precision and even the Accuracy.

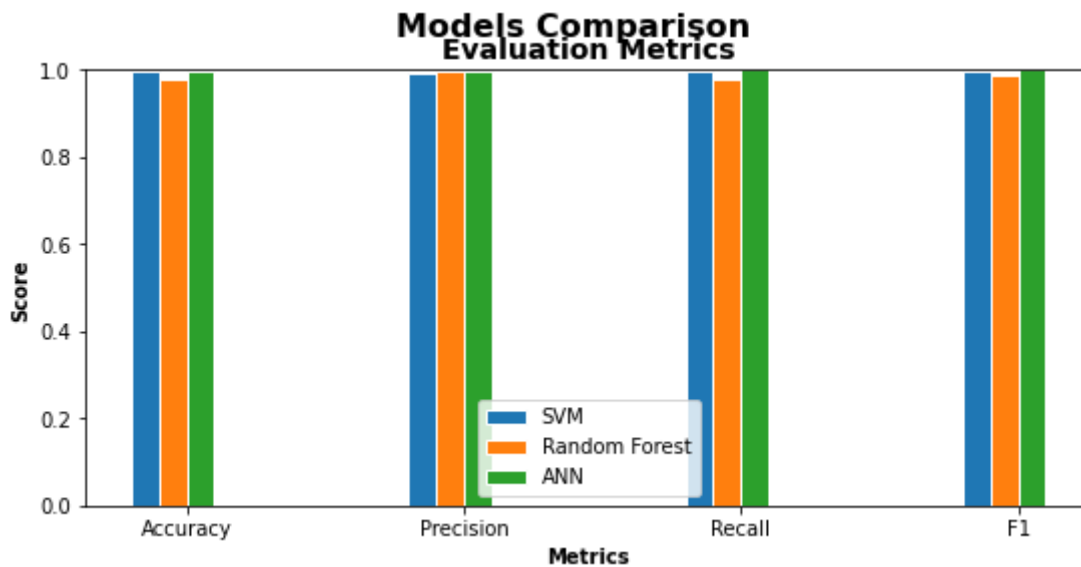


Figure 4. 17: Specific features models' evaluation metrics comparison

ML models	Evaluation Metrics				
	Accuracy	Precision	Recall	F1-score	Training time
Spec_SVM	0.9954	0.9947	0.9989	0.9968	89.563
Spec_RF	0.9808	0.9967	0.9765	0.9865	4.1824
Spec_ANN	0.9990	0.9990	0.9996	0.9993	99.663

Table 4. 9: Comparison between specific models' performance metrics

➤ **A comparison between the fully-featured dataset models and the specific featured dataset models**

After testing several machine learning algorithms with different dataset subsets of different features, we noticed that all the algorithms give good results, but when it comes to the optimal one that detects the DoS/DDOS attacks, the ANN classifier that was tested with the specific features dataset, since Recall represents the true positive rate (i.e., the rate of the number of attacks that was detected as attacks correctly), is considered the optimal, for its Recall score along with the other metrics scores that were high compared to other classifiers( table 4.9).

	<b>SVM</b>	<b>Spec_SVM</b>	<b>RF</b>	<b>Spec_RF</b>	<b>ANN</b>	<b>Spec_ANN</b>
<b>Accuracy</b>	0.9957	0.9954	0.9831	0.9808	0.9991	0.9990
<b>Precision</b>	0.9953	0.9947	0.9989	0.9967	0.9992	0.9990
<b>Recall</b>	0.9988	0.9989	0.9777	0.9765	0.9995	<b>0.9996</b>
<b>F1-score</b>	0.9971	0.9968	0.9882	0.9865	0.9993	0.9993
<b>Training time</b>	582.549	89.563	4.6270	4.1824	81.130	99.663
						Best ML model

*Table 4. 10: Comparison between all the models*

#### 4.6. Comparison with related work models

In order to compare our model with the related works cited in chapter 2, we used the Recall accuracy rate along with precision and f1 score. Table 4.8 summarizes the results of our model and the other three works in which we notice that our model gives way better performances compared to them.

<b>Work</b>	<b>Specific featured dataset</b>	<b>Recall</b>	<b>Precision</b>	<b>F1_score</b>
[12]	48 specific futures	Table 2.2	Table 2.2	Table 2.2
[18]	48 specific features	0.9980	0.9980	0.9980
[19]	36/18/8 specific features	Table 2.4	Table 2.4	Table 2.4
<b>Our model</b>	55 specific features	0.9996	0.9990	0.9993

*Table 4. 11: Comparison with related works models*

## 4.7. Conclusion

In this chapter, we dealt with the implementation part in a detailed way, where we started by describing the tools, the environments we worked on, and the used dataset. Next, we presented the preprocessing steps we went through in order to build and evaluate our models and explained all the experimentations and analyzed the models' results to make a comparative study with the related works.

# General Conclusion

The implementation of machine learning-based models for intrusion detection has been widely used, and always caught the researchers' interest, especially in the Software-Defined Networks which are considered a challenge to address.

Through this dissertation, we have proposed three machine learning models (ANN, SVM and RF) for classifying and detecting DoS/DDoS attacks in SDN using the newly generated InSDN dataset for training and testing after going through some preprocessing steps and implementing the forward selection method to select the best features that achieve higher performance after discussing some related works that dealt with the same problem as ours.

The obtained results after training and testing the models are quite satisfying. After comparing the results of each model, we got the ANN model that was evaluated on the specific features subset as the best performance in terms of the evaluation metrics (Recall, F1-score, Precision).

Despite the quality of the results obtained, the feature selection method and models' hyperparameters need to be improved in order to reach the optimal result.

Therefore, as future work, we aspire to develop a genetic algorithm-based model to get the optimal features and optimal parameters in terms of F1-score, Recall, Precision... of a dataset for detecting several types of attacks instead of just DoS/DDoS attacks in a realistic environment.

# BIBLIOGRAPHY

- [1] Benzekki, Kamal; El Fergougui, A., Elbelrhiti Elalaoui, Abdelbaki (2016). Software-defined networking (SDN): a survey. *Security and Communication Networks*, 9(18), 5803–5833. doi:10.1002/sec.1737
- [2] Sezer, Sakir; Scott-Hayward, Sandra; Chouhan, PushPinder; Fraser, Barbara; Lake, David; Finnegan, Jim; Viljoen, Niel; Miller, Marc; Rao, Navneet (2013). Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine*, 51(7), 36–43. doi:10.1109/MCOM.2013.6553676
- [3] Feamster, Nick; Rexford, Jennifer; Zegura, Ellen (2014). The road to SDN. *ACM SIGCOMM Computer Communication Review*, 44(2), 87–98. doi:10.1145/2602204.2602219
- [4] Journals | Telsoc. (2022). Retrieved 10 March 2022, from <https://telsoc.org/journal/ajtdev3-n4/a28>.
- [5] Karakus, M., & Durresi, A. (2017). A survey: Control plane scalability issues and approaches in Software-Defined Networking (SDN). *Computer Networks*, 112, 279–293. doi:10.1016/j.comnet.2016.11.017
- [6] Rao, S.K. (2014). SDN AND ITS USE-CASES-NV AND NFV A State-of-the-Art Survey.
- [7] NFV, Network Virtualization, OpenFlow and SDN Use Cases, [Online]. Available: <http://www.sdncentral.com/sdn-use-cases>. [Accessed 03/15/2022]
- [8] Shu, Zhaogang; Wan, Jiafu; Li, Di; Lin, Jiayang; Vasilakos, Athanasios V.; Imran, Muhammad (2016). Security in Software-Defined Networking: Threats and Countermeasures. *Mobile Networks and Applications*, 21(5), 764–776. doi:10.1007/s11036-016-0676-x
- [9] K. Cabaj, J. Wytrębowicz S. Kukliński P. Radziszewski K. Truong Dinh “SDN Architecture Impact on Network Security” Position papers of the 2014 Federated Conference on Computer Science and Information Systems pp. 143–148 DOI: 10.15439/2014F473 ACSIS, Vol.3
- [10] Ahmad, A., Harjula, E., Ylianttila, M., & Ahmad, I. (2020). Evaluation of Machine Learning Techniques for Security in SDN. 2020 IEEE Globecom Workshops (GC Wkshps). doi:10.1109/gcwkshps50303.2020.93
- [11] I. Ahmad; S. Namal; M. Ylianttila; A. Gurtov (2015). Security in Software Defined Networks: A Survey. doi:10.1109/COMST.2015.2474118

- [12] MAHMOUD SAID ELSAYED, NHIEN-AN LE-KHAC, (Member, IEEE), AND ANCA D. JURCUT (2020). InSDN: A Novel SDN Intrusion Dataset. School of Computer Science, University College Dublin, Dublin 4, Ireland Corresponding.
- [13] Zhang, Kai; Qiu, Xiaofeng (2018). [IEEE 2018 IEEE International Conference on Consumer Electronics (ICCE) - Las Vegas, NV, USA (2018.1.12-2018.1.14)] 2018 IEEE International Conference on Consumer Electronics (ICCE) - CMD: A convincing mechanism for MITM detection in SDN, 1–6. doi:10.1109/ICCE.2018.8326334
- [14] Nisar, Kasif; Welch, Ian; Hassan, Rosilah; Sodhro, Ali Hassan; Pirbhulal, Sandeep (2020). A Survey on the Architecture, Application, and Security of Software Defined Networking. Internet of Things, 100289–. doi:10.1016/j.iot.2020.100289
- [15] Scott-Hayward, Sandra; Natarajan, Sriram; Sezer, Sakir (2015). A Survey of Security in Software Defined Networks. IEEE Communications Surveys & Tutorials, 1-1. doi:10.1109/COMST.2015.2453114
- [16] Shin S, Porras P, Yegneswaran V, Gu G (2013) A framework for integrating security services into software-defined networks. In: Proceedings of the 2013 Open Networking Summit (Research Track poster paper)
- [17] International Journal of Scientific & Engineering Research, Volume 2, Issue 1, January-2011 1 ISSN 2229-5518 IJSER © 2010 <http://www.ijser.org> Importance of Intrusion Detection System (IDS) Asmaa Shaker Ashoor (Department computer science, Pune University) Prof. Sharad Gore (Head department statistic, Pune University)
- [18] Kemmerer, R.A.; Vigna, G. (2002). Intrusion detection: a brief history and overview. Computer, 35(4), suppl27–suppl30. doi:10.1109/MC.2002.1012428
- [19] Prasad, K.M., Reddy, D.A., & Rao, K.V. (2014). DoS and DDoS Attacks: Defense, Detection and Traceback Mechanisms - A Survey. Global journal of computer science and technology, 14.
- [20] Nagamalai, Dhinakaran; Renault, Eric; Dhanuskodi, Murugan (2011). [Communications in Computer and Information Science] Advances in Parallel Distributed Computing Volume 203. A Recent Survey on DDoS Attacks and Defense Mechanisms, 10.1007/978-3-642-24037-9(Chapter 57), 570–580. doi:10.1007/978-3-642-24037-9\_57
- [21] Kamboj, Priyanka; Trivedi, Munesh Chandra; Yadav, Virendra Kumar; Singh, Vikash Kumar (2017). [IEEE 2017 4th IEEE Uttar Pradesh Section International conference on Electrical,



Computer and Electronics (UPCON) - Mathura, India (2017.10.26-2017.10.28)] 2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)-Detection techniques of DDoS attacks: A survey., (), 675–679. doi:10.1109/UPCON.2017.8251130

[22] Mitchell TM. (1997). Machine learning. New York: McGraw-Hill.

[23] Gramajo, M.G., Ballejos, L.C., & Ale, M.A. (2020). Seizing Requirements Engineering Issues through Supervised Learning Techniques. *IEEE Latin America Transactions*, 18, 1164-1184.

[24] Praveen Kumar, D.; Amgoth, Tarachand; Annavarapu, Chandra Sekhara Rao (2019). Machine learning algorithms for wireless sensor networks: A survey. *Information Fusion*, 49(), 1–25. doi:10.1016/j.inffus.

[25] Sultana, N., Chilamkurti, N.K., Peng, W., & Alhadad, R. (2019). Survey on SDN based network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and Applications*, 12, 493-501.

[26] Uddin, Mohammad Shorif; Bansal, Jagdish Chand (2020). [Algorithms for Intelligent Systems] Proceedings of International Joint Conference on Computational Intelligence (IJCCI 2018). Leveraging Machine Learning Approach to Setup Software-Defined Network (SDN) Controller Rules During DDoS Attack., 10.1007/978-981-13-7564-4(Chapter5), 49–60. doi:10.1007/978-981-13-7564-4\_5

[27] Cortes, C., & Vapnik, V.N. (2004). Support-Vector Networks. *Machine Learning*, 20, 273-297.

[28] Yuan, R., Li, Z., Guan, X., & Xu, L. (2010). An SVM-based machine learning method for accurate internet traffic classification. *Information Systems Frontiers*, 12, 149-156.

[29] Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., & Lopez, A. (2020). A comprehensive survey on support vector machine classification: applications, challenges and trends. *Neurocomputing*. doi: 10.1016/j.neucom.2019.10.118

[30] Pisner, D. A., & Schnyer, D. M. (2020). Support vector machine. *Machine Learning*, 101–121. doi:10.1016/b978-0-12-815739-8.00006-7

[31] Takeuchi, Yuki; Sakai, Kazuya; Fukumoto, Satoshi (2018). [ACM Press the 47th International Conference - Eugene, OR, USA (2018.08.13-2018.08.16)] Proceedings of the 47th

International Conference on Parallel Processing Companion - ICPP '18 - Detecting Ransomware using Support Vector Machines., (), 1–6. doi:10.1145/3229710.3229726

[32] Breiman, L. (2004). Random Forests. *Machine Learning*, 45, 5-32.

[33] Cutler, D. Richard; Edwards, Thomas C.; Beard, Karen H.; Cutler, Adele; Hess, Kyle T.; Gibson, Jacob; Lawler, Joshua J. (2007). RANDOM FORESTS FOR CLASSIFICATION IN ECOLOGY., 88(11), 2783–2792.

[34] Tian, Shaohong; Zhang, Xianfeng; Tian, Jie; Sun, Quan (2016). Random Forest Classification of Wetland Landcovers from Multi-Sensor Data in the Arid Region of Xinjiang, China. *Remote Sensing*, 8(11), 954–.

[35] V.F. Rodriguez-Galiano; M. Chica-Olmo; F. Abarca-Hernandez; P.M. Atkinson; C. Jeganathan (2012). Random Forest classification of Mediterranean land cover using multi-seasonal imagery and multi-seasonal texture., 121(none), 0–107

[36] Gislason, P.O.; Benediktsson, J.A.; Sveinsson, J.R. (2004). [IEEE IEEE International IEEE International IEEE International Geoscience and Remote Sensing Symposium, 2004. IGARSS '04. Proceedings. 2004 - Anchorage, AK, USA (20-24 Sept. 2004)] IEEE International IEEE International IEEE International Geoscience and Remote Sensing Symposium, 2004. IGARSS '04. Proceedings. 2004 - Random forest classification of multisource remote sensing and geographic data. , 2(0), 1049–1052.]

[37] David Reby; Sovan Lek; Ioannis Dimopoulos; Jean Joachim; Jacques Lauga; Stéphane Aulagnier (1997). Artificial neural networks as a classification method in the behavioural sciences, 40(1), 35–43

[38] Sadhu, T. (2022). Machine Learning: Introduction to the Artificial Neural Network - Durofy - Business, Technology, Entertainment and Lifestyle Magazine. Retrieved 24 May 2022, from <https://durofy.com/machine-learning-introduction-to-the-artificial-neural-network>.

[39] Belaala, A. (2021). Big Data analytics using Artificial Intelligence techniques in medical PHM (PH.D). Mohamed Khider University - Biskra.

[40] Pradip Dhal; Chandrashekhara Azad; (2021). A comprehensive survey on feature selection in the various fields of machine learning . *Applied Intelligence*. doi:10.1007/s10489-021-02550-9.

[41] M. Di Mauro; G. Galatro; G. Fortino; A. Liotta; (2021). Supervised feature selection techniques in network intrusion detection: A critical review. *Engineering Applications of Artificial Intelligence*. doi: 10.1016/j.engappai.2021.104216

- [42] Krishnan, P., Duttagupta, S., & Achuthan, K. (2019). VARMAN: Multi-plane security framework for software defined networks. *Comput. Commun.*, 148, 215-239.
- [43] Omer Elsier Tayfour; Muhammad Nadzir Marsono; (2021). Collaborative detection and mitigation of DDoS in software-defined networks. *The Journal of Supercomputing*, (), – . doi:10.1007/s11227-021-03782-9
- [44] van Staden, J., & Brown, D. (2022). An Evaluation of Machine Learning Methods for Classifying Bot Traffic in Software Defined Networks. Grahamstown, South Africa: National Research Foundation (120654).
- [45] Data Scaling for Machine Learning—The Essential Guide. (2022). Retrieved 6 April 2022, from <https://betterdatascience.com/data-scaling-for-machine-learning/>
- [46] Kulkarni, A., Chong, D., & Batarseh, F.A. (2021). Foundations of data imbalance and solutions for a data democracy. *ArXiv*, *abs/2108.00071*.
- [47] Hossin, M. and Sulaiman, M.N. (2015) *A Review On Evaluation Metrics For Data Classification Evaluations*. *International Journal of Data Mining & Knowledge Management Process*, 5 (2). pp. 1-11. ISSN 2230 – 9608
- [48] What is Python? Executive Summary. (2022). Retrieved 15 June 2022, from <https://www.python.org/doc/essays/blurb/>

## APPENDIX A: Source Code

### 1. Data Preprocessing

#### 1.1. Standardization

```
Standardisation = preprocessing.StandardScaler()

X_train_stand = Standardisation.fit_transform(X_train)
X_test_stand= Standardisation.fit_transform(X_test)
```

#### 1.2. Data balancing

```
smote = SMOTE()

X_train_smote, Y_train_smote = smote.fit_resample(X_train_stand.astype('float'), Y_train)
```

### 2. Hyperparameters Tuning

In order to optimize the performance our models, avoid the overfitting problems, we tuned the hyperparameters of each model. For that, we used the GridSearchCV function to determine the optimal values for its simplicity in implementation.

#### 2.1. RF

For the random forest classifier, we specify a group of values to choose the best ones, we focused on:

- `n_estimators`: the number of trees in the forest.
- `max_depth`: represents the depth of each tree in the forest.
- `min_samples_split`: minimum number of samples required to split an internal node.
- `max_features`: number of features to consider when looking for the best split.

```
forest = RandomForestClassifier()
params = {
    'n_estimators': [100, 150, 200],
    'max_features': ['auto', 'log2'],
    'max_depth' : [4, 5, 6, 7],
    'min_samples_split' : [2, 3, 8],
    'max_samples': [100, 150, 200]
```

```

}
CV_rfc = GridSearchCV(estimator=forest, param_grid=params, cv= 5
)
CV_rfc.fit(x_train, y_train)
CV_rfc.best_params_

```

## Result

```

{'max_depth': 7,
 'max_features': 'auto',
 'max_samples': 200,
 'min_samples_split': 2,
 'n_estimators': 150}

```

## 2.2. SVM

For the SVM hyperparameters tuning, we focused on the most effective, the kernel, the C parameters that permits to regulate how soft the margin can be and the Gamma in which specifies how far the influence of a single training example can reach, giving each of them a group of values to select the optimal for a better performance.

```

params = {
    'C': [0.1, 1, 10, 100, 1000],
    'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
    'kernel': ['rbf', 'linear']
}

clf = GridSearchCV(
    estimator=SVC(),
    param_grid=params,
    cv=5,
    n_jobs=5,
    verbose=1
)

```

## Result

```

— {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}

```

## 2.3. ANN

Concerning the ANN classifier, we focused on the most essential parameters that affect on the classifier performance that are: the number of hidden layer and nodes, the activation function for the hidden layer, number of epochs to use and the solver algorithm for optimizing the weight in which we fixed on 'adam' algorithm because it is the most known and effective.

```
ann = MLPClassifier()
params = {
    'hidden_layer_sizes': [(2,), (6,), (8,)],
    'max_iter': [100, 150, 300],
    'activation': ['tanh', 'relu'],
    'solver': ['adam']
}
CV_ann = GridSearchCV(estimator=ann, param_grid=params, cv= 5, n
_jobs=-1)
CV_ann.fit(x_train_stand, y_train)
CV_ann.best_params_
```

## Results

```
{'activation': 'relu',
 'hidden_layer_sizes': (8,),
 'max_iter': 150,
 'solver': 'adam'}
```

## 3. Model Training

### 3.1. SVM

```
SVM = SVC(C=10, gamma=0.0001, kernel='rbf')

start=time()
SVM.fit(X_train_undersampled, Y_train_undersampled)
end=time()
```

### 3.2. RF

```
RFclassifier = RandomForestClassifier(max_depth= 7,max_features
='auto', max_samples=200,min_samples_split = 2,n_estimators=150)
```

```

start=time()
RFclassifier.fit(X_train_smote, Y_train_smote)
end=time()

```

### 3.3. ANN

```

ANNclassifier = MLPClassifier(hidden_layer_sizes=(8,),max_iter =
    150,activation = 'relu',solver = 'adam')
start=time()
ANNclassifier.fit(X_train_smote, Y_train_smote)
end=time()

```

## 4. Performance evaluation visualization script

### 4.1. Confusion Matrix

```

print("\n\n----- Confusion matrix -----
-----\n\n")
cmann=confusion_matrix(Y_test,ANNPred)
group_names = ['True Neg', 'False Pos', 'False Neg', 'True Pos']
group_counts = ["{0:0.0f}".format(value) for value in cmann.flat
ten()]
group_percentages = ["{0:.2%}".format(value) for value in cmann.
flatten()/np.sum(cmann)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in zip(group_names,
group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
heatmap=sb.heatmap(cmann, annot=labels, fmt='', cmap='Blues')
print(heatmap)

```

### 4.2. Evaluation metric comparison bars

```

fig, (ax1) = plt.subplots(1)
fig.suptitle('Models Comparison', fontsize=16, fontweight='bo
ld')
fig.set_figheight(4)
fig.set_figwidth(9)
fig.set_facecolor('white')
## set bar size
barWidth = 0.1
svm_score = [accuracy_score(Y_test,SVMPred), metrics.precisio
n_score(Y_test,SVMPred), metrics.recall_score(Y_test,SVMPred)
,metrics.f1_score(Y_test,SVMPred)]

```

```

rf_score = [accuracy_score(Y_test, RFPred), metrics.precision
_score(Y_test, RFPred), metrics.recall_score(Y_test, RFPred),
metrics.f1_score(Y_test, RFPred)]
ann_score = [accuracy_score(Y_test,ANNPred), metrics.precisio
n_score(Y_test,ANNPred), metrics.recall_score(Y_test,ANNPred)
,metrics.f1_score(Y_test,ANNPred)]
## Set position of bar on X axis
r1 = np.arange(len(svm_score))
r2 = [x + barWidth for x in r1]
r3 = [x + barWidth for x in r2]
r4 = [x + barWidth for x in r3]
## Make the plot
ax1.bar(r1, svm_score, width=barWidth, edgecolor='white', lab
el='SVM')
ax1.bar(r2, rf_score, width=barWidth, edgecolor='white', labe
l='Random Forest')
ax1.bar(r3, ann_score, width=barWidth, edgecolor='white', lab
el='ANN')
## Configure x and y axis
ax1.set_xlabel('Metrics', fontweight='bold')
labels = ['Accuracy', 'Precision', 'Recall', 'F1']
ax1.set_xticks([r + (barWidth * 1.5) for r in range(len(svm_s
core))], )
ax1.set_xticklabels(labels)
ax1.set_ylabel('Score', fontweight='bold')
ax1.set_ylim(0, 1)
## Create legend & title
ax1.set_title('Evaluation Metrics', fontsize=14, fontweight='
bold')
ax1.legend()

```



## APPENDIX B: Additional results of feature selection tests

The feature selection process went through several test where we tested and trained our models on several subsets each with a specific number of features, then made a comparison to define and precise the best subset that give the best performance results.

### 1. Subset of 28 features

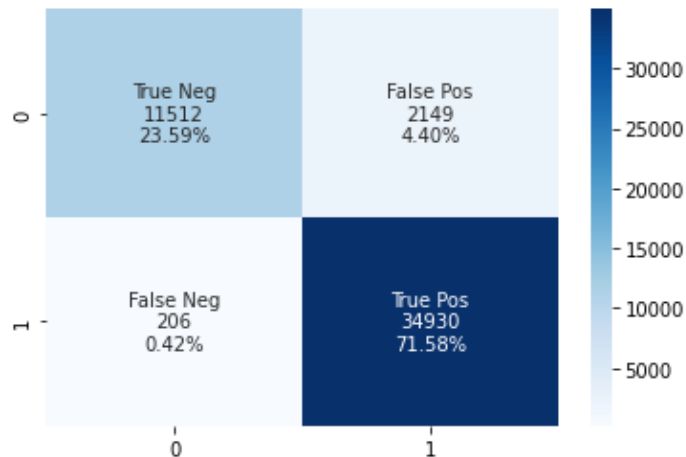
#### 1.1. SVM evaluation results

```
-----classification report-----
-->
      precision    recall  f1-score   support

     0       0.98      0.84      0.91     13661
     1       0.94      0.99      0.97     35136

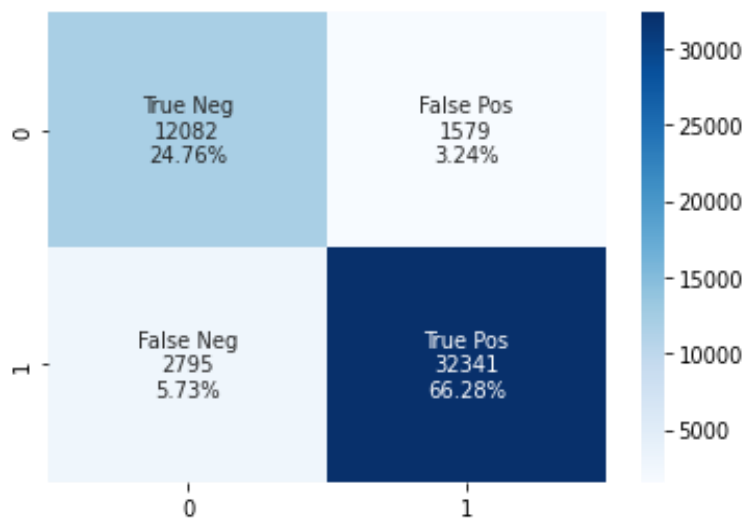
 accuracy: 0.95
macro avg: 0.96      0.92      0.94     48797
weighted avg: 0.95      0.95      0.95     48797

-----SVM evaluation metrics-----
--> accuracy : 0.9517388364038772
--> precision score is : 0.9420426656598074
--> Recall score is : 0.9941370673952641
--> f1 score is : 0.9673890465969675
--> training time: 405.40307784080505
```



## 1.2. RF evaluation results

```
-----classification report-----  
  
-->  
      precision    recall  f1-score   support  
  
   0       0.81      0.88      0.85     13661  
   1       0.95      0.92      0.94     35136  
  
 accuracy          0.91     48797  
 macro avg       0.88     0.90     0.89     48797  
weighted avg       0.91     0.91     0.91     48797  
  
-----RF evaluation metrics-----  
  
--> accuracy : 0.91036334200873  
--> precision score is : 0.9534492924528302  
--> Recall score is : 0.9204519581056466  
--> f1-score is : 0.9366601019462465  
--> training time: 4.785613298416138
```



### 1.3. ANN evaluation results

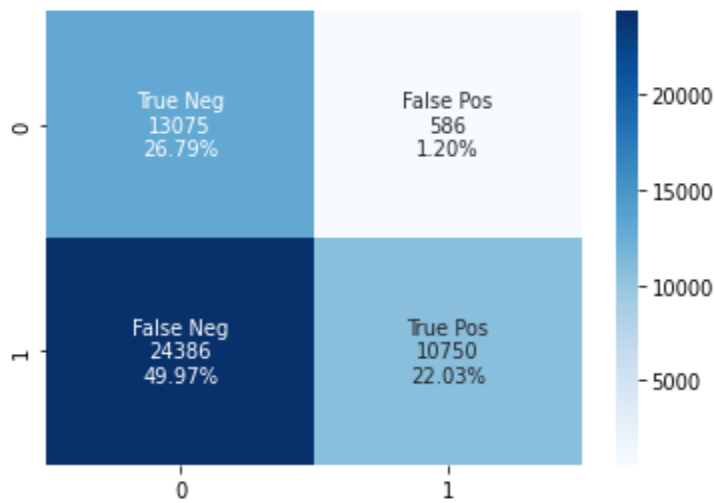
```
classification report:
      precision    recall  f1-score   support

     0       0.35      0.96      0.51    13661
     1       0.95      0.31      0.46    35136

 accuracy          0.49    48797
 macro avg       0.65    0.63    0.49    48797
 weighted avg    0.78    0.49    0.48    48797
```

-----ANN evaluation metrics-----

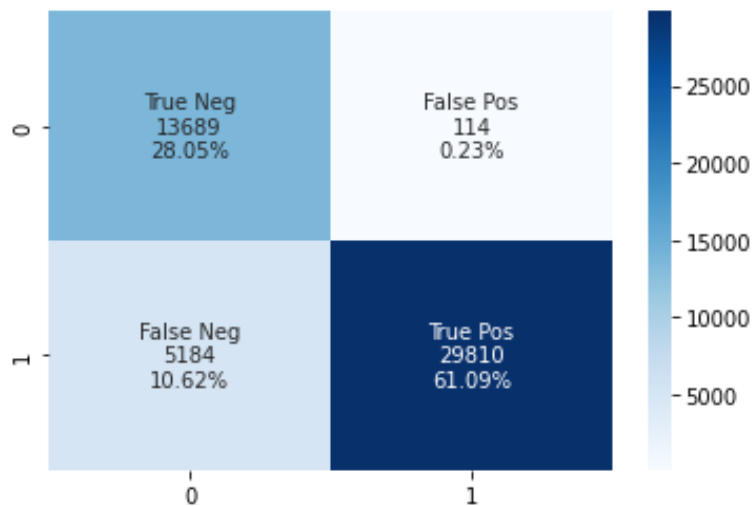
```
--> accuracy : 0.488247228313216
--> precision score is : 0.9483062808750882
--> Recall score is : 0.3059540072859745
--> f1-score is : 0.46264417283525566
--> training time: 136.28979086875916
```



## 2. Subset of 33 features

### 2.1. SVM evaluation results

```
-----classification report-----  
  
-->  
  
              precision    recall  f1-score   support  
  
     0       0.73       0.99       0.84     13803  
     1       1.00       0.85       0.92     34994  
  
 accuracy          0.89     48797  
 macro avg       0.86     0.92     0.88     48797  
 weighted avg    0.92     0.89     0.90     48797  
  
-----SVM evaluation metrics-----  
  
--> accuracy : 0.8914277517060475  
--> precision score is : 0.9961903488838391  
--> Recall score is : 0.8518603189118135  
--> f1 score is : 0.9183893527218953  
--> training time: 300.013774394989
```



## 2.2. RF evaluation results

```
-----classification report-----

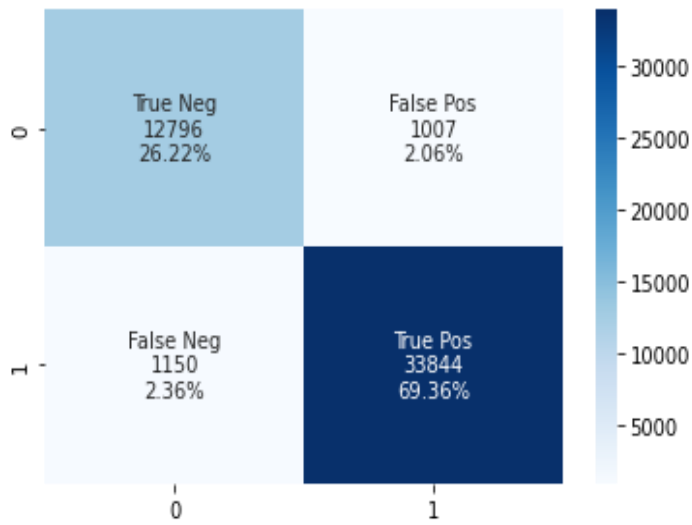
-->
      precision    recall  f1-score   support

     0       0.92     0.93     0.92     13803
     1       0.97     0.97     0.97     34994

 accuracy      0.96     48797
 macro avg     0.94     0.95     0.95     48797
weighted avg     0.96     0.96     0.96     48797

-----RF evaluation metrics-----

--> accuracy : 0.9557964628973092
--> precision score is : 0.9711055636854036
--> Recall score is : 0.9671372235240326
--> f1-score is : 0.9691173312334456
--> training time: 4.559622526168823
```



### 2.3. ANN evaluation results

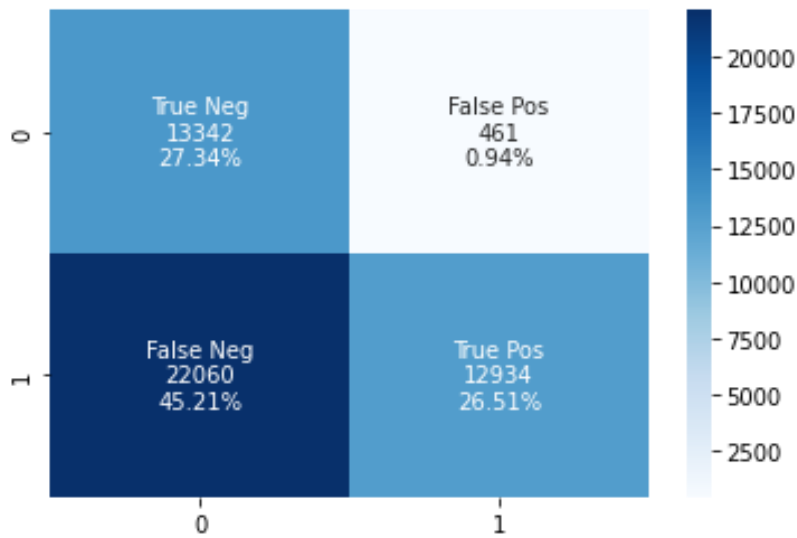
```
classification report:
              precision    recall  f1-score   support

     0       0.38         0.97         0.54     13803
     1       0.97         0.37         0.53     34994

 accuracy          0.54     48797
 macro avg         0.67         0.67         0.54     48797
 weighted avg      0.80         0.54         0.54     48797
```

-----ANN evaluation metrics-----

```
--> accuracy : 0.5384757259667603
--> precision score is : 0.9655841731989548
--> Recall score is : 0.3696062182088358
--> f1-score is : 0.5345843063506168
--> training time: 138.75455141067505
```



### 3. Subset of 55 features

#### 3.1. SVM evaluation results

```
-----classification report-----
-->
      precision    recall  f1-score   support

     0       1.00      0.99      0.99     13701
     1       0.99      1.00      1.00     35096

 accuracy          1.00      48797
 macro avg         1.00      0.99      0.99     48797
 weighted avg      1.00      1.00      1.00     48797

-----SVM evaluation metrics-----

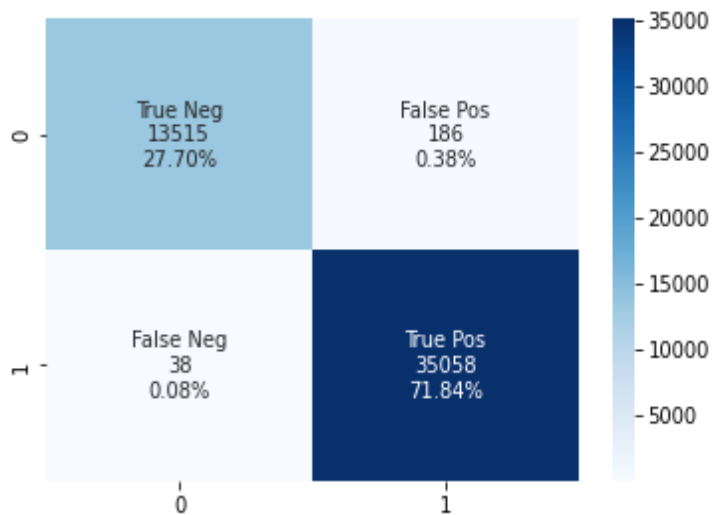
--> accuracy : 0.9954095538660164

--> precision score is : 0.994722505958461

--> Recall score is : 0.9989172555276955

--> f1 score is : 0.9968154677281774

--> training time: 89.56371474266052
```



### 3.2. RF evaluation results

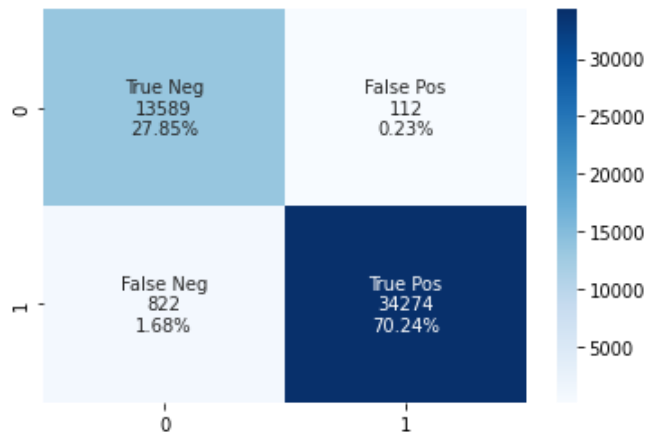
```
-----classification report-----
-->
      precision    recall  f1-score   support

     0       0.94       0.99       0.97     13701
     1       1.00       0.98       0.99     35096

 accuracy         0.98     48797
 macro avg       0.97     0.98     0.98     48797
weighted avg       0.98     0.98     0.98     48797

-----RF evaluation metrics-----

--> accuracy : 0.9808594790663361
--> precision score is : 0.9967428604664689
--> Recall score is : 0.9765785274675177
--> f1-score is : 0.9865576696122738
--> training time: 4.182465314865112
```



### 3.3. ANN evaluation results



```

classification report:
              precision    recall  f1-score   support

     0           1.00       1.00       1.00     13701
     1           1.00       1.00       1.00     35096

 accuracy          1.00          1.00          1.00     48797
 macro avg          1.00          1.00          1.00     48797
 weighted avg       1.00          1.00          1.00     48797

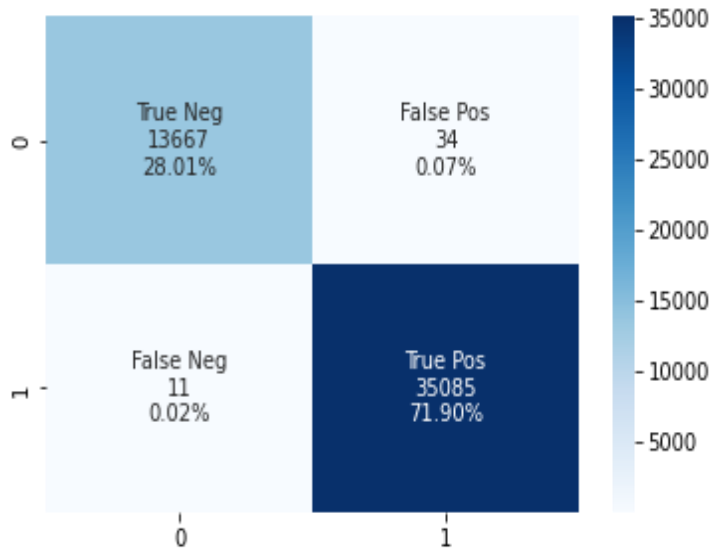
```

-----ANN evaluation metrics-----

```

--> accuracy : 0.9990778121605837
--> precision score is : 0.9990318630940517
--> Recall score is : 0.9996865739685434
--> f1-score is : 0.9993591113010041
--> training time: 99.66347360610962

```



- **Discussion**

According to the obtained results by evaluating the models on subsets of different number of features, clearly the last subset with 55 features give better performance in terms of the evaluation metrics used compared to the other subsets

		<b>Accuracy</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>Training time</b>
<b>28 features</b>	<b>SVM</b>	0.5717	0.9420	0.9941	0.9673	405.40
	<b>RF</b>	0.9103	0.9534	0.9204	0.9366	4.7856
	<b>ANN</b>	0.4882	0.9483	0.3059	0.4626	136.289
<b>33 features</b>	<b>SVM</b>	0.8914	0.9961	0.8518	0.9183	300.0139
	<b>RF</b>	0.9557	0.9711	0.9671	0.9691	4.5596
	<b>ANN</b>	0.5384	0.9655	0.3696	0.5345	138.754
<b>55 features</b>	<b>SVM</b>	0.9954	0.9947	0.9989	0.9968	89.563
	<b>RF</b>	0.9808	0.9967	0.9765	0.9865	4.1824
	<b>ANN</b>	0.9990	0.9990	<b>0.9996</b>	0.9993	99.663