Islamic University of Gaza

Computer Engineering Department

Intrusion Detection System using Machine learning

A graduation project is submitted to graduate in partial fulfillment of the requirements the degree of Bachelor in Computer Engineering

By:

Ismael Al-Safadi 120160618

Submitted to:

Prof. Aiman Abu Samra

Gaza, Palestine

January 2021

Dedication

- **My parents**

For their continuous support

- **My supervisor Prof. Aiman Abu samra**

Who always gives me an ideas , motivation and support

- **My colleagues**

- **To everyone who thought me any new technology**

Abstract

Cyber-attacks became such a big issue in 2020 according to **RiskBased** :“Security revealed that a shocking **7.9 billion** records have been exposed by data breaches in the first nine months of 2019 alone”[1].

One of the most important ways to defense against these attacks is to build a strong “defense in depth “infrastructure , and also of its main components is the Intrusion detection system (IDS)

IDS are software or hardware that analyzes the traffic and it can detect if this traffic is normal or attack.

IDS can be work in many ways one of them is Machine learning based IDS, in this mechanism we using huge dataset of previous attacks and hacking methodologies and analyzing its traffic then use it as features in our dataset.

In my project I’m using **Python programming language,** to build the training and testing modules than I will use some methods to enhance the accuracy to get a good accuracy of detection.

My project will help to detect cyber-attacks on cloud, local networks and also it works as portable IDS on servers and hosts, also it can be enhanced to work on IOT devises.

I have used agile concept with my project development process.

My methodology includes data filtering manually, preprocessing using fitTransform and normalization and the final this is feature selection.

The final results for this project after I did the training and testing for 4 algorithms (KNN, XGBoost,AdBoost and GNB) is that the best algorithm for IDS is the XGBoost with accuracy of detection about 92.5%.

Acknowledgement

I would like to express my deepest gratitude for my parents who have always been there to support me. I also thank my wife who has been strongly supportive to me
to the end of this thesis.

I am also greatly thankful to my supervisor, Prof. Aiman Abu-samra, whose
encouragement, guidance and support from the initial to the final phase enabled me to
develop a deep and thorough understanding of the subject. Finally, I offer my regards
and blessings to all of those who supported me in any respect during the completion of
the thesis.

Table of content

Figures

Chapter 1

1.1 Introduction

As a company or individual computer user I'm thinking how to protect myself against attacks, there are a lot of products that offering protection such as cisco firewalls, snort IDS and others, but the issue is these products are not smart enough to detect the attacks by it selves, it's need an administrator to manage it and writing some roles on it and this can be passed easily by professional hackers.

My project came with new generation of intrusion detection systems that using Machine learning, that can detects attacks without depending on roles that provided by administrator.

The new generation of IDS will be smart enough to detect attacks with high accuracy and also it can be developed to be a self-learning system using recurrent algorithms like "Recurrent neural network (RNN)", that gives a feedback to the training model.

1.2 Statement of problem

Most of the IDS products works as Network based IDS and it's need a lot of configurations and rules to be ready for commercial usage, in my project the IDS will be fixable enough to works on Host and Network also in statistical based and ML with high detection percentage for the malicious traffic.

1.3 Objectives

Develop an IDS works on:

1) Local network(NIDS on span server)
2) Host based
3) Cloud based

1.4 Importance of Project

In terms of the huge number of attacks that happens around the world, every company should focus on how to protect their sensitive data, and the great way to do that is to use detection systems (IDS), but also it must be smart enough to catch the bypassing

techniques, the solution came with Machine learning mechanism, that will provide high accuracy and the cost will be on hand.

Chapter 2 Literature review

2.1 Intrusion detection system (IDS)

2.1.1 History of IDS

By the 1960s, financial systems began to introduce audit practice into their processes to inspect data and check for fraud or errors in systems. However, some questions have arisen: what should be detected, how to analyze what has been discovered and how to protect the various levels of security clearance on the same network without compromising security? Between 1984 and 1986, Dorothy Denning and Peter Neumann developed a first model of IDS, a prototype named as Intrusion Detection Expert System (IDES) [2].

The IDES model is based on the hypothesis that the behavior pattern of an intruder is different enough from a legitimate user to be detected by usage statistics analyzes. Therefore, this model tries to create a pattern of behavior for users in relation to programs, files and devices, both in the short and long term, to make the detection, besides feeding the system based on the rules for representing known violations. By the end of the 1980s, many other systems were developed, based on an approach combining statistical and expert systems [2] .

2.1.2 Definition of IDS

The IDS refer to a mechanism capable of identifying or detecting the presence of intrusive activities. In a broader concept, this encompasses all the processes used in the discovery of unauthorized uses of network devices or computers. This is done through software designed specifically to detect unusual or abnormal activities[2].

However, we must differentiate between IDS and IPS (Intrusion Prevention System). The first one is software that automates the process of intrusion detection; the latter is an intrusion prevention software, which aims to prevent possible attacks. One therefore works in a reactive and informative way, while the IPS reduces the risk of compromising an environment.

2.1.3 Types of IDS

Intrusion detection systems can be categorized into four groups, depending on the type of event they monitor and how they are deployed:

a) Network Based

This type of IDS monitors network traffic on a segment or device, and analyze network and protocol activity to identify suspicious activity. This system is also capable of detecting numerous types of events of interest, and is generally deployed in a security topology as the boundary between two networks, where traffic is tapered. Because of this, in many cases, the IDS feature itself is integrated directly into the firewall [3].

Figure 1.0 depicts a passive deployment of NIDS, where it is connected to a network switch configured with the port mirroring technology.

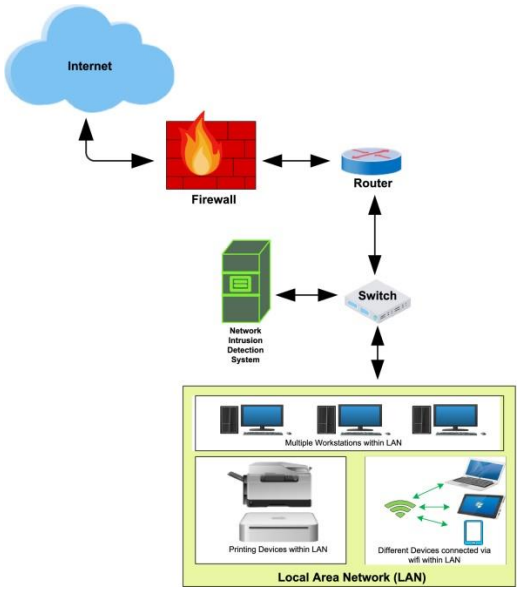

Figure 1.0 the network IDS location connected with the SPAN port in switch to mirroring the traffic form the whole entire network.

b) Host Based

Host refers to an actual device or asset. In this case, we can consider a user's computer, or a server, as a host. Intrusion detection, in this format, monitors device characteristics and the events that happen with it in search of suspicious activity. Usually, a host based IDS can be installed individually for both corporate computers within a corporate network and endpoints. Among its main features are the network traffic to the device, running processes, system logs as well as access and changes in files and applications [4].

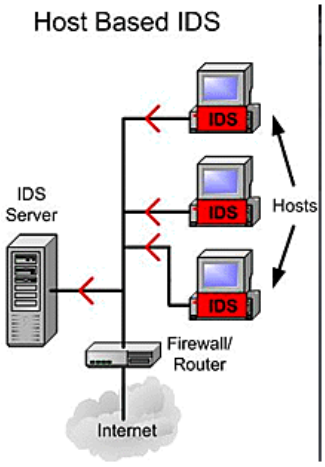

Figure 1.1

Figure 1.1 the host IDS systems, there is a monitoring system connected with every machine in the network, that's why we call it a host-based IDS.

c) Protocol-based Intrusion Detection System (PIDS):

Protocol-based intrusion detection system (PIDS) comprises of a system or agent that would consistently resides at the front end of a server, controlling and interpreting the protocol between a user/device and the server. It is trying to secure the web server by regularly monitoring the HTTPS protocol stream and accept the related HTTP protocol. As HTTPS is un-encrypted and before instantly entering its web presentation layer then this system would need to reside in this interface, between to use the HTTPS [4].

d) Application Protocol-based Intrusion Detection System (APIDS):

Application Protocol-based Intrusion Detection System (APIDS) is a system or agent that generally resides within a group of servers. It identifies the intrusions by monitoring and interpreting the communication on application specific protocols. For example, this would monitor the SQL protocol explicit to the middleware as it transacts with the database in the web server.

By the technique of detection we can divide it into

1) **Knowledge-based /Signature-based**

A knowledge/signature-based IDS references a database of known system vulnerability profiles to identify active intrusion attempts. In this case, it is very important that the structure has a policy of continuous updating of the database (signatures) to ensure continuity of security in the environment, since what is not known will literally not be protected [4].

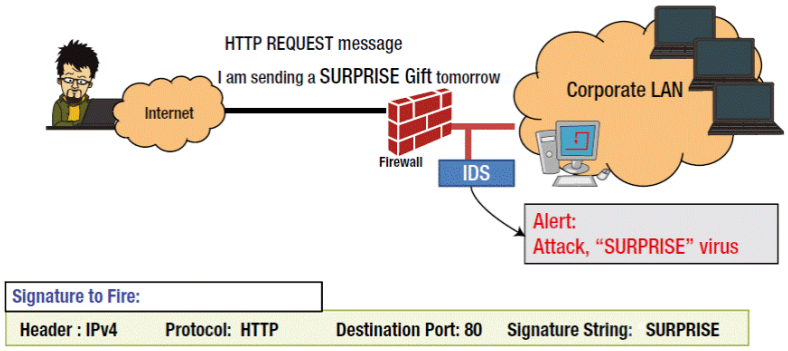

Figure 1.2 the process of Signature-based IDS, we setting some roles on the configuration file then it will check for packets that matches the signatures.

2) **Anomaly-based Method**

Anomaly-based IDS was introduced to detect the unknown malware attacks as new malware are developed rapidly. In anomaly-based IDS there is use of machine learning to create a trustful activity model and anything coming is compared with that model and it is declared suspicious if it is not found in model. **Machine learning based method** has a better generalized property in comparison to signature-based IDS as these models can be trained according to the applications and hardware configurations [4].

2.2 Machine learning

2.2.1 What is machine learning?

At a very high level, machine learning is the process of teaching a computer system how to make accurate predictions when fed data.

Those predictions could be answering whether a piece of fruit in a photo is a banana or an apple, spotting people crossing the road in front of a self-driving car, whether the use of the word book in a sentence relates to a paperback or a hotel reservation, whether an email is spam, or recognizing speech accurately enough to generate captions for a YouTube video [5].

2.2.2 Types of machine learning

Machine learning is generally split into two main categories: supervised and unsupervised learning.

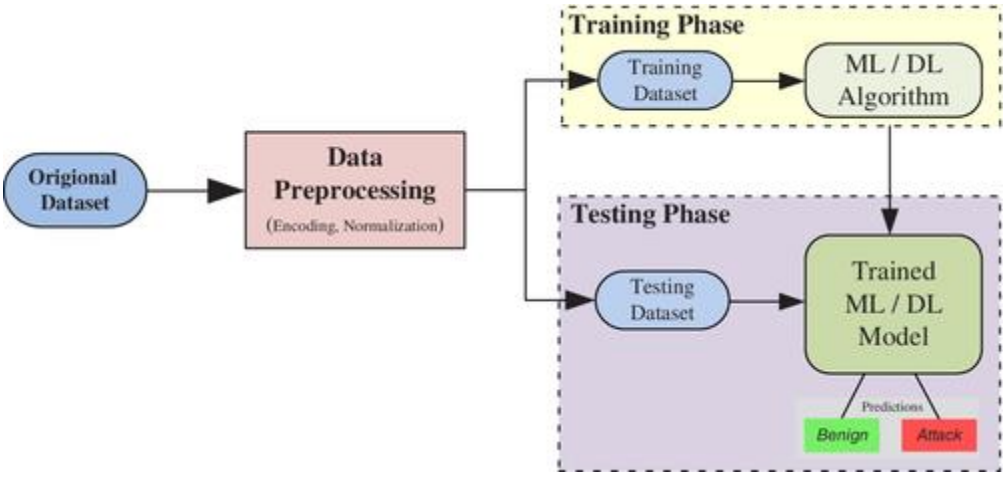

Figure 2.0 the process for machine learning, as shown in the figure the dataset will going through the pre-processing algorithms, then we split it into two parts a Training and Testing

a) **SUPERVISED**

During training for supervised learning, systems are exposed to large amounts of labeled data, for example images of handwritten figures annotated to indicate which number they correspond to. Given sufficient examples, a supervised-learning system would learn to recognize the clusters of pixels and shapes associated with each number and eventually be able to recognize handwritten numbers, able to reliably distinguish between the numbers 9 and 4 or 6 and 8.

However, training these systems typically requires huge amounts of labelled data, with some systems needing to be exposed to millions of examples to master a task [5].

b) UNSUPERVISED

In contrast, unsupervised learning tasks algorithms with identifying patterns in data, trying to spot similarities that split that data into categories.

An example might be Airbnb clustering together houses available to rent by neighborhood, or Google News grouping together stories on similar topics each day.

Unsupervised learning algorithms aren't designed to single out specific types of data; they simply look for data that can be grouped by similarities, or for anomalies that stand out [5] .

2.2.3 Machine learning with IDS

A NIDS developed using ML and DL methods usually involves following three major steps as depicted in **Figure 1.4**, that is, (i) Data preprocessing phase, (ii) Training phase, and (iii) Testing phase. For all the proposed solutions, the dataset is first preprocessed to transform it into the format suitable to be used by the algorithm. This stage typically involves encoding and normalization. Sometimes, the dataset requires cleaning in terms of removing entries with missing data and duplicate entries, which is also performed during this phase. The preprocessed data is then divided randomly into two portions, the training dataset, and the testing dataset. Typically, the training dataset comprises almost 80% of the original dataset size and the remaining 20% forms testing dataset.The ML or DL algorithm is then trained using the training dataset in the training phase. The time taken by the algorithm in learning depends upon the size of the dataset and the complexity of the proposed model. Normally, the training time for the DL models requires more training time due to its deep and complex structure. Once the model is trained, it is tested using the testing dataset and evaluated based on the predictions it made. In the case of NIDS models, the network traffic instance will be predicted to belong to either benign (normal) or attack class [5] .

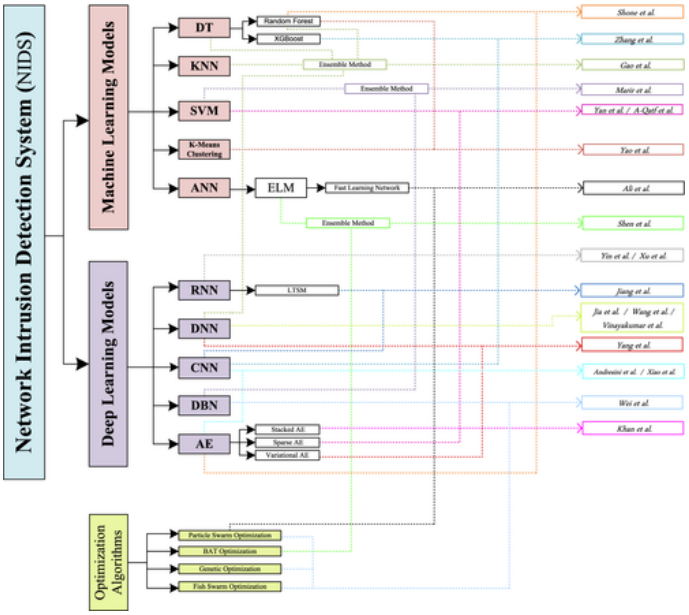

Figure 2.1 the taxonomy of recent ML- and DL-based techniques used for NIDS.

2.2.4 EVALUATION METRICS

All the evaluation metrics are based on the different attributes used in the Confusion Matrix, which is a two-dimensional matrix providing information about the Actual and Predicted class and includes

- True Positive (TP): The data instances correctly predicted as an Attack by the classifier.
- False Negative (FN): The data instances wrongly predicted as Normal instances.
- False Positive (FP): The data instances wrongly classified as an Attack.

- True Negative (TN): The instances correctly classified as Normal instances.

The different evaluation metrics used in the recent studies are:

Precision: It is the ratio of correctly predicted Attacks to all the samples predicted as Attacks.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

Recall: It is a ratio of all samples correctly classified as Attacks to all the samples that are actually Attacks. It is also called a Detection Rate.

$$\text{Recall} = \text{Detection Rate} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

False alarm rate: It is also called the false positive rate and is defined as the ratio of wrongly predicted Attack samples to all the samples that are Normal.

$$\text{False Alarm Rate} = \frac{\text{FP}}{\text{FP} + \text{TN}}.$$

True negative rate: It is defined as the ratio of the number of correctly classified Normal samples to all the samples that are Normal.

$$\text{True Negative Rate} = \frac{\text{TN}}{\text{TN} + \text{FP}}.$$

Accuracy: It is the ratio of correctly classified instances to the total number of instances. It is also called as Detection Accuracy and is a useful performance measure only when a dataset is balanced.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

F-Measure: It is defined as the harmonic mean of the Precision and Recall. In other words, it is a statistical technique for examining the accuracy of a system by considering both precision and recall of the system.

$$\text{F Measure} = 2\left(\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}\right).$$

		Predicted class	
		Attack	Normal
Actual Class	**Attack**	True Positive	False Negative
	Normal	False Positive	True Negative

Figure 2.2 the classification results and how it considered.

2.2.5 Dataset

I selected **UNSW-NB15**: This dataset is created by the Australian Center for Cyber Security. It contains approximately two million records with a total of 49 features, which are extracted using Bro-IDS, Argus tools, and some newly developed algorithms. This dataset contains the types of attacks named as, Worms, Shellcode, Reconnaissance, Port Scans, Generic, Backdoor, DoS, Exploits, and Fuzzers.you can download the dataset from the following link

https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

Chapter 3

Tools used

1) Python

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast.

Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on.

The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

2) Scikit-Learn

Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib!
The functionality that scikit-learn provides include:

- Regression, including Linear and Logistic Regression
- Classification, including K-Nearest Neighbors
- Clustering, including K-Means and K-Means++
- Model selection
- Preprocessing, including Min-Max Normalization [6]

3) **Numpy**

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more [7].

4) **Pandas**

pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals [8].

5) **Matploitlib**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib [9].

6) **Google colab**

Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

Chapter 4 Methodology

4.1 Agile Software Development (ASD):

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like:

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing

Figure 4.1 the process cycle of Agile methodology.

At the end of the iteration, a working product is displayed to the customer and important stakeholders.

Agile model believes that every project needs to be handled differently and the existing

methods need to be tailored to best suit the project requirements. In Agile, the tasks are 17 divided to time boxes (small time frames) to deliver specific features for a release.Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features;

the final build holds all the features required by the customer [10].

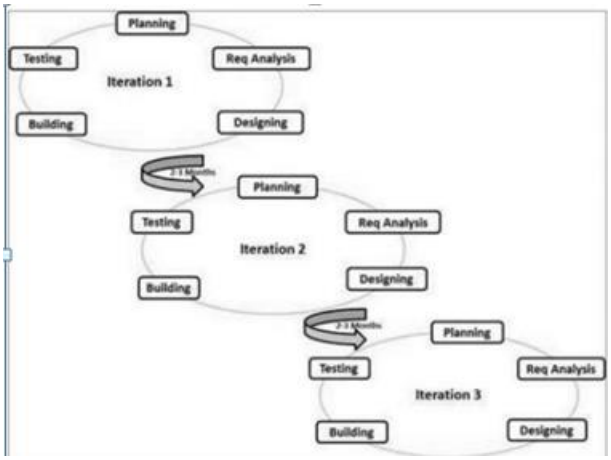

Figure 4.2 the parts of development that follows, we developing a scrum then we doing testing and if everything is goes fine we moving to the next scrum

The Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability. The most popular Agile methods include Rational Unified Process (1994), Scrum (1995),Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995).These are now collectively referred to as Agile Methodologies, after the Agile Manifesto was published in 2001 [10].

Following are the Agile Manifesto principles:

• Individuals and interactions: In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.

• Working software: Demo working software is considered the best means of 18 communications with the customers to understand their requirements, instead of just depending on documentation.

• Customer collaboration: As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.

• Responding to change: Agile Development is focused on quick responses to change and continuous development.

4.2 Why choose Agile module?

There are certain concrete benefits to using agile web development methodologies compared to traditional or other methods. The following section outlines these benefits

1. Keeping Up With Change: The old adage —change is the only constant‖ couldn't be truer when it comes to web development. Instead of viewing changes as unexpected obstacles to overcome, agile developers embrace change as an inevitable part of the learning process.

2. In Agile, testing phase must be done at the end of every sprint. Therefore, we will have an early attention if there is a bug so, we can solve it early.

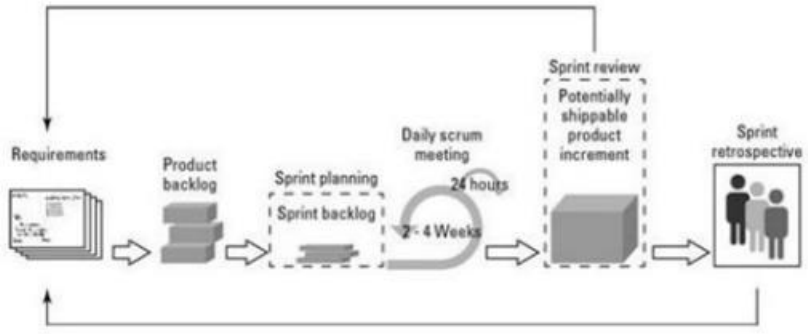

Figure 4.3 the full cycle of Agile from analyzing the requirements moving through backlogs and sprint until finishing the whole entire software

1. **Higher Productivity**: Agile processes provide ample opportunities to measure productivity, which helps project managers better estimate workloads..

2. **Lower Costs**: Faster and more efficient development eliminates the need for overtime pay.

3. **Greater Customer Satisfaction**: Since applications go through so many checks before launch, customers are more likely to get a bug-free product that they are happy with.

4. **Improved Worker Morale**: When team members get to see the fruits of their labor in shippable increments, they know they're progressing in the right direction, which encourages them to keep working toward a goal. Team members are also more likely to feel a sense of shared ownership in the project, which is more motivating than artificial urgency.

5. **Focuses on Business Value**: By allowing the client to determine the priority of features, the team understands what's most important to the client's business, and can deliver the features that provide the most business value.

6. **Better Accountability**: Iterative methodologies make it easier to track the performance of individual team members, and it gives managers a way to measure the team's overall commitment level throughout the development process. This information can help them set more realistic goals and give executives accurate timelines.

7. **No More Detailed Project Plans**: Forgoing a project plan isn't the same as not planning. It simply means that you don't need one specific document that you update every time you learn something new. Agile development's focus on frequent releases encourages an iterative learning process, so there's no reason to waste time and energy managing a document.

Chapter 5

Requirements and Design

5.1 Requirements

Functional requirements

1) Detecting the following attacks

a) Backdoors, software that makes a hidden connection between victim and attacker machines.

b) DoS, large amount of requests that makes servers or networks not able to serve.

c) Exploits, a piece of software that exploit a vulnerability on system

d) Fuzzers, entering random data into a program and analyzing the results to find potentially exploitable bugs

e) Port scans, scan the open ports on system (scanning is the second step of hacking process)

f) Reconnaissance, identify the IP address and discover basic info (first step of hacking process)

I) Shell code, a code that used to initiate a connection between two devices

j) worms , malwares that spread-out automatically

2) Get high accuracy of detection

3) System should working without adding roles on it like other IDS systems

4) System should be portable to work ad NIDS and HIDS

Non-functional requirements

1) high performance needed
2) user friendly
3) high privacy

5.2 System Design

My system created to serve on NIDS and HIDS, with high accuracy of detection.

The complete product will start by:

1) Select proper dataset
2) Doing preprocessing for dataset
3) Execute feature selection to extract the most important features
4) Doing fit-transform to make sure that data is able to processing
5) Doing normalization to enhance results
6) Train the model
7) Test the model
8) monitoring data from incoming and outgoing traffic
9) Process this data and extract features
10) Pass these features to the trained model that prepared previously
11) Gets result of the prediction

The following Figure4.1 showing the process of my project

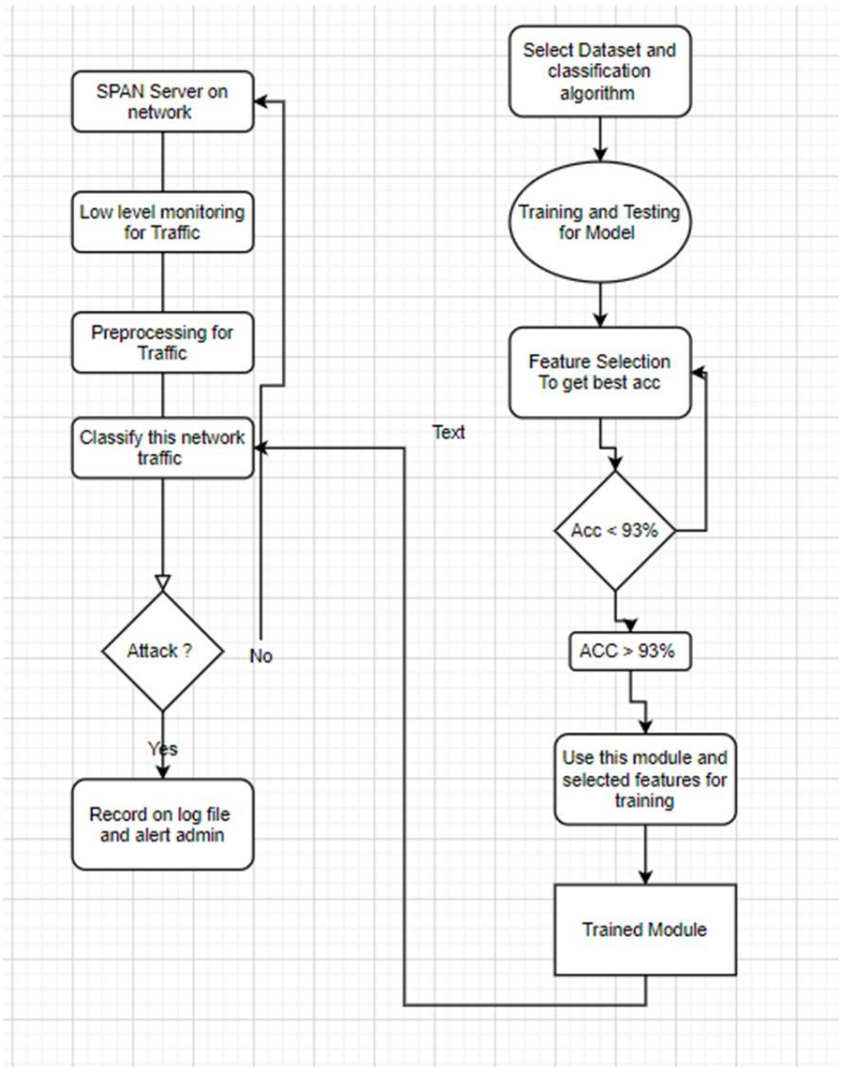

Figure 5.1 the Flow chart design of my project that describes process from getting data from SPAN port moving through data splitting and model creation then prediction.

The expected ACC was 93% but the real ACC is 91.4%.

Chapter 6

Implementation and results

6.1 Dataset Pre-processing

I found some Impurities in the dataset that cozzes some errors during the training process, these errors like non-usual values and nulls

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	
1	srcip	sport	dstip	dsport	proto	state	dur	sbytes	dbytes	sttl	dttl	sloss	dloss	service	sload	dload	spkts	dpkts	swin	dwin	stcpb	dtcpb	smeansz	dm
2	59.166.0.9	7045	149.171.1	25	tcp	FIN	0.201886	37552	3380	31	29	18	8	smtp	1459438	130766.9	52	42	255	255	1.42E+09	3.57E+09	722	
3	59.166.0.9	9685	149.171.1	80	tcp	FIN	5.864748	19410	1087890	31	29	2	370	http	26404.54	1481983	364	746	255	255	3.9E+08	3.95E+08	53	
4	59.166.0.2	1421	149.171.1	53	udp	CON	0.001391	146	178	31	29	0	0	dns	419841.8	511862	2	2	0	0	0	0	73	
5	59.166.0.2	21553	149.171.1	25	tcp	FIN	0.053948	37812	3380	31	29	19	8	smtp	5503374	489360.1	54	42	255	255	4.05E+09	1.9E+09	700	
6	59.166.0.8	45212	149.171.1	53	udp	CON	0.000953	146	178	31	29	0	0	dns	612801.7	747114.4	2	2	0	0	0	0	73	
7	59.166.0.0	59922	149.171.1	6881	tcp	FIN	8.633186	25056	1094788	31	29	38	390	-	23166.42	1013311	446	858	255	255	4.97E+08	5.27E+08	56	
8	175.45.17(	49582	149.171.1	80	tcp	FIN	0.189983	13304	268	254	252	6	1	http	529100	9432.423	18	6	255	255	1.58E+08	4.69E+08	739	
9	175.45.17(	0	149.171.1	0	sctp	INT	0.000009	440	0	254	0	0	0	-	1.96E+08	0	2	0	0	0	0	0	220	
10	175.45.17(	0	149.171.1	0	sctp	INT	0.000009	440	0	254	0	0	0	-	1.96E+08	0	2	0	0	0	0	0	220	
11	175.45.17(	0	149.171.1	0	sctp	INT	0.000009	440	0	254	0	0	0	-	1.96E+08	0	2	0	0	0	0	0	220	
12	175.45.17(	0	149.171.1	0	sctp	INT	0.000009	440	0	254	0	0	0	-	1.96E+08	0	2	0	0	0	0	0	220	
13	175.45.17(	0	149.171.1	0	sctp	INT	0.000009	440	0	254	0	0	0	-	1.96E+08	0	2	0	0	0	0	0	220	
14	175.45.17(	0	149.171.1	0	sctp	INT	0.000009	440	0	254	0	0	0	-	1.96E+08	0	2	0	0	0	0	0	220	
15	175.45.17(	0	149.171.1	0	sctp	INT	0.000009	440	0	254	0	0	0	-	1.96E+08	0	2	0	0	0	0	0	220	
16	59.166.0.6	49434	149.171.1	80	tcp	FIN	1.366053	1684	10168	31	29	3	5	http	9159.234	56243.79	14	18	255	255	1.3E+09	3.54E+09	120	
17	59.166.0.1	32219	149.171.1	53	udp	CON	0.001028	146	178	31	29	0	0	dns	568093.4	692607	2	2	0	0	0	0	73	
18	59.166.0.1	53228	149.171.1	53	udp	CON	0.001016	130	162	31	29	0	0	dns	511811	637795.3	2	2	0	0	0	0	65	
19	59.166.0.3	2803	149.171.1	33687	tcp	FIN	0.040832	2750	25564	31	29	7	15	-	526645.8	4894788	44	44	255	255	2.38E+09	2.4E+09	63	
20	59.166.0.6	50368	149.171.1	27436	tcp	FIN	0.021008	2542	22128	31	29	7	14	-	944021.3	8226200	40	42	255	255	1.15E+08	2.26E+09	64	
21	59.166.0.0	23677	149.171.1	6881	tcp	FIN	9.887264	36762	1641360	31	29	55	583	-	29700.43	1327021	660	1278	255	255	2.64E+09	5.14E+08	56	
22	59.166.0.0	44972	149.171.1	45817	tcp	FIN	0.022846	3984	2560	31	29	7	7	-	1317692	851615.1	18	20	255	255	4.15E+08	2.56E+09	221	
23	59.166.0.5	55280	149.171.1	55325	tcp	FIN	0.05809	2646	25564	31	29	7	15	-	355723.9	3440592	42	44	255	255	2.14E+09	2.16E+09	63	
24	59.166.0.6	53913	149.171.1	52000	tcp	FIN	0.030868	2750	25564	31	29	7	15	-	696643.8	6474796	44	44	255	255	1.15E+08	1.28E+08	63	
25	59.166.0.0	60479	149.171.1	6881	tcp	FIN	10.374	35516	1551472	31	29	52	551	-	27346.05	1195440	638	1208	255	255	4.97E+08	2.66E+09	56	
26	59.166.0.5	2992	149.171.1	22	tcp	FIN	5.182327	26664	49682	31	29	78	103	ssh	41042.57	76479.93	342	356	255	255	3.44E+09	1.29E+09	78	
27	59.166.0.0	6697	149.171.1	111	udp	CON	0.004779	568	320	31	29	0	0	-	713119.9	401757.7	4	4	0	0	0	0	142	
28	59.166.0.1	43183	149.171.1	32912	tcp	FIN	0.006673	424	8824	31	29	1	4	ftp-data	444777.5	9697588	8	12	255	255	2.2E+09	63057888	53	
29	59.166.0.0	49549	149.171.1	80	tcp	FIN	1.92579	1684	10168	31	29	3	5	http	6497.074	39896.36	14	18	255	255	3.51E+09	1.41E+09	120	
30	59.166.0.0	8107	149.171.1	2618	udp	CON	0.001685	520	304	31	29	0	0	-	1851632	1082493	4	4	0	0	0	0	130	
31	59.166.0.1	4482	149.171.1	11680	tcp	FIN	0.004005	320	1938	31	29	1	2	ftp-data	533333.3	3387765	6	8	255	255	2.2E+09	59453931	53	
32	59.166.0.3	1305	149.171.1	53716	tcp	FIN	0.022919	2646	22128	31	29	7	14	-	901610.1	7540295	42	42	255	255	2.38E+09	2.35E+08	63	

Figure 6.01 a part of my dataset that used in this project and there is a lot of hyphenated data inside it

As shown in [Figure 6.01], on column “N” there are a lot of hyphenated data the cozzes many errors while we doing the Training or Testing.

1) First I need to delete these hyphens manually, also deleting nulls.
2) Second we need to doing a fit-transform that will change all nun-numerical numbers to a numerical numbers, using the following code

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
srcip = le.fit_transform(list(data["srcip"]))
dstip = le.fit_transform(list(data["dstip"]))
proto = le.fit_transform(list(data["proto"]))
state = le.fit_transform(list(data["state"]))
service = le.fit_transform(list(data["service"]))
```

Figure 6.02 code doing a part of pre-processing by importing LabelEncoder that will convert the non- numerical values into numerical values
This piece of.

6.2 Dataset Visualization

Doing dataset Visualization to understand the nature of our dataset

Code for Visualization

```
from yellowbrick.features import PCA
visualizer = PCA(scale=True, classes=classes)
visualizer.fit_transform(X_train, y_train)
visualizer.show()
```

Figure 6.03 code will visualize my dataset the get the intensity of the Normal and attack values and how its distributed in a visualize view

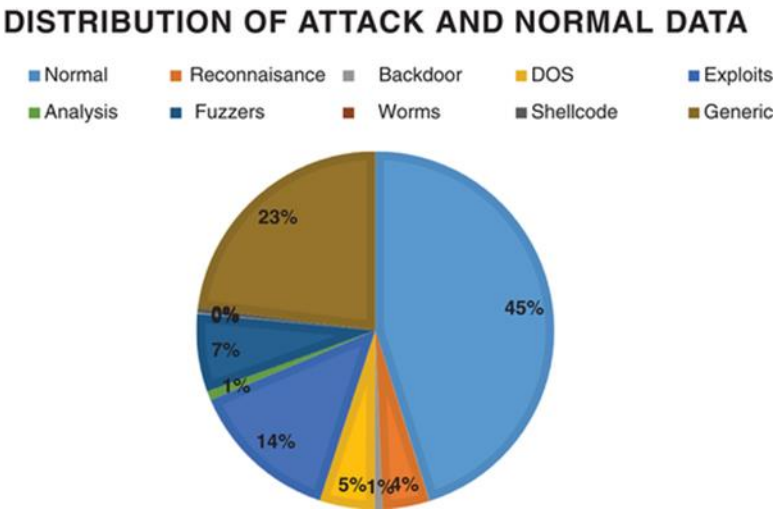

Figure 6.04 is the result of executing Code 1.2, It shows the distribution of attack types and the percentage of every type of attacks.

Writing the processed data into a new dataset file CSV

```
with open('New_dataset.csv', mode='w', newline='') as
New_dataset:
    New_dataset_writer = csv.writer(New_dataset, delimiter=',')
    for i in X:
        New_dataset_writer.writerow(i)
```

Figure 6.05 the code I just rewrite the processed data in a new CSV file the reduce the processing with every run of my code, because Pre-processing will take an extra time to be done, instead of doing it every time I just did it once than saving it into new file.

6.3 Feature Selection

Doing feature selection using one of the most known algorithms its Best-k selection
Using the following code

```python
import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
data = pd.read_csv("dataset.csv")
X = data.iloc[:,0:30]
y = data.iloc[:,-1]
bestfeatures = SelectKBest(score_func=chi2, k=30)
fit = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']
print(featureScores.nlargest(30,'Score'))
```

Figure 6.06 this piece of code im doing Feature selection that helps with implementation of the software and also it enhanced the accuracy.

The SelectKBest class just scores the features using a function (in this case f_classif but could be others) and then "removes all but the k highest scoring features"

The following diagram shows how this algorithm works

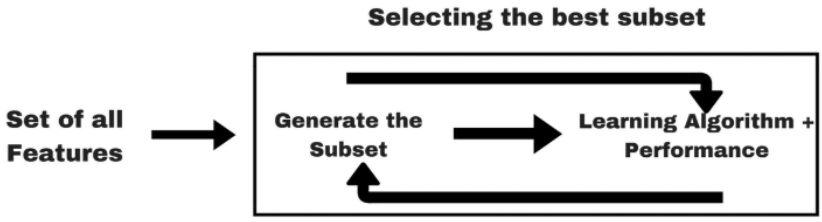

Figure 6.07 show the process of executing the feature selection and how it can be used.

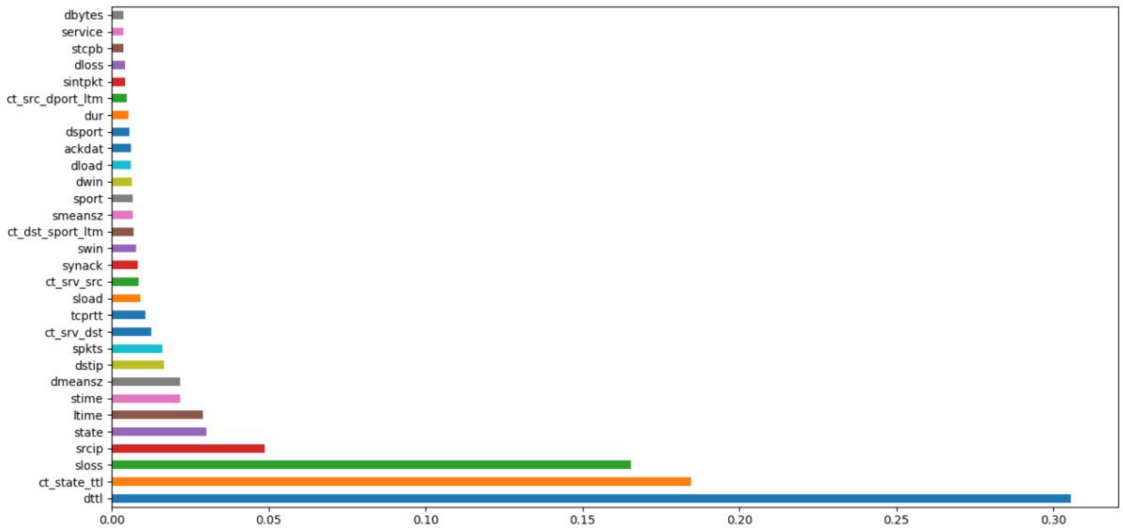

Figure 6.08 the most important features that effects accuracy, recording to SelectKBest algorithm

6.4 Dataset Normalization

The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges.

For example, consider a data set containing two features, age, and income(x2). Where age ranges from 0–100, while income ranges from 0–100,000 and higher. Income is about 1,000 times larger than age. So, these two features are in very different ranges. When we do further analysis, like multivariate linear regression, for example, the attributed income will intrinsically influence the result more due to its larger value. But this doesn't necessarily mean it is more important as a predictor. So we normalize the data to bring all the variables to the same range [11].

Not normalized dataset

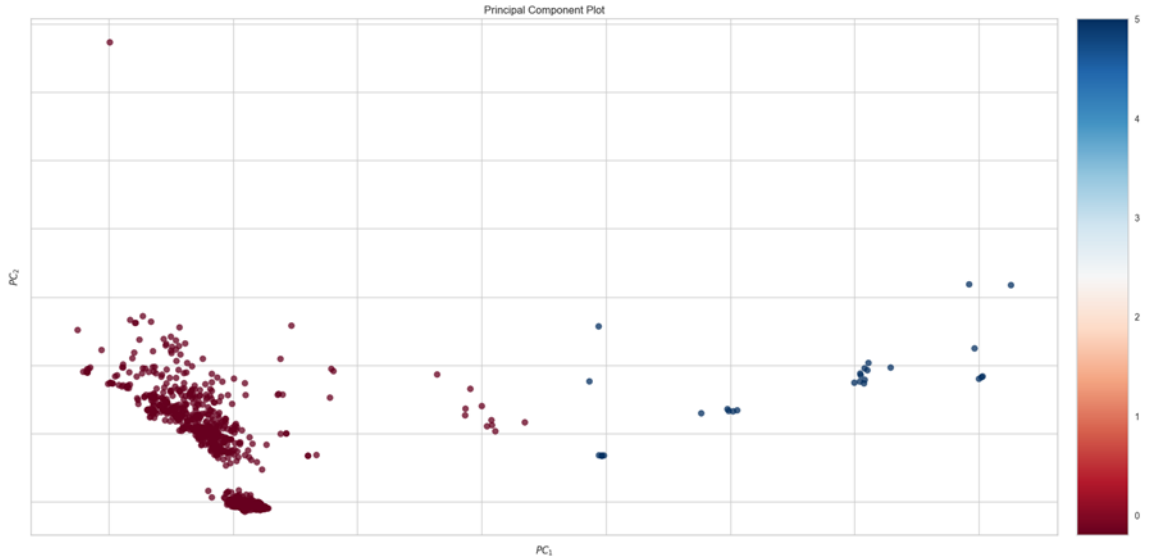

Figure 6.09 the different between normalized dataset and

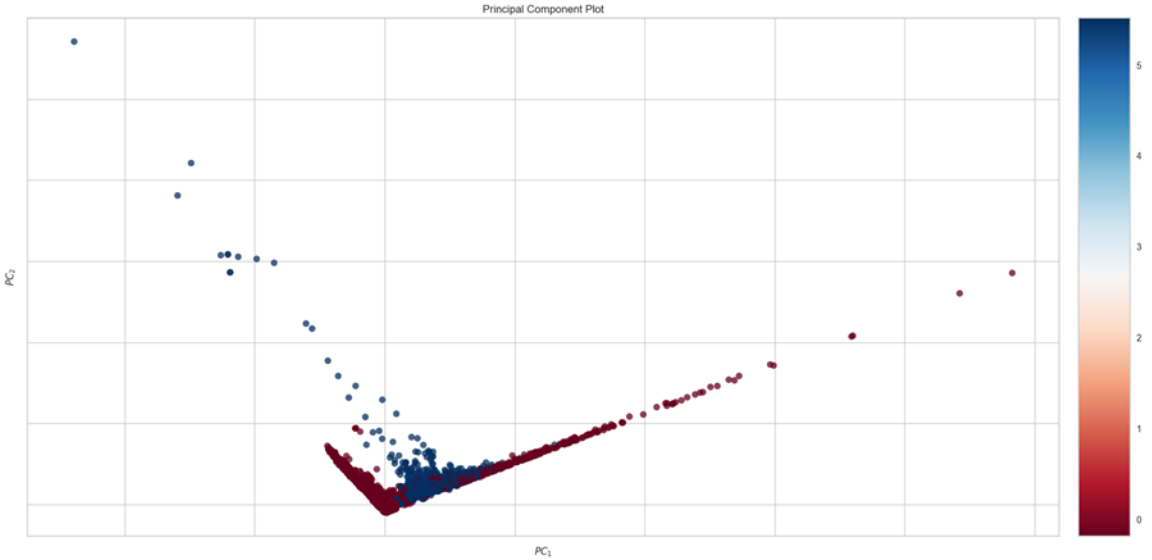

Figure 6.10 (Normalized dataset), dataset is stable and the values became near to each other, this will help with getting high accuracy of testing

6.5 Training and Testing

Split dataset into two parts Training and Testing

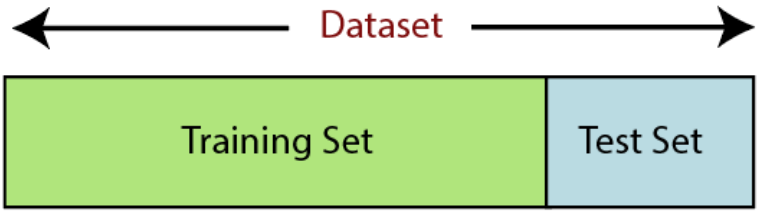

Figure 6.11 how the dataset will be spitted into training and testing

Training using multiple algorithms as follows XGBoost,GNB,KNN and AdBoost

After training we testing the accuracy of the model and the results

Algorithm	Training time(sec)	Testing Time(sec)	Accuracy	Sensitivity	Specificity	F-Measure
GNB	20.204	3.506	74.51%	0.7642	0.7055	0.746
XGBoost	57.77	1.153	92.53 %	0.951	0.902	0.996
KNN	260,280	53.85	90.56%	0.892	0.8001	0.884
AdBoost	9569.65	**43.69**	84.06%	0.827	0.810	0.822

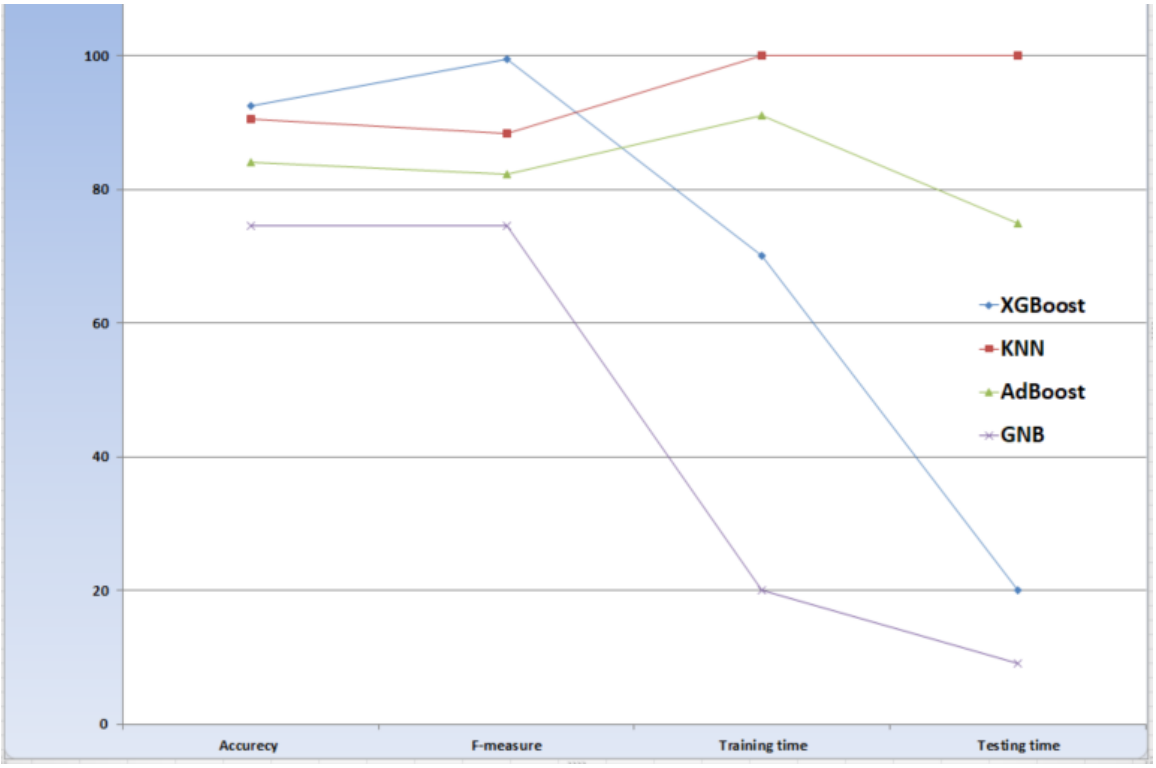

Figure 6.12 visualization chart represents the Accuracy, F-Measure, Training time and testing time

Training and testing time are scaled to be proper with the same scale 100, because the time calculated in seconds e.g. The training time for KNN was about 206298 seconds, but we can't represent the number in the same curve that's why I'm using scaling.

Results analysis

In IDS systems we have hug data that moving through the IDS system, that means we need:

1) a high performance system to process this large amount of data
2) low testing time (detection time) to give a fast alert the administrator
3) High accuracy of detection rather than other systems like signature based.

Referring to figure 6.12 that shows the results of testing we can conclude the following

a) **KNN: the accuracy is good but the testing time is so high then the attack will be executed and we will get the alert after hacker finishing his attack.**
b) **GNB: Testing and Training time is perfect but the accuracy is not the good, so some attacks can penetrate our system.**
c) **AdBoost: Good accuracy but the same problem as KNN high detection time**
d) **XGBoost: Such a good accuracy and perfect detection time, this algorithm will be perfect for the project.**

The following table describes this comparison

Algorithm	Training time(sec)	Testing Time(sec)	Accuracy
GNB	✓	✓	×
XGBoost	✓	✓	✓
KNN	×	×	✓
AdBoost	×	×	✓

4) so we can conclude that the less testing time will consider better that large testing time, also the accuracy of detection will be considered as an important point, depending on the results, we can conclude that is the best algorithm for IDS system is **XGBoost.**

References

[1] https://sectigostore.com/blog/cyber-attacks-2020-notable-2020-cyber-attacks/

[2] https://en.wikipedia.org/wiki/Intrusion_detection_system

[3] https://blog.eccouncil.org/how-do-intrusion-detection-systems-ids-work/

[4] https://onlinelibrary.wiley.com/doi/full/10.1002/ett.4150

[5] https://www.expert.ai/blog/machine-learning-definition/

[6] https://scikit-learn.org/stable/

[7] https://numpy.org/

[8] https://pandas.pydata.org/

[9] https://matplotlib.org/

[10] https://www.guru99.com/agile-scrum-extreme-testing.html

[11] https://towardsai.net/p/data-science/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff

[12] https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[13] Black Hat Python: Python Programming for Hackers and Pentesters by Justin seitz