LEBANESE UNIVERSITY
FACULTY OF TECHNOLOGY

Thesis for Obtaining
Master's degree II
ENGINEERING OF INFORMATION SYSTEMS

LABORATORY OF EMBEDDED AND NETWORKED SYSTEMS

DNS Traffic Classification

Presented by:
Manar Hussein Sayyah

Supervisor: Dr. Hussein El Ghor

Promotion: 2019-2020

ACKNOWLEDGMENT

*Foremost, I would like to express the deepest appreciation to my thesis supervisor **Dr. Hussein El Ghor** for the continuous helping in my thesis study and research, for his patience, motivation, and his genius knowledge that he has added to the thesis. His instructions helped me all the time of research and writing of the thesis. I could not have a better advisor or mentor like Dr. Hussein for my thesis study. He is really a genius person with a lot of experience and amazing knowledge that open my mind to important tasks and an important world.*

*I would like to thanks also **Dr. Jawad Khalife** that was the watchful eye for applying my thesis and was the source of support for us with his experience and the extent of his love for giving to others. I thank him from my heart because we had a father who took care of his children. In all thesis duration, he supports us with his practical experience and expanded our field of thinking.*

*My sincere thanks, also go to the dean of Faculty of Technology University **Prof. Mohammed Al Hajjar**, and **Dr. Haissam Al Hajjar** the head department of information system engineering, for their continuous encouragement, their support for my career goals, and their patience to make us reach this goal. And because he gives us a quiet study atmosphere at the university.*

Lastly, I would like to thank my family for giving me a good personality, teaches me how to face life, supporting my academic trip, and their motivation to continue my master's degree that is the clincher point in my career life.

Saida, 2020

Manar Hussein sayyah

TABLE OF CONTENTS

TABLE OF FIGURES

ABBREVIATIONS

DNS	:	Domain Name System
DT	:	Decision Tree
DNS	:	Domain Name System
SGD	:	Stochastic Gradient Descent
RELU	:	Rectified Linear Unit
SELU	:	Scaled Exponential Linear Unit
RMSprop	:	Root Mean Square Propagation
CSV File	:	Comma-Separated Values File
JUPYTER	:	Loose acronym meaning Julia, Python, and R
KNN Algorithm	:	K-Nearest Neighbors
NN	:	Neural Network
CART	:	Classification and regression trees
ML	:	Machine Learning
MLT	:	Machine Learning Techniques
Sklearn	:	Sci-kit learn
AI	:	Artificial Intelligence
UL	:	Unsupervised Learning
SL	:	Supervised Learning

ABSTRACT

In our thesis, we will talk about an important subject that is DNS Tunneling, which is recognized as cyber-attack security throughout DNS protection where machine learning considered as one of the important technique to detect this type of threat, and researcher count hardly on it like Decision tree algorithms or neural network.

Furthermore, every day billions of legitimate packets traverse computer networks where regrettably, malicious traffic also traverses these same networks so, imagine how danger it is! As an example of this is traffic, is the misusage Domain Name System (DNS) protocol in extracting sensitive data, build backdoor tunnels, or control botnets, that's why this report will study how to use machine learning methods to detect DNS Tunneling with the comparison of its results and which method is better to detect anomalies. However, the interest in data science that increases, machine learning is become an important task of computer science methods. By which, undergraduate students in the university starting to learn machine learning and data science, these courses usually cover only unsupervised and supervised learning classification, permit students to graduate with a few culture of other important areas of machine learning which is a very important subject to learn it, also support this culture by giving them an intensive course about machine learning as a whole.

Our study depends on a comparison between the results based on decision tree algorithms and deep learning (NN). Where, each decision tree algorithms is discussed deeply with its way of calculation. Thus, it's important today to use machine learning in data analysis that could compute millions of data and gives excellent credible results after that.

Let's assume that researchers uses their hands to analyze billions of data it would takes at least one year for that, for that DNS anomalies is a very dangerous cyber-attack that should have a fast way to detect before it happens. So, we can't wait for one year to analyze the data and get its predication. Whereby, Machine learning will reveal the attack results before they happen by depending on old data that the machine was learned.

Chapter I:

INTRODUCTION

Domain Name System (DNS) is one of the important protocols that has a necessary role in web activities such as, browsing and emailing. It is the main network services to convert or translate the domain name into an IP address. So that DNS protocol is not for data transferring but converting the friendly URLs into numerical (IP address).

One of the concepts that consider the information security community in the last decade is DNS Tunneling that is a very critical problem in the Internet's world. However, DNS Tunneling is the tunnel that permits any wrong data to pass through it. The clever hackers noted that they could communicate with a target PC by sneaking in sensitive data or commanding into the DNS protocol. Domain name server converted by DNS tunnel is a type of attack that will destroy any data that will pass through it.

In our thesis, we covered critical and serious problems that face the Internet world from year 2004. This problem affects the users of the DNS protocol where their sensitive data and secret information were stolen by hackers and consequently data may be destroyed. Some hackers make abuse and destroy the data by stealing all the DNS tunneling data in other words we can call it goast tunneling. So, DNS Tunneling is a very important issue that we should study, discuss, and search for ways to detect it. This issue is very dangerous because it bypasses the firewalls easily so we should be attuned to get rid of it.

This thesis aims to discuss this problem deeply and search for solutions to help discover those abusers. Many organizations and companies do not take into consideration any threats regarding the DNS, since it is not linked to the data transfer. However, many companies could be vulnerable to numerous types of threats throughout the DNS. This is due to the respected amount of traffic that should be subjected to DNS threats, so this thesis will show how much this issue is dangerous and should be taken into consideration by users and researchers.

The DNS Tunneling can be detected by Traffic analysis or payload analysis. Also, Machine Learning was recently used in detecting DNS tunnels due to their power to analyze and predict the occurrence of DNS tunneling. Nowadays, Machine learning is an important concept that should consider it. It is a field of computer science that is developed from pattern recognition study and computational learning theory in artificial intelligence. Where, Building and learning such algorithms can learn from the detailed data to make a future prediction, also Machine learning will help humans to predict things that may happen in the future using data that has been stored, so we will avoid problems before happening by using machine learning methods, For an example, from the dataset (PUF) we can know if there are anomalies or not, but there are 29000 rows approximately, and consequently, it is very difficult for humans to analyze all this data. Hence, using machine learning methods to analyze this data will give the predictions and results that the user needed. It is mostly used for efficient tunneling detection and can be a good way to prevent this threat. ML gives a method to define the normal behavior in a network, so it can discover the presence of anomalies (DNS TUNNEL in our case). In addition, we can found many MLTs such as Support Vector Machine, Naïve Bayes, Decision Tree, K-nearest neighbor, so we can say that machine learning plays an essential role in DNS detection we will start to discuss in this report each concept in details to reach the target.

We focus on the Decision tree algorithm, which is a supervised learning method that is the most used in machine learning. A decision tree is the simplest way in machine learning and it gives good results in detecting any type of anomalies, also we discuss many DT algorithms where each one has its way to give their results and each one has the path that walks through it to detect the anomalies (from top to down, from greater to the lowest), so For each decision tree algorithms (ID3, C4.5, and CART) we will study deeply, and their ways to calculate each one using (Entropy, GINI Index, Information gain). We study the algorithms and the way of calculation to reach the results where emphasis also studied Neural Network to detect anomalies such as, deep learning. Neural networks are used to change input, so the network leads to the best possible result without needing to redo the output criteria. By the comparison of Neural Network and Decision tree algorithms the results studied showed the best algorithm after comparing the simulation results.

1. Background of the work:

With the power of the firewalls to catch the threats in our issue it does not make sense. The hackers that use DNS Tunneling will easily pass through the firewall without any alert to the victims. So we use machine learning, which is a good tool that alert the DNS users before the problem happens.

Based on decision tree algorithms, there are many problems faced during our working on these algorithms when using the C4.5 algorithm it demands from us to write special code in an external file and called with python code to use in anaconda navigators, since in anaconda there is no default library defined to use and calculate the C4.5 algorithms.

Also, we face a problem in the results of decision tree and NN because the dataset contained is huge that will lead to a big tree for decision tree with many branches and any number of nodes for NN that make our analysis is more complicated. But we calculate the Evaluation (Accuracy, Precision, Recall, and F1_score) and Basic values (TP, TN, FP, and FN) for each DT algorithms and NN that allow us easily to walk in the road toward having complete analysis operation.

The NN values were not acceptable in the beginning, that's why we directed toward using the weighted process to enhance these values and to get more benefits. But using the weighted process the result in the testing set affected negatively all the analysis and hence, we discuss below the NN graph analysis after calculating the Evaluation values.

In this thesis, we attempt to answer these questions in the computer network traffic classification context.

2. Methodology

Firstly, we start with the DNS Tunneling definition and how much it is critical in the Cyber-security domain. For that we take this issue in deep detail by considering all its edges. So during this thesis we will explain every concept.

After that, we take steps toward the detection of this problem by using machine learning methods that is a worldwide domain that many researchers work on it. Where we study the below Decision

tree methods in supervised machine learning classification with its algorithms and we choose from unsupervised machine learning the Neural Network that is deep machine learning.

Thirdly, we compute the Evaluation and Basic values for each DT algorithms with a full discussion of the techniques used in each calculation and also for the NN techniques which facilitates the analysis operation and the compare between two methods, evaluation simulation values are calculated using the python code and many experimental setups are studied.

Subsequently, we made a comparison to the calculated results of these methods to demonstrate the better machine learning method for DNS Tunneling Detection.

3. Thesis Overview

The structure of our thesis is like the following: chapter I shows the machine learning definitions, methods, supervised Classification and regression, decision tree algorithms, and neural network with a deep analysis of them. Chapter II presents how to deal with the new dataset how we deal with the dataset that contained and the change that has been done on it. Moreover, in chapter III, we present the experimental setup that has done with python language in anaconda navigator with sci-kit learn libraries and the implementation of the three DT algorithms that we choose and the NN techniques with the preformed calculations. Finally, in chapter IV, we make a comparison between the results we get from each machine learning method to discover the better machine learning method we should use to detect the DNS Tunneling anomalies.

Chapter II:

Introduction to DNS Tunneling

A covert channel is a channel that permits to send or transfer of information or data in a way that violates system security policies. These channels do his violence without any alarms. Covert channel involves hiding data in the least significant bits of each pixel of the bitmap image. The image will be similar to the original and from a technical look, it is indistinguishable.

There are many types of covert channels. One of those types is called network covert channels that are used to bypass firewalls without any alarms or response. This thesis studies DNS Tunnels, a realization of network covert channels.

Domain Name System is one of the important protocols that plays a significant role in terms of web browsing and emailing. DNS provides a way to translate a domain name to an IP address. This transmission had made since IP addresses are hard to remember every time users have to go to any website on the internet. Tunneling is transferring one network protocol inside another one this process is called encapsulation.

DNS Tunnel is network covert channels that permit the transfer of arbitrary data by using the DNS infrastructure. The abuser that uses this tunnel is just to hide their communication sessions to bypass the local security. The DNS Tunnel domain still has limited research, and there are few about it working way. DNS tunnel, the objective of DNS tunneling is to basically use the DNS protocol as the transport mechanism this can be used by abuser or attacker to collect aggregate and transport are exfiltrate sensitive information using DNS.

The cause of using DNS protocol because of its necessity in every application people use from web browsing to email to other IP application DNS is the critical ingredient or protocol that used to translate human-readable text web address into computer-readable or useable binary addresses or IP addresses. DNS general permitted to through networks attackers.

DNS tunneling in details:

DNS is a protocol that all time used by internet users. DNS basically translates domain names or URLs to IP addresses. For example, if somebody is bored from the work so, it's time to go to example.com. He goes to his browser and type in example.com he doesn't have to remember or memorize the IP address of example.com the DNS protocol will actually translate the web address that he typed into the IP address his computer can get there now. When his computer doesn't know the IP address of the website you're trying to go to call DNS server. When they ask the DNS server to respond with the IP address that asks is called a query.

Data exfiltration is one of the workstations in the internet environment that become infected with malware that has access to sensitive data like (credit card number, personal information, or intellectual property). Having access is bad, but it becomes a much bigger problem if the malware can move the sensitive data out of the environment to where it can be exploited. To do this, the malware can try to send the data directly out, but chances are there is a firewall in place to prevent that but because of the importance of DNS in network communications, and it needs to reach external DNS servers to do its job, almost all firewalls have holes to allow DNS traffic. To take advantage of this, the malware packages the sensitive data up as DNS queries and sends it to the local DNS server. The queries are designed so that the local DNS server can't directly respond and will therefore forward the query through the firewall to an external DNS server (Root DNS Server) for an answer. The external DNS server may or may not return an answer, but it will most certainly collect the information contained in the DNS query this is the exfiltrated data. Over time, the malware continues to package up more data and send it out in steady trickle until all the data has been exfiltrated all of this while using DNS in exactly the way it was designed to work.

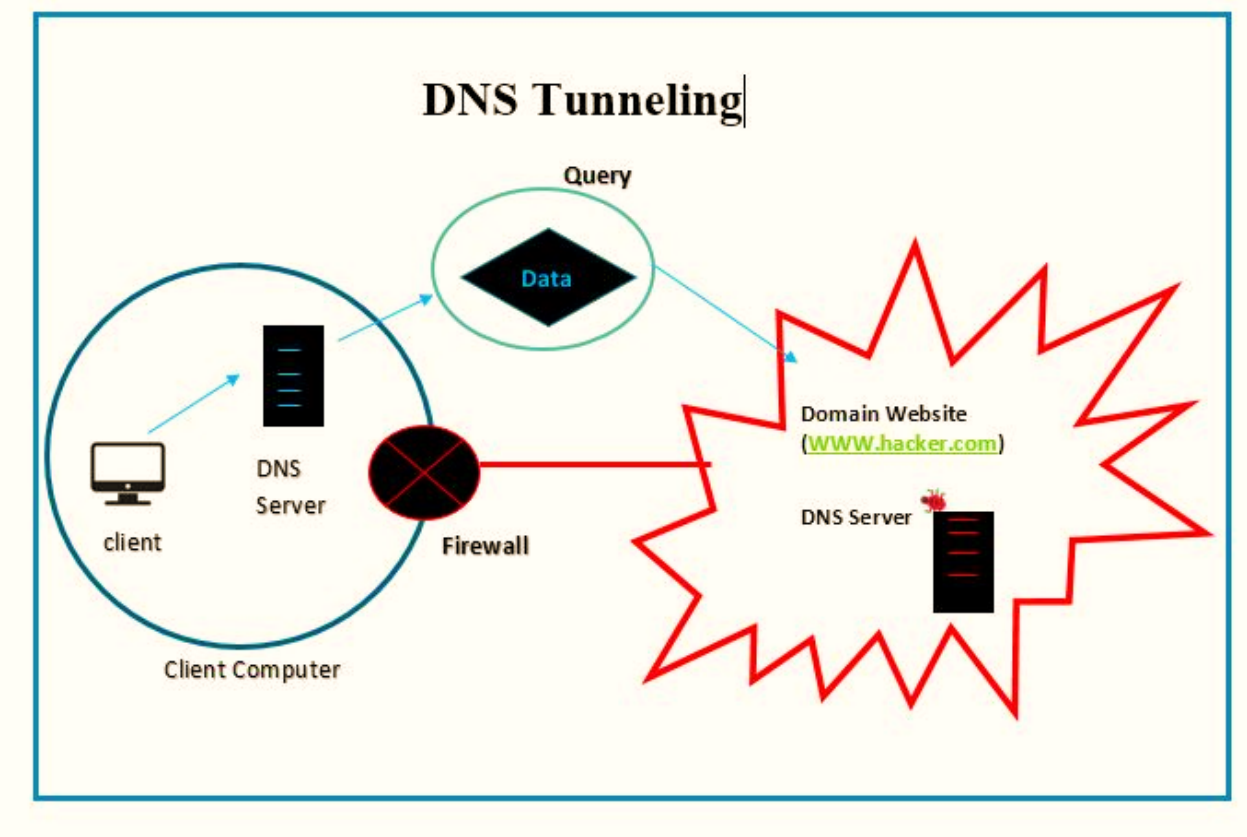


Figure 1: DNS Tunneling

As shown in the figure1, the hacker tries to exfiltrate the client's sensitive information or data. So, the hacker tries to use typical and traditional internet methods to abuse the clients but the problem is in a corporate environment because the client has a firewall that is inspecting and blocking this type of traffic. So, he is going to abuse the DNS protocol to create a covert channel and actually exfiltrate that's sensitive data out. From his perspective, the attackers really need two things to achieve his mission they are Domain Website (www.hackers.com) and needs Local DNS Server. So, when the client tries to go to the attacker's website it'll typically be passed through a local DNS server. DNS server probably doesn't have the IP address of the attacker's website so, it's going to query this website, while the client asks and expects a return to IP address by querying the attacker's website that is simply just a packet that configures and manipulate. The sensitive data lay on the query which is in the hidden part of packets. When the client sends the request (query) to the attacker's website to get his response actually this query will carry the sensitive information of the client and it will be in a place where the attacker can inspect and own it. The attacker has

the client sensitive data by using the DNS tunneling without any firewall's alarm and in an easy way. Actually, the attacker plants his malware in the victim's computers to get his information by querying his website.

DNS tunneling is such a prominent and effective data exfiltration method really for two reasons one because there is much legitimate querying DNS traffic there's actually leaving user's networks so all the user firewalls (DLP's, Proxies, and ID's) typically don't focus or don't monitor DNS traffic and the other thing that users could do with a query is chopped up the sensitive data so it's likely to be caught on it's way out. Once the sensitive data reaches somewhere the attacker can actually control, decrypt it, reconstruct it, and put it back to gather actually piece it to gather the sensitive data.

Chapter III:

Machine Learning

ML is the science of making computers learn this means that, it is a method of teaching computers based on some data and acting like humans by feeding data and information without being explicitly programmed, which is focused on the improvement of computer programs that can reach data and use it to learn for themselves. Nowadays, we need this method that will facilitate our work in using the huge data we extract to study some techniques. [4]

In the DNS detection technique researchers

Researchers

1. Some popular methods of machine learning

1.1 Supervised learning

SL model can predict with the help of labeled datasets and supervised learning is basically of two types (Classification and regression).

Supervised learning is working like a teacher, where the machine will train at a group of features and then from this data that has been trained, we will start to predict. SL gets data output from the previous experience.

For an example, let's assume that we have a basket of vegetable, where we start to train the machine with the three types of vegetables one by one. First, with tomato, potato, and onion for potato it has a rounded shape and brown color, but for tomato red color with a circular shape, and onion has a circular shape with white color. After training the data, let's assume that we give the machine a new vegetable that is tomato, since we already train the machine about some of these items previously, so that it can easily recognize the new vegetables that will added. The machine

should begin to classify the vegetable as its color and shape. Then it should give the true predication which is tomato. Thus, we can say that SL deal with already labeled data make you have a big chance to achieve a correct prediction.

1.2 Unsupervised learning

UL is used for the unlabeled and not classified data, by which unsupervised Learning the algorithm is trained using data that is disabled. While, for unsupervised Learning is basically of two types (Clustering and Association) UL is contradictory to the supervised learning where its work doesn't depend on previous knowledge; it's like a baby of a cat that should learn from itself every new things, which UL doesn't have any trained data like SL, but it demonstrate the new predictions by making its own pattern. That's why, the system does not give the "right answer." The algorithm must learn what it detects. The goal is to search the data and find some structure within. For example, let's take an example of a little boy with his family dogs. He can know and identify these dogs but after few weeks later, a family brings another new dog that is long and tries to play with the little boy. The little boy has not seen this dog earlier. But it knows many features (2 ears, eyes, walking on 4 legs) are like her pet dog. The little boy observes the new animal as a dog. Where we do not learn from unsupervised learning but we will learn from the data (in this case data about the dog.). With supervised learning, the family would have told the little boy that it's a dog.

1.3 Reinforcement learning

The reinforcement learning system is comprised of two main components (agent and environment), where it is used for robotics, gaming, and navigation. The algorithm finds through trial and error using reinforcement learning which actions yield the greatest rewards. Reinforcement learning has three main components: the **AGENT** (decision-maker or the learner),

the **ENVIRONMENT** (everything the decision-maker interacts with), and **ACTIONS** (what the decision-maker can do). Furthermore, the goal is for learner to choose actions that maximize the expected reward over a given amount of time. The decision-maker will reach the goal much faster by following a good policy. Where, learning the best policy is the goal that reinforcement learning must perform.

1.1.1 Classification

Classification is the process or a way of discovering a function or model that helps to divide input data into multiple discrete classes or labels. In addition, Classification is a method that defines whether if an input value can be part of a group that has been defined before or not.

1) Types of classification in supervised learning:

There are many types of supervised learning like Naïve Bayes Classifier, Random Forest Classifier, Decision trees, Support Vector Machines (SVM) and KNN algorithm. In our thesis the focused

a) Naïve Bayes Classifier:

Naive Bayes is a very simple and strong algorithm that is used in predictive modeling. It is a simple probabilistic machine-learning algorithm that can be exploited for a huge range of classification tasks. It is called **Naïve** since it predicts the features that are used in the model.

b) Support Vector Machines (SVM):

SVM is a supervised learning method that observes data and sorts it into one of the two categories. It uses ML technique most based on the classification technique available. It was developed by Vapnik since 1995 and it is based on statistical learning theories. It looks that the cross-validation process would be the best technique for parameter selections of SVMs but few concerns such as running time must not be ignored.

c) KNN Algorithm:

By now, we all know that machine learning models make predication by learning from the past old data. K nearest neighbors is one of simplest supervised machine learning algorithm that classifies new cases based on a similarity measure (e.g., distance functions) and is mostly used for storing all available cases, so we can say it classifies a data point based on how its neighbors are classified. KNN is used in pattern recognition and statistical estimation as a non-parametric technique.

d) Decision Tree:

The Decision Tree is a tree-shaped diagram we can use to compute a course of action DT builds both (classification and regression models) in the form of a tree structure. The decision tree divides the data sets into smaller and smaller subsets while the associated decision tree is gradually improved. A tree consists of decision nodes and leaf nodes that is the final result. A decision node (e.g. Level of Experience) has two or more branches (e.g., Beginner, Experienced, and Intermediate). The Leaf node (e.g., Risk or Not) represents a classification or decision. The main decision node (it is also called Upper decision Node), which agrees with the best predictor, is called the **root node or head node**. DT can deal with both (**categorical data** and **numerical data**).

2) Decision tree in more detail

We study the algorithms of the decision tree in deep detail. From the decision tree, we study (ID3, C4.5, CART) algorithms. Those three algorithms are the most used ones.

I. ID3:

The basic algorithm for building decision trees is called **ID3** that was introduced by Ross Quinlan that uses a top-down greedy search through the space of possible branches with no holdback. ID3 employ (**Entropy** and **Information Gain**) to make a decision tree. On every iteration, the ID3 algorithm chooses the appropriate attribute to split the datasets. As we mentioned before, the ID3 algorithm uses entropy to construct the decision tree.

i- What is entropy?

The question: what is the appropriate entropy that should be used and how to calculate it?

A decision tree includes splitting the data into subsets that have instances with the same values (homogeneous) and built top-down from a root node. The computation of the homogeneity of a sample by ID3 algorithm should use entropy. If the sample is equally separated, then it has an entropy of **one** and if the sample is completely homogeneous the entropy is **zero.** [6]

ii. How to calculate the entropy

From the following example, the concept of entropy will be clear.

Let's take an example of an Assurance Company that would like to know the kind of assurance and its percentage that should be given to his customer using the data that was collected to make these predictions.

So, in the dataset that was collected the features presents are (Level of experience, Marital Status, Small Car, Economical Class, and Risky). This entire feature is called predictors except Risky that is called target.

In this figure below, we will be able to see a sample of this assurance company's dataset with its predication values. From these values, we will be able to calculate entropy and draw the decision tree.

Level of experience	Marital Status	Small Car	Economical Class	Risky?
Beginner	M	Yes	Middle	Yes
Beginner	M	Yes	Low	Yes
Experienced	S	Yes	High	No
Intermediate	S	No	Low	No
Intermediate	S	Yes	High	Yes
Experienced	M	No	High	No
Intermediate	M	Yes	Low	Yes
Intermediate	S	Yes	Low	Yes
Intermediate	M	No	High	No
Intermediate	S	Yes	Middle	Yes
Experienced	S	No	Low	No
Beginner	S	No	High	Yes
Experienced	M	Yes	High	No
Beginner	M	Yes	High	Yes
Beginner	M	No	Middle	No
Beginner	M	No	Low	No

Figure 2: Assurance Company's dataset

As shown in the figures below, let's start discussing the calculation of entropy. Firstly, the dataset should be split at each predicator (at Level of experience, Marital Status, Small Car, Economical Class).

If we split by "Level of experience"

Beginner	Experienced	Intermediate
Yes	No	No
Yes	No	Yes
Yes	No	Yes
Yes	No	Yes
No		No
No		Yes

Splitting on "Level of experience" in an overall entropy of 12 (6+0+6.)

Figure 3:split at the level of experience

If we split by "Small Car"

Yes	No
Yes	No
Yes	No
No	No
Yes	No
Yes	Yes
Yes	No
Yes	No
No	
Yes	

Splitting on "Small Car" results in an overall entropy of 16 (9+7)

Figure 4:split at the Small Car level

If we split by "Economical Class"

Middle	Low	High
Yes	Yes	No
Yes	No	Yes
No	Yes	No
	Yes	No
	No	Yes
	No	No
		Yes

Splitting on "Economical Class" results in an overall entropy of 16 (3+6+7)

Figure 5: split at the Economical Class level

If we split by "Marital Status"

Single	Married
No	Yes
No	Yes
Yes	No
Yes	Yes
Yes	No
No	No
Yes	Yes
	No
	No

Splitting on "Marital Status" in an overall entropy of 16 (7+9.)

Figure 6: split at the Marital Status level

After predicators feature splitting as shown in the four figures (Figure 3, Figure 4, Figure 5, and Figure 5) where we should calculate the entropy for each, which is the summation of Yes and No numbers. As shown in figure 2, the splitting is made at the level of experience, and after that each feature (Beginner, Experienced, and Intermediate) will have the value of Yes or No. At the feature that is called Beginner, which gives 4 Yes, and 2 No, so its entropy is equal to 6.

At the Experienced feature, there are 4 No which means that the entropy will be zero no mixed data observed. However, the last feature Intermediate has 4 Yes and 2 No so entropy is equal to 6. Then, all these numbers that is given by each feature will be the sum of them which gives an entropy equal 12.

These steps will also make the remaining predicator features as shown in figure 4, figure 5, and figure 6, where each figure will give an entropy value, then these values will be compared with each other. Which means that, entropy will be calculated for each predictor feature (Marital Status, Small Car, and Economical Class) so, it for each feature belong to Predicator features that are (single, Married, Middle, Low...). After this comparison step, the lowest value between them will be selected and it will be the root node of the decision tree where this step will repeat till the end of the tree (Leaf node) and the leaf node will be the decision of prediction.

iii. If we want to do an improvement suppose, for example, that the number of those who said yes more or who said no more means if the change is on this level, will the tree shape and the value of the entropy change?

From this enhancement experiment, I conclude that if the number of Yes increases, the number of No decreases, hence the entropy will decrease. Also, the same if the No increases, the Yes decreases.

So, we can say that entropy is how much the data is clean. For example, we have 5 inputs, 4 are yes and 1 is No or reversing the data will be cleaner since there is no diversity.

Hence, if we have 3 Yes, 2 No the entropy will increase because of the diversity, and the data will be less cleaned.

Not matter if the Yes or the No increase the only thing that matters is the increase of inputs that have the same value if Yes or No.

II. C4.5 (Enhancement of ID3):

C4.5 is an addition to the ID3 algorithm which is known as a statistical classifier. By which, this algorithm also uses the concept of information entropy like the ID3 algorithm. C4.5 algorithm divides the attributes into subsets of one class or other, also the field that has the maximum information gain finds the results. Where both are continuous and discrete attributes handle by the Expansion of the ID3 algorithm. For holding the constant fields, the algorithm divided the list into smaller lists with attributes having values above the threshold and having values equal to or less than the threshold values.

i- The C4.5 algorithm calculation

Level of Experience	Economical class	Marital Status	Small Car	Risky?
Intermediate	Middle	M	No	No
Intermediate	Middle	M	Yes	No
Beginner	Middle	M	No	Yes
Experienced	High	M	No	Yes
Experienced	Low	S	No	Yes
Experienced	Low	S	Yes	No
Beginner	Low	S	Yes	Yes
Intermediate	High	M	No	No
Intermediate	Low	S	No	Yes
Experienced	High	S	No	Yes
Intermediate	High	S	Yes	Yes
Beginner	High	M	Yes	Yes
Beginner	Middle	S	No	Yes
Experienced	High	M	Yes	No

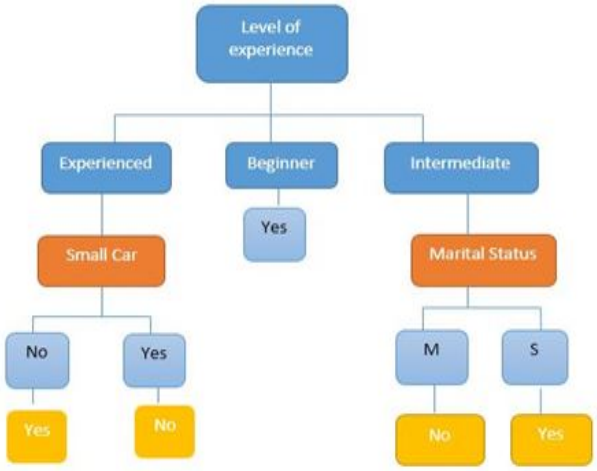


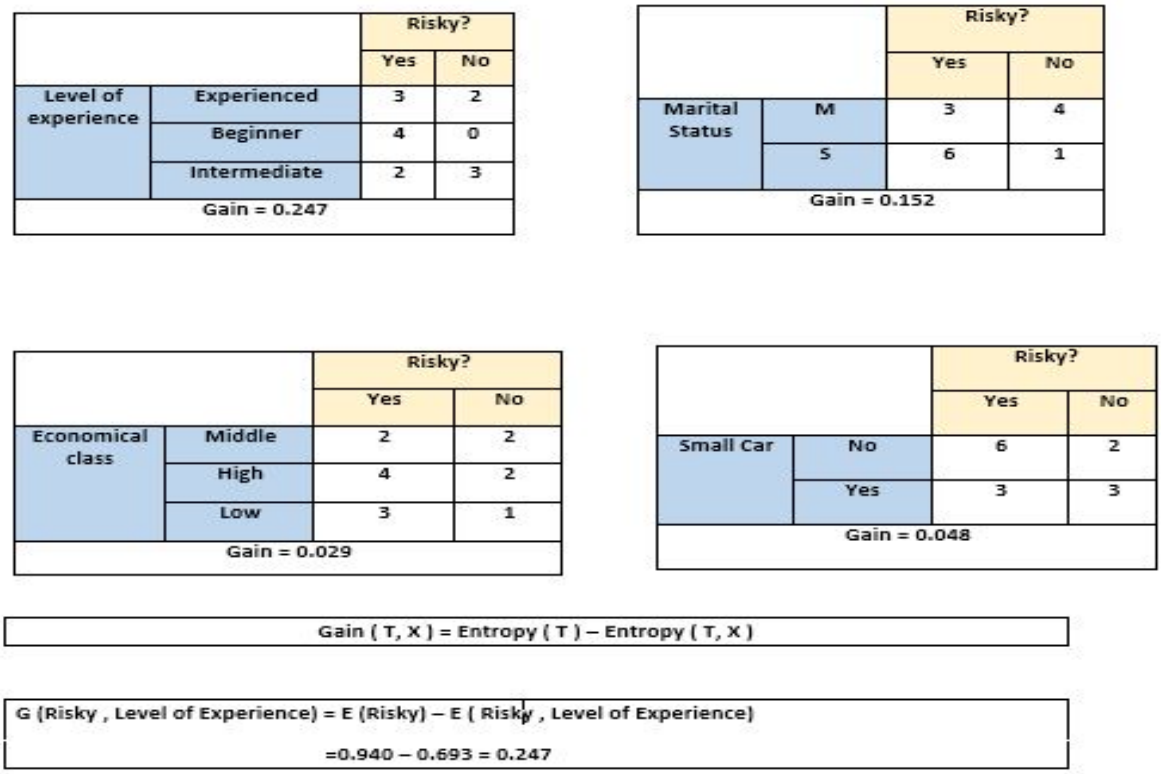
Figure 7: Dataset sample of the C4.5 algorithm

Step 1: Calculate (the target entropy that is Risky).

$$\text{Entropy}(\text{Risky}) = \text{Entropy}(5,9)$$
$$= \text{Entropy}(0.36, 0.64)$$
$$= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64)$$
$$= 0.94$$

Figure 8: Step1/ C4.5

Step 2: We will split the data set used into different attributes. The calculation of entropy for each is done. Relatively added, to get total entropy for the split. The resulting entropy will subtract from the entropy before we make the split. The result will decrease the entropy or Information Gain.



		Risky?	
		Yes	No
Level of experience	Experienced	3	2
	Beginner	4	0
	Intermediate	2	3
Gain = 0.247			

		Risky?	
		Yes	No
Marital Status	M	3	4
	S	6	1
Gain = 0.152			

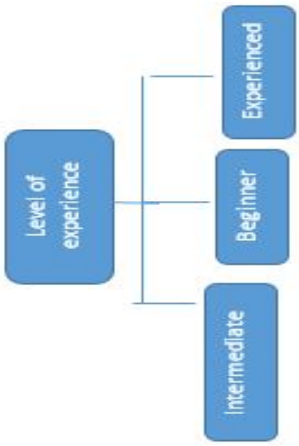
		Risky?	
		Yes	No
Economical class	Middle	2	2
	High	4	2
	Low	3	1
Gain = 0.029			

		Risky?	
		Yes	No
Small Car	No	6	2
	Yes	3	3
Gain = 0.048			

Figure 9: Step 2/ C4.5

Step 3: The attributes should choose the largest information gain as the decision node, divide the dataset by its branches, and repeat the same operation on every branch.

		Risky?	
		Yes	No
Level of experience	Experienced	3	2
	Beginner	4	0
	Intermediate	2	3
Gain = 0.247			



Level of Experience	Economical class	Marital status	Small Car	Risky?
Experienced	High	M	No	Yes
Experienced	Low	S	No	Yes
Experienced	Low	S	Yes	No
Experienced	High	S	No	Yes
Experienced	High	M	Yes	No

Beginner	Middle	M	No	Yes
Beginner	Low	S	Yes	Yes
Beginner	High	M	Yes	Yes
Beginner	Middle	S	No	Yes

Intermediate	Middle	M	No	NO
Intermediate	Middle	M	Yes	No
Intermediate	High	M	No	No
Intermediate	low	S	No	Yes
Intermediate	High	S	Yes	Yes

Figure 10: Step3 / C4.5

Step 4a: The branch that carries entropy that equal to **0** is called a leaf node.

Economical class	Marital status	Small Car	Risky?
Middle	M	No	Yes
Low	S	Yes	Yes
High	S	Yes	Yes
Middle	S	No	Yes

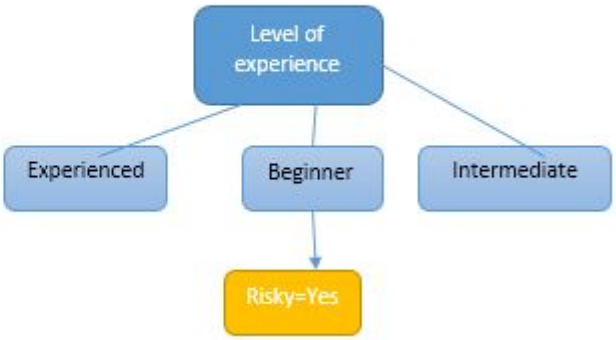


Figure 11: Step 4a / C4.5

Step 4b: The branch that carries entropy more than 0 needs more and more splitting.

Economical class	Marital status	Small Car	Risky?
High	M	No	Yes
Low	S	No	Yes
High	S	No	Yes
Low	S	Yes	No
High	M	Yes	No

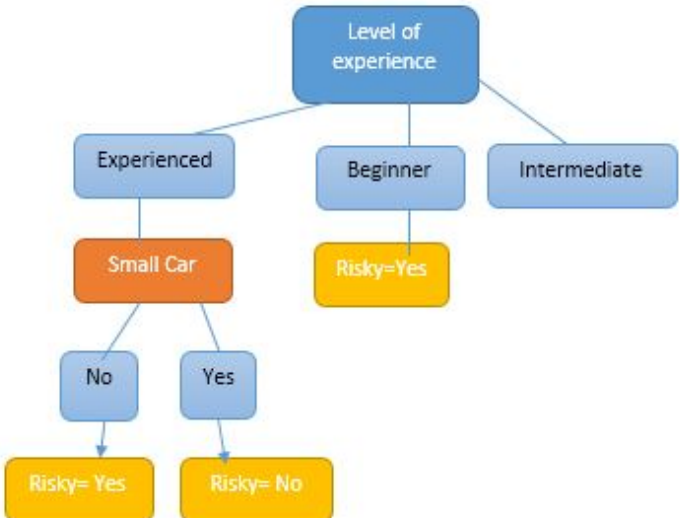


Figure 12: Step 4b / C4.5

From this decision tree, we can easily predict any new customer if we should give him the assurance or not. If it is risky then put 'Yes' if not 'No' using the other features. Where we begin from the root node until the leaf node, then the result of each new customer will be computed using these features so our work will be easier.

III- CART:

Breiman introduced CART in 1984, which can be used for both classifications and regressions predictive modeling problems Construction of the Cart classification tree is based on binary splitting of the attributes. it uses the **Gini index** in selecting the splitting attribute. [7]

Cart is basic for many important machine-learning methods like Decision Tree, Random Forest, and Boosted decision tree.

The most used decision tree algorithms for the large datasets is the **C4.5** and another algorithm called **SPRINT.** But if we want to use small datasets with fewer rows, we should use the CART algorithm that is friendlier with the SCI-KIT learn. Since there are many parameters in Sci-kit learning, that facilities modeling of a decision tree using the CART algorithm.

So, let's test this idea in our dataset (PUF) and test how true it is.

i- What is the Gini index

Gini index (or **Gini** impurity) can measure the probability of a specific variable that is wrongly classified when it is randomly chosen. Note that if all the elements belong to a single class, then it can be called pure

Formulas in Cart:

Gini = 1 – Σ (Pi) ^{2} for i =1 to number of classes

ii- How to calculate the Gini index

The classification tasks in CART will be measured by using the Gini Index. Gini index saves the sum of squared probabilities of each class. [8]

Steps to pick a decision Node:

- ✓ Make a calculation of Gini index for each attribute
- ✓ Make Calculation of the weighted sum of Gini index for the feature
- ✓ Pick the attributes with the lowest Gini index value
- ✓ Repeat steps 1, 2, and 3 until a generalized tree has been constructed

Let's see an example that will discuss what Gini index to be clearer is:

Level of Experience	Economical class	Marital Status	Small Car	Risky?
Experienced	Middle	M	No	No
Experienced	Middle	M	Yes	No
Beginner	Middle	M	No	Yes
Intermediate	High	M	No	Yes
Intermediate	Low	S	No	Yes
Intermediate	Low	S	Yes	No
Beginner	Low	S	Yes	Yes
Experienced	High	M	No	No
Experienced	Low	S	No	Yes
Intermediate	High	S	No	Yes
Experienced	High	S	Yes	Yes
Beginner	High	M	Yes	Yes
Beginner	Middle	S	No	Yes
Intermediate	High	M	Yes	No

Figure 13: Dataset sample

Figure 13 shows a sample of the dataset that will be predicted if the company should give the assurance or not from the Risky feature if it is yes or no, also by using this data above we will be able to calculate the Gini Index for the CART algorithm, and then read the data carefully to be able to achieve the other steps easily.

The splitting that was made firstly at the **Level of Experienced** feature. By which, the **Level of Experienced** is a nominal feature that can be Experienced, Beginner, or Intermediate. I will summarize the final decisions for the **Level of Experience feature**.

Level of Experience:

Feature2:		Yes	No	Total
Level of Experience	Experienced	2	3	5
	Beginner	4	0	4
	Intermediate	3	2	5
	Total	10	4	

Figure 14: Splitting in Level of Experience feature

Gini (risky, Level of Experience = Experienced) = 1- (2/5) ^2 – (3/5) ^2 = 0.48

Gini (risky, Level of Experience = Beginner) = 1- (4/4) ^2 – (0/4) ^2 = 0

Gini (risky, Level of Experience = Intermediate) = 1- (3/5) ^2 – (2/5) ^2 = 0.48

The Gini index of the level of experience (children node) = 5/14 * 0.48 + 4/14*0 +5/14*0.48 = 0.3429

Using the rule of the Gini Index mentioned before we calculated each value of Level of Experience features (Experienced, Beginner, and Intermediate).

Economical Class

The **Economical Class** is similar to the Level of Experience feature that is a nominal feature and it could have 3 different values: **Middle, High, and Low**. Let's briefs decisions for the **Economical Class feature**.

Economical Class	Yes	No	No of instance
Middle	2	2	4
Low	3	1	4
High	4	2	6

Figure 15: Splitting at Economical Class feature

Gini (Economical Class = Middle) = 1 – (2/4) ^2 – (2/4) ^2 = 0.5

Gini (Economical Class = Low) = 1 – (3/4) ^2 – (1/4) ^2 = 0.375

Gini (Economical Class = High) = 1 – (4/6) ^2 – (2/6) ^2 = 0.445

Below is the weighted sum that is calculated for Economical Class feature

Gini (using Economical class) = (4/14)*0.5 + (4/14)*0.375m+ (6/14) *0.445 = 0.142 + 0.107 +0.190 =0.439

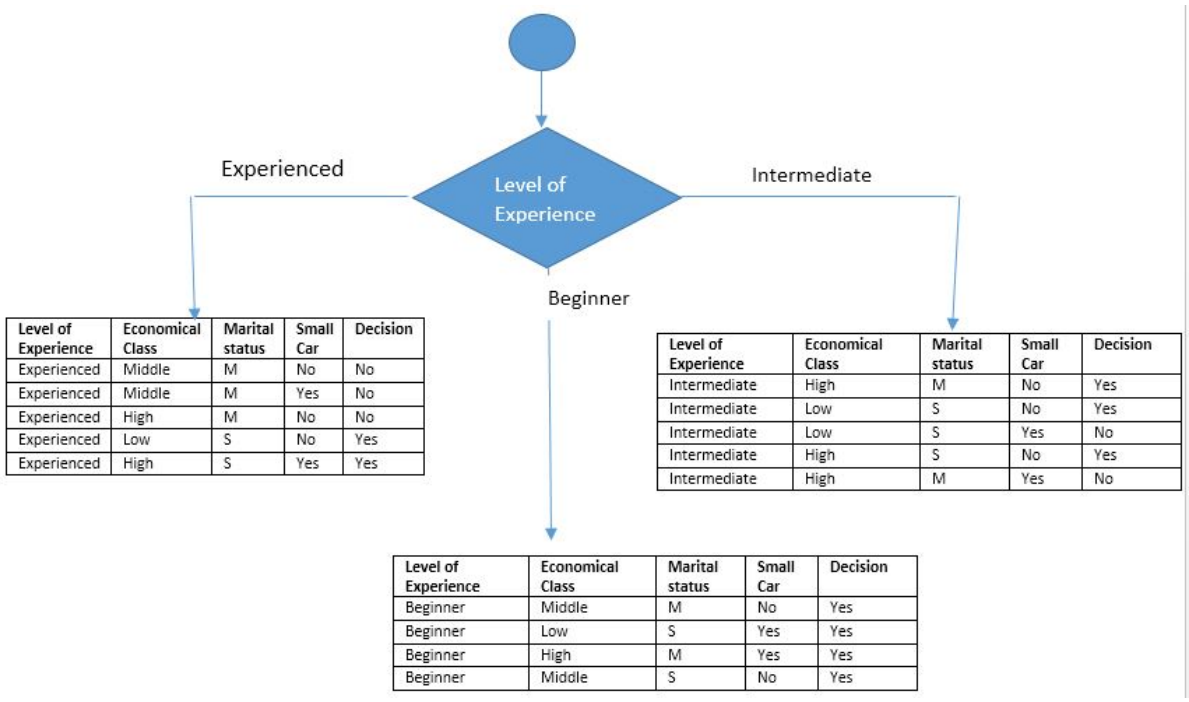
Marital Status and Small Car feature perform the same calculation of weighted sum for the Level of Experience and Economical Class feature. This result is shown in Figure 16.

Time to decide

Features	Gini Index
Level of Experience	0.342
Economical Class	0.439
Marital Status	0.367
Small Car	0.428

Figure 16: Gini indexes' values

As shown in figure 16, we calculate the Gini index values for each feature, the winner will be the Level of Experience feature because it records the lowest cost, that's why we'll locate the Level of Experience feature decision at the top of the tree as a node of the tree.



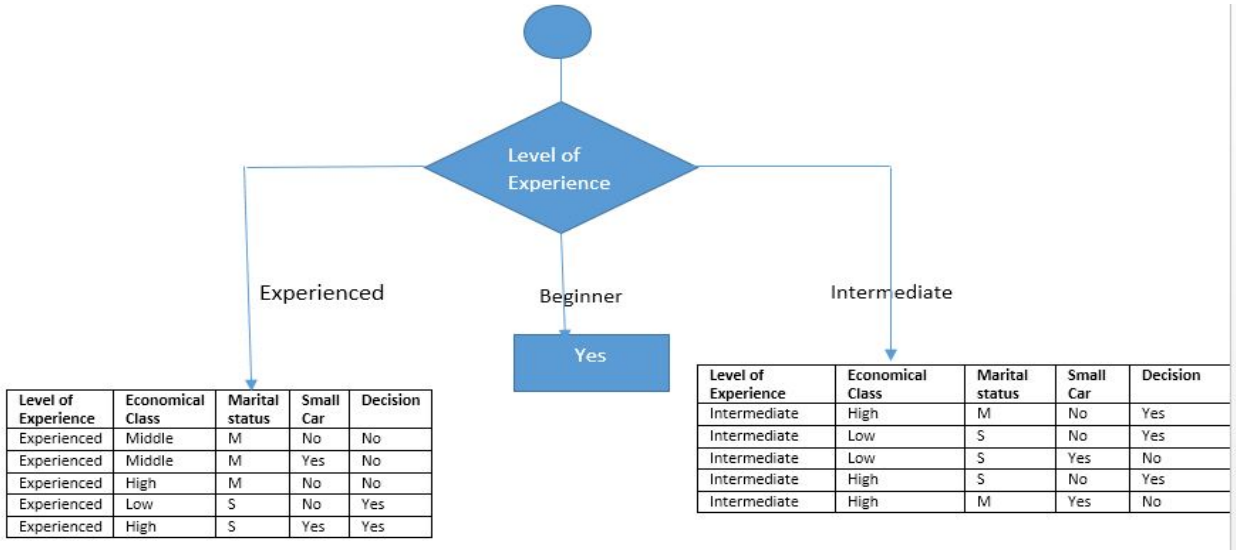
Level of Experience	Economical Class	Marital status	Small Car	Decision
Experienced	Middle	M	No	No
Experienced	Middle	M	Yes	No
Experienced	High	M	No	No
Experienced	Low	S	No	Yes
Experienced	High	S	Yes	Yes

Level of Experience	Economical Class	Marital status	Small Car	Decision
Intermediate	High	M	No	Yes
Intermediate	Low	S	No	Yes
Intermediate	Low	S	Yes	No
Intermediate	High	S	No	Yes
Intermediate	High	M	Yes	No

Level of Experience	Economical Class	Marital status	Small Car	Decision
Beginner	Middle	M	No	Yes
Beginner	Low	S	Yes	Yes
Beginner	High	M	Yes	Yes
Beginner	Middle	S	No	Yes

Figure 17: Level of Experience feature splitting's Decision tree

Figure 17 shows the decision tree of the Level of Experience feature split where each value (Beginner, Intermediate, and Experienced) of the Level of Experience feature is shown in the figure below, as we note each value of the Level of Experience feature is shown with its decision value.



Level of Experience	Economical Class	Marital status	Small Car	Decision
Experienced	Middle	M	No	No
Experienced	Middle	M	Yes	No
Experienced	High	M	No	No
Experienced	Low	S	No	Yes
Experienced	High	S	Yes	Yes

Level of Experience	Economical Class	Marital status	Small Car	Decision
Intermediate	High	M	No	Yes
Intermediate	Low	S	No	Yes
Intermediate	Low	S	Yes	No
Intermediate	High	S	No	Yes
Intermediate	High	M	Yes	No

Figure 18: Tree is over for Beginner Level of Experience leaf

Figure 18 shows the decision tree after using the decision value where Experienced value has 3 No and 2 Yes. While the Beginner has 4 Yes that means that there is no need to make another splitting on it. It gives yes directly. On other hand, Intermediate has 3 Yes and 2 No. So, for Experienced and Intermediate values, there is a need of feature splitting.

We should understand that the sub dataset in the Beginner leaf has only Yes decisions. This means that the Beginner leaf is ended.

Level of Experience	Economical Class	Marital Status	Small Car	Decision
Experienced	Middle	M	No	No
Experienced	Middle	M	Yes	No
Experienced	High	M	No	No
Experienced	Low	S	No	Yes
Experienced	High	S	Yes	Yes

Figure 19: Decision Values of the features

As figure 19 shows, we stratify the same principles to those sub-datasets in the following steps. Focus on the sub dataset for Level of Experience, we need to find the Gini index scores for Economical Class, Small Car, and Marital Status features respectively.

Gini of Economical Class for Level of Experience is experienced feature

Economical Class	Yes	No	Number of instances
Middle	0	2	2
Low	1	0	1
High	1	1	2

Figure 20: Gini of Economical Class for a Level of Experience is an Experienced feature

Gini (Level of Experience = Experienced and Economical Class= Middle) =1- (0/2) ^2 – (2/2) ^2=0

Gini (Level of Experience = Experienced and Economical Class= Low) =1- (1/1) ^2 – (0/1) ^2=0

Gini (Level of Experience = Experienced and Economical Class= High) =1- (1/2) ^2 – (1/2) ^2=0.5

Gini (Level of Experience=Experienced and Economical Class) = (2/5)*0 + (1/5)*0 + (2/5)*0.5 = 0.2

Gini of Marital Status for Level of Experience is Experienced feature

Marital Status	Yes	No	Number of instances
M	0	3	3
S	2	0	2

Figure 21: Gini of Marital Status for an Experienced in the level of Experience feature

Gini (Level of Experience=Experienced and Marital Status=M) =1-(0/3) ^2 - (3/3) ^2=0

Gini (Level of Experience=Experienced and Marital Status=S) =1-(2/2) ^2 - (0/2) ^2=0

Gini (Level of Experience=Experienced and Marital Status) = (3/5) *0 - (2/5) *0=0

Gini of Small Car for an Experienced in the level of Experience feature

Small Car	Yes	No	Number of instances
Yes	1	2	3
No	1	1	2

Figure 22: Gini of Small Car for Level of an Experience is an Experienced feature

Gini (Level of Experience=Experienced and Small Car=No) =1-(1/3) ^2 - (2/3) ^2=0.266

Gini (Level of Experience=Experienced and Small Car=Yes) =1-(1/2) ^2 - (1/2) ^2=0.2

Gini (Level of Experience=Experienced and Small Car) = (3/5) *0.266 - (2/5) *0.2=0.466

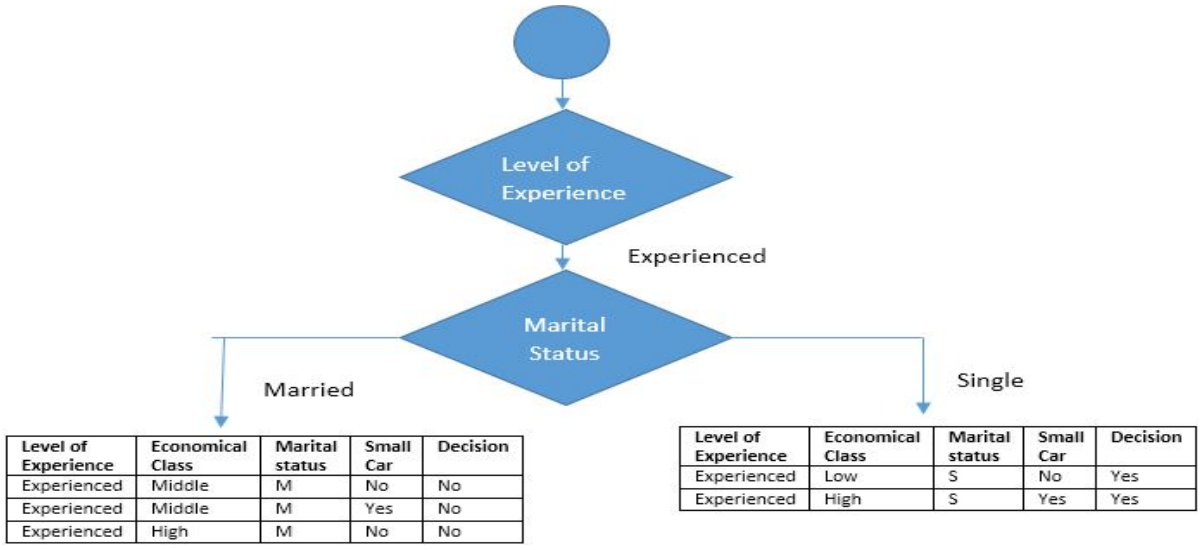
The Decision for Level of Experience is Experienced

We calculated Gini index scores for the features we have when the Level of Experience is Experienced. Marital Status is the winner because it has the lowest value as shown in the figure below.

Features	Gini Index
Economical Class	0.2
Marital Status	0
Small Car	0.466

Figure 23: The decision for Level of Experience is an Experienced feature

We'll put the Marital Status check at the branch of Level of Experience, which is the Experienced feature

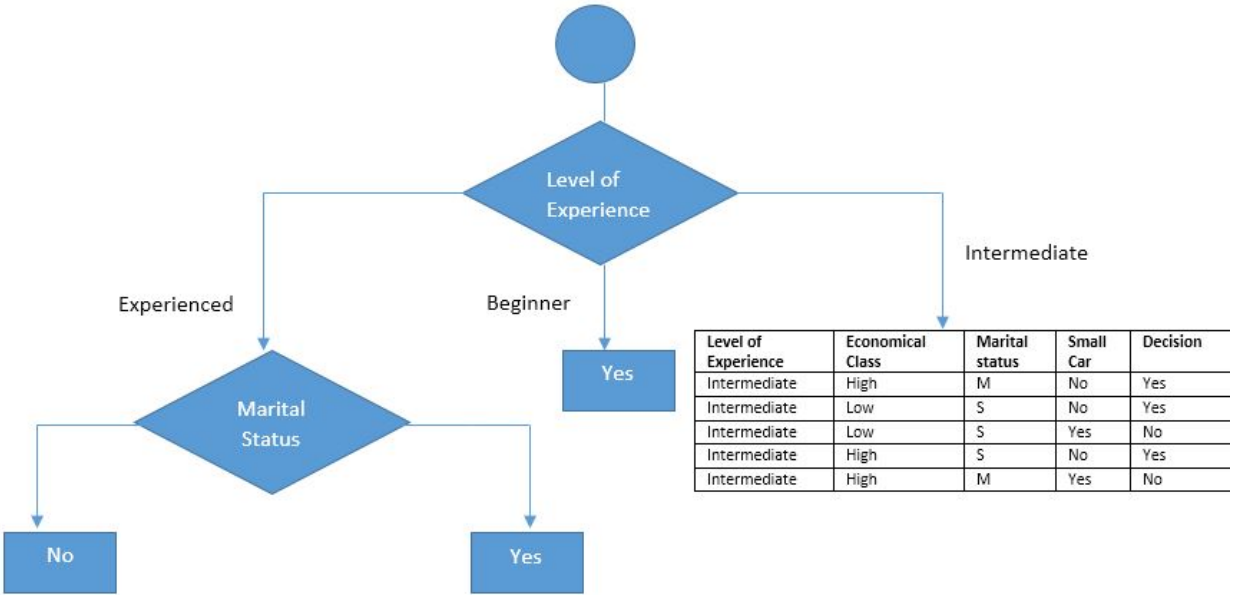


Level of Experience	Economical Class	Marital status	Small Car	Decision
Experienced	Middle	M	No	No
Experienced	Middle	M	Yes	No
Experienced	High	M	No	No

Level of Experience	Economical Class	Marital status	Small Car	Decision
Experienced	Low	S	No	Yes
Experienced	High	S	Yes	Yes

Figure 24: Sub datasets for Marital Status (M and S)

As seen in figure 24, the decision is always No for married Marital Status, and Level of Experience is Experienced. On the other hand, the decision will always be Yes for single Marital status, and the Level of Experience is Experienced. This branch is over.



Level of Experience	Economical Class	Marital status	Small Car	Decision
Intermediate	High	M	No	Yes
Intermediate	Low	S	No	Yes
Intermediate	Low	S	Yes	No
Intermediate	High	S	No	Yes
Intermediate	High	M	Yes	No

Figure 25: Decisions for Marital Status (M or S)

Now, focus on the Level of Experience when it is Intermediate now.

The Level of Experience is Intermediate

Level of Experience	Economical Class	Marital Status	Small Car	Decision
Intermediate	High	M	No	Yes
Intermediate	Low	S	No	Yes
Intermediate	Low	S	Yes	No
Intermediate	High	S	No	Yes
Intermediate	High	M	Yes	No

Figure 26: The Level of Experience is Intermediate

When the Level of Experience is Intermediate, we'll calculate Gini index scores for economical Class, Marital Status, and Small Car features.

Gini of Economical Class for the Level of Experience is Intermediate

Economical Class	Yes	No	Number of instances
Low	1	1	2
High	2	1	3

Figure 27: Gini of Economical Class for the Level of Experience is Intermediate

Gini (Level of Experience=Intermediate and Economical Class=Low) =1-(1/2) ^2 - (1/2) ^2=0.5

Gini (Level of Experience= Intermediate and Economical Class=high) =1-(2/3) ^2 - (1/3) ^2=0.444

Gini (Level of Experience= Intermediate and Economical Class) = (2/5) *0.5 + (3/5) *0.444=0.466

Gini of Marital Status for the Level of Experience is Intermediate

Marital Status	Yes	No	Number of instances
Married	1	1	2
Single	2	1	3

Figure 28: Gini of Marital Status for the Level of Experience is Intermediate

Gini (Level of Experience=Intermediate and Marital Status=Low) =1-(1/2) ^2 - (1/2) ^2=0.5

Gini (Level of Experience= Intermediate and Marital Status=high) =1-(2/3) ^2 - (1/3) ^2=0.444

Gini (Level of Experience= Intermediate and Marital Status) = (2/5) *0.5 + (3/5) *0.444=0.466

Gini of Small Car for the Level of Experience is Intermediate

Small Car	Yes	No	Number of instances
No	3	0	3
Yes	0	2	2

Figure 29: Gini of Small Car for the Level of Experience is Intermediate

Gini (Level of Experience=Intermediate and Small Car=No) =1-(3/3) ^2 - (0/3) ^2=0

Gini (Level of Experience= Intermediate and Small Car=Yes) =1-(0/2) ^2 - (2/2) ^2=0

Gini (Level of Experience= Intermediate and Small Car) = (3/5) *0 + (2/5) *0=0

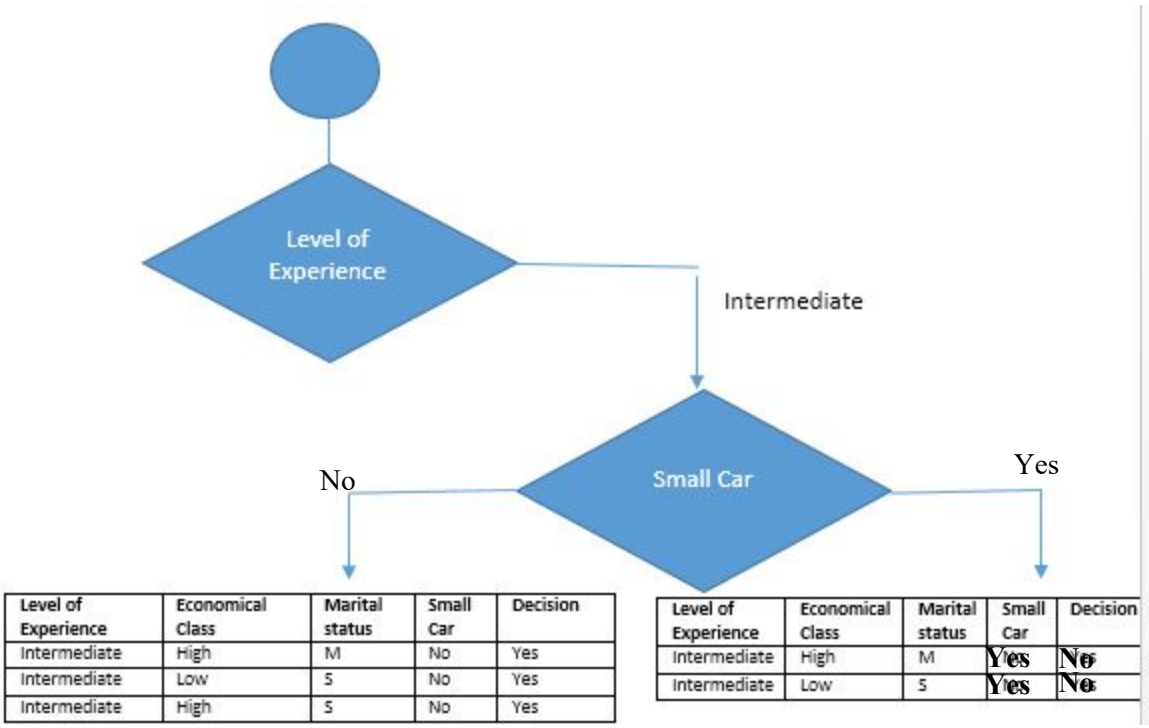
The Decision for the Level of Experience is Intermediate

The winner for the Small Car feature for Level of Experience is Intermediate because it has the minimum Gini index score in features.

Feature	Gini index
Temperature	0.466
Humidity	0.466
Wind	0

Figure 30: Decision for Level of Experience is Intermediate

Put the Small Car feature for the Level of Experience as the Intermediate branch and monitor the new sub data sets.



Level of Experience	Economical Class	Marital status	Small Car	Decision
Intermediate	High	M	No	Yes
Intermediate	Low	S	No	Yes
Intermediate	High	S	No	Yes

Level of Experience	Economical Class	Marital status	Small Car	Decision
Intermediate	High	M	Yes	No
Intermediate	Low	S	Yes	No

Figure 31: Sub data sets for Small Car (Yes and No) and for Intermediate Level of Experience

As seen in figure 31, the decision is always gives Yes when the Small Car is 'No'. On the other hand, the decision is always gives No if the Small Car is "Yes". This means that, this branch is get over or ended.

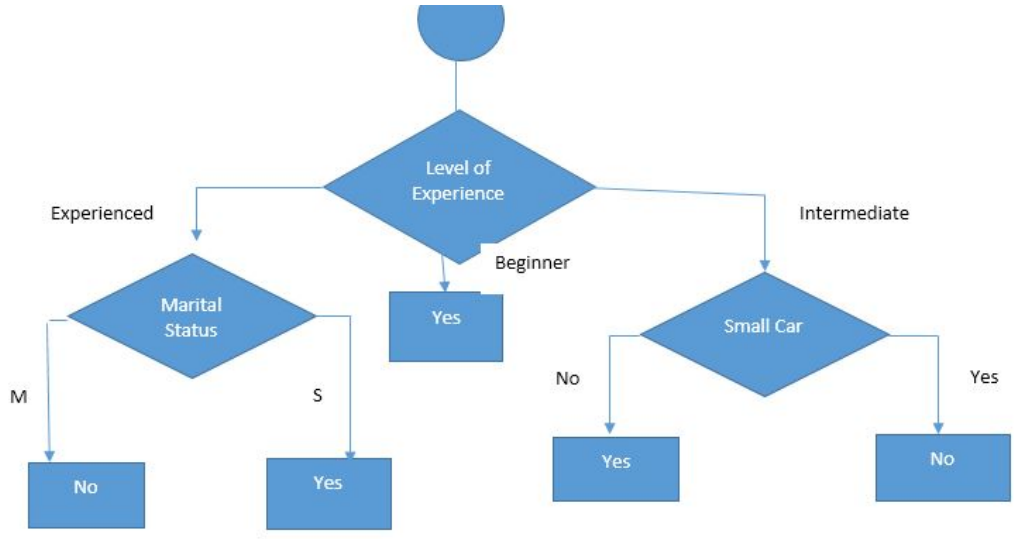


Figure 32: Final form of the decision tree built by CART algorithm

Then the decision tree building is over and Decision trees have been built by hand BTW, we might understand that we've created the same tree in the example of ID3 algorithm, this does not mean that ID3 and CART algorithms pro have the same trees always, we are just lucky. Finally, I believe that CART is easier than ID3 and C4.5 that obtained in the testation operation.

3) Regression

The process of finding a model that predicts the continuous value based on its input variables is called regression, furthermore in regression problems, the goal is mathematically estimated a mapping Regression can be used in all supervised learning types, but they use other ways than that used in classification, where Regression has given targets values between an interval that is specified by the dataset maker. So let's take an example of **DNS Tunneling** dataset with targeted values that have an interval begins with 0 and ended at 10 including the decimal values, where each row in the dataset will take its value from this interval by knowing the affections of each feature on the target value that results during this, then we can assign the value to each row if there anomalies or not but in regression target values. So, it is not 0 or 1 it may be 0.1 or 2.5 it depends on how much it affects the results, while classification gave the target value true or not, 0 or 1 there is no existence of interval in the classification that is supervised learning type (anomalies get 1, while no anomalies get 0 for the example we mentioned).

1.2.1 Unsupervised Learning

Here we chose the Neural Network to compute them, and compare them at last with DT algorithms

Neural Network (NN)

Deep learning make the use of an Artificial NN which behaves similarly to the NN in our brain, which starts its functions when some input data are fed to it, then this data will proceed using the perceptron to produce the desired output. Also, we can use NN to understand and translate the inputted data to reach the desired outputted.

Moreover, we can say that the deep learning using AI, where it's a group of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the behavior of the human brain operation in order to solve the complex data-driven problems. In addition, the artificial NN concept are gotten from the way neurons of the human brain function together to understand inputs from human senses and human biology.

Neural networks are approaches used in machine learning algorithms and just one of the tools. The neural network is used as a part of many different machines learning algorithms to convert complicated data inputs into a space that computers can understand it.

Neural networks are used or applied to many real-life problems today including spam email filtering, image recognition, finance, speech, and medical diagnosis, to name a few.

Three types of layers:

- Input layer
- Hidden layers (processing layer)
- Decision-making.

Each layer is composed of nodes that performs basic processing (applying activation function)

After applying the activation function, the results given are feeds to the forward layer.

Each node in the layer is connected to all the nodes in the layer ahead. By which, the information flows between layers to reach the decision-making layer where some nodes affect the results more than other nodes, where each connection is assigned a weight that resembles the participation of that connection in the decision-making process. For instance, to identify a friend's eyes (visual input) is more important than the sense of touch thus visual Input weight will be higher than other senses.

I. Training a neural network

i. Object

The training phase's goal is to find the best weights of each layer and node. For now, keep that in mind while we introduce some definitions.

ii- Process of training and hyperparameters explanation

If we want to do an improvement suppose, for example that the number of those who said yes more or that who said no more this means if the change is on this level the tree shape and the value of the training phase's goal is to find the best weights of each layer and node. For now, keep that in mind while we introduce some definitions.

In the training of the neural network, the algorithm divides the training data for small pieces, that maximizes technique the learning from each subsample these subsamples are called batches. Then after the algorithm finishes running on all the data and completing the full cycle, then the algorithm repeats the process, where each cycle of training is called an epoch.

Usually, if the data is too complicated, the batch size must be small and the number of epochs will be high, after each epoch, an evaluation process takes place to check if the network is learning, then the evaluation calculates the number of errors in the result where this is called loss function. The training phase tries to minimize the loss function, as well as the loss is decreasing, then the algorithm is performing well, and the only problem that comes to the surface is that when should the training phase ends up . In other words, how many epoch are needed? Overfitting references are too much training earlier, then the object of training is to find the best set of weights of the network initially the weights will be random then with each iteration (epoch) the weights will be updated to minimize the loss. Where this arouses a question of what are the changes in weights allowed per each epoch, which is referred by the learning rate. Finally, I will not dive into details just keep in mind that the best approach for learning rate value is to be adaptive (change as the training advances)

The optimizer function is responsible for the loss rate. Where some optimizers are faster and better than other Stochastic Gradient Descent is an example of a good optimizer, Adam optimizer is

another powerful optimizer, where it is the first one that had published in 2014. We can say that, the NN is an adaptive learning rate optimizer algorithm designed especially for training deep NN. Furthermore, Adam optimizer can consider as a combination of RMSprop and Stochastic Gradient Descent with momentum, where it takes the advantage of momentum by using the moving average of the gradient instead of the gradient itself like SGD with momentum and it uses the squared gradients to scale the learning rate like RMSprop. For more information about Adam please refer to.

Despite, another hyperparameter is the activation function, where its function has two rules the first is to control the output, In other words, it limits the output between a minimum and a maximum of each layer some activation functions then the other rule of the activation function is calculating the loss value.

A question pops to our minds is how to select the best loss function, optimizer, batch size, number of epochs these parameters is called the hyperparameters the human must select these Luckily in selecting the loss function there is a general approach for each kind of problem for example, the binary classification and the loss function used is binary cross-entropy. Other hyperparameters are tricky so we will would try and error to reach the best hyperparameters for each problem.

In our work, we run a series for each hyperparameter for the optimizer function we only tested the best optimizers known for our problem Adam and SGD, we can say that Adam optimizer outperformed Stochastic Gradient Descent. Concerning network architecture, where we started to test the small and then increased the number of layers. As we go deeper, the results become more bad, then we testes all known activation functions, also for our case this is the best network architecture. There are four layers in our network, the Input layers are two hidden layers and output layer, such that the input layer has 12 nodes one node for each attribute in the dataset. The second layer contains 24 nodes the activation function for the two layers is selu. Then we have a 6 nodes layer the activation function is a soft plus the output layer has one node the activation function is sigmoid.

II- Proposed Work

Our work is categorized into two groups:

I. Working on the network

In the neural network section, we started by selecting the best hyperparameter for our problem luckily researches do not need to test everything, some problems have common design patterns. For example, the loss function for a binary classification problem is binary cross-entropy; the last layer activation function is sigmoid then concerning the optimizer, we only tested the best optimizers available (Adam, and SGD). After that, we started by selecting a network structure that best fits our problem (we select a fully connected network with enabled backward propagation). Then we started testing the number of layers and number of nodes per layer as well as selecting the activation function for each layer, then selecting the activation function is a process based on the activation function nature (nonlinear) and the output values as well as the derivative of the activation function (for calculating the loss function). After a serious of testing, we reached an optimal structure.

There are four layers in our network, an Input layer, two are hidden layers and two output layer. Where the input layer has 12 nodes, one node for each attribute in the dataset, and the second layer contains 24 nodes where the activation function for the two layers is selu. Then, we have a 6 nodes layer where the activation function is softplus. The output layer has one node the activation function is sigmoid.

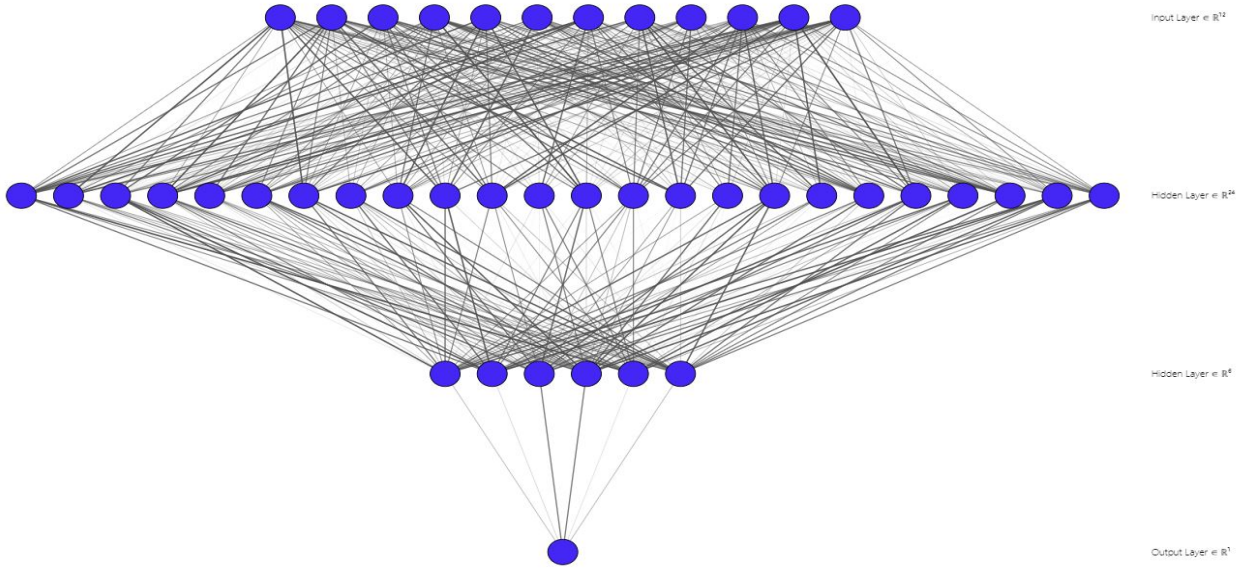


Figure 33: Network architecture

II. Working on the dataset

a) Data distribution problems and solutions

Different scales need normalization or scaling.

Machine learning algorithms aim to map input data to an output result. Moreover, the algorithm must improve the performance to be categorized in the machine learning zone.

Machine learning algorithms tend to rely heavily on the data provided, but the Neural networks have no exceptions. In the case of numeric features, suppose that the feature A mean is 100 and feature B means is 1. While adding one unit to the mean of feature A is not significant, while the addition one to the mean of attribute B then its mean will be doubled, but another problem is that when there is a big difference between the attributes Preprocessing the input data decreases the complexity of the problem thus decrease the computation needed and it amplifies the performance.

Suppose that class A has two instances:

- {10, 50, A}
- {100, 51, A}

This problem adds complexity to the training process and in many cases; the network fails to learn which leads to the poor performance of the algorithm.

A common approach is to normalize the data, which is in definition scaling the mean and the range of vector (instance of data).

Standardization is another solution which involves centric the data around the mean with a threshold of allowance to be different from the mean (standard deviation). In the above example, the standardization process will center the data on the mean.

- {20,50,A}
- {70,51,A}

As an example, implies the standardization targets attributes meanwhile the normalization targets the instances, the normalization technique modifies the distribution of the instances or the shape of the data.

III. Imbalanced dataset:

An Imbalanced dataset is a dataset that has non-even occurrences classes. Then, this problem has a strong impact on many machine learning algorithms. Fortunately, deep learning is not affected by that problem, then we performed a simple yet effective balancing technique, which implies to add weight for instances of the submissive class results. This implies that, in the training phase there is an increase of the performance of the deep learning, while in the testing phase this decreases the performance significantly. Back in figure 49, we can see the results.

Chapter IV:

Dealing with datasets

To deal with the new dataset, we should know many concepts that will start the definition of the CSV files with which are the main concepts in the domain of datasets. By which, dealing with CSV files will facilitate the use of datasets.

I. CSV files

I would like to show the details of the CSV files in the paragraphs below. Firstly, we can say that the first line of the CSV file contains the table column labels, when each of the subsequent lines represents a row of the table, commas divide each cell in a row, which is where the name comes from.

This below is an example of a CSV file:

This example from the dataset we use that has many columns or features. The dataset we have contains five rows including the header row.

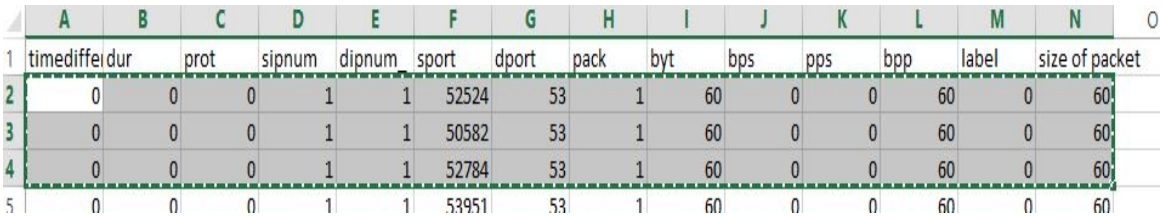
' timedifference ', ' dur', 'prot ', ' sipnum ' dipnum' , 'sport', 'dport', 'pack','byt','bps','pps','bpp'and 'label'.

'timedifference ', ' dur', 'prot ', ' sipnum ' dipnum' , 'sport', 'dport', 'pack','byt','bps','pps','bpp','label'

0,0,0,1,1,52524,53,1,60,0,**0**,60

0,0,0,1,1,50582,53,1,60,0,**0**,60

0,0,0,1,1,52784,53,1,60,0,**0**,60



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	timediffe	dur	prot	sipnum	dipnum	sport	dport	pack	byt	bps	pps	bpp	label	size of packet	
2	0	0	0	1	1	52524	53	1	60	0	0	60	0	60	
3	0	0	0	1	1	50582	53	1	60	0	0	60	0	60	
4	0	0	0	1	1	52784	53	1	60	0	0	60	0	60	
5	0	0	0	1	1	53951	53	1	60	0	0	60	0	60	

Figure 34: PUF dataset in the spreadsheet

Since a CSV file is just a text file, we can create one in almost any text editor. We can also export CSV files from any spreadsheet program, such as (Microsoft Word, OpenOffice Calc, or Google Sheets).

a) How to open the CSV file

The Opening of the CSV file is very simple and friendly. Just choose **File -> Open** and then select the CSV file we want.

We'll go into more detail on opening a CSV file in Microsoft Excel in the paragraph below.

b) Open a CSV file Microsoft Excel

You should first download excel if it is not available in your PC, then just double-click on CSV file to open it in Excel. After that, we may see a prompt asking about the program that we want to use or it should open with, after that choose Microsoft Excel.

But, if the excel is already installed, we can select File **> Open** and choose the CSV file wanted. Microsoft Excel will show us the data in a new workbook. For more information, we also can import CSV files using google sheets and more other ways.

We get our dataset from its source, then we open it in Ms excel. After that, we can edit or update the dataset (add, remove, and update), also we can make this step manually or by write a python code

HINT:

If we have a few numbers of rows in datasets we can analyze data using excel and update it manually, while if we have a large number of the row we can use Jupyter notebook in anaconda navigator by written any programming language to use it.

II. The LABELING of the dataset and ITS MODIFICATION

In this section, we should talk about the changes that had done in datasets, which is presented here with discussions. We use in our thesis a dataset called PUF that is related to the thesis subject (DNS tunneling detection). In addition, some Updates have been done on the features of PUF.

The PUF dataset included DNS flow only which are transferred from the internal network towards the Internet using Destination port 53. However, the log entries flows with Destination port 53 only the status of an action is (block, blocked, close, and deny).

The collection of flows still for three days and amounting to 298463 flows that subjected to the following modifications:

- The **et(end time)** and **st(start time)** both deleted because it is not important to take it in our case.
- The features sip (Source IP) and dip (Destination IP) should maintain the privacy of users and prevent publicity. Besides, Individual **Source IP** and **Destination IP** do not play many roles in detecting anomalies or attacks, so it is not important in our case. However, statistical information contained in these attributes can be important like **entropy this concept will discuss deeply below**, deviation, etc. Moreover, Source IP and Destination IP are categorical attributes that ae generally required to be converted into a numerical form to use it in the detection process. Thus, in our dataset, both Source IP and Destination IP are encoded into numerical form, and then converted to numeric values.

2. Labeling of the dataset (before the modification):

- dur: flow duration.
- prot: Protocol of the flow.

- sipnum: Source IP (encoded in numerical form).
- dipnum: Destination IP (encoded in numerical form).
- sport: Source Port.
- dport: Destination Port.
- pack: number of packets.
- byt: number of bytes.
- bps: number of bits per second.
- pps: number of packets per second.
- bpp: number of bytes per packet.
- label: Flow Label that 0 for (benign) or 1 for (anomalous)

After modification, we add time difference (merge between the start and end time).

- The changes to numerical values are made at the values of the (prot) feature that can be changed (zero for UDP, and one for TCP).
- Also, we have start time and end time features all of the instances that have the same start time and end time expect 3287 instances or rows so we merged these two features in one feature named (time difference).
- The feature that is named (Dport) that has repeated value 1500 we scale down it and subtract (1500-1499) so we will have 1 instead of 1500.

Chapter V:

Experimental setup

In this section, we calculate the values for both DT algorithms and NN machine learning methods using python language. First, we install an Anaconda Navigator that is known as a desktop graphical user interface (GUI). Despite, some users may ask why we use the Anaconda Navigator. So, we can say that we use to launch applications, manage the conda packages, environments, and channels without using command-line commands including Jupyter Notebook. Jupyter Notebook is a tool primarily used with IPython that creates notebooks in the browser.

Second, we use the Jupyter notebook to write the python code for each algorithm or method we used to study DNS Tunneling Detection and helped in.

Then, for the anaconda navigator, we install Sci-kit learn that contains a lot of efficient tools for the use of machine learning. Sklearn library contains statistical modeling including classification, regression, and clustering.

Finally, to use Sci-kit learn in anaconda (Jupyter notebook) we write python code in Jupyter to install Sci-kit learn on it and its package.

conda install sci-kit learn

i. DT classification in python implementation

In this section, we start with learning the Decision Tree Classification, and then make the measurements of attribute selection, also we should be aware about how to build and optimize Decision Tree Classifiers using the Python Sci-kit learn package.

As mentioned before, we use three decision tree algorithms. For each algorithm, we compute its Evaluation vales and its Basic values. This has done by writing a python code in anaconda navigator to use these values in our thesis and the figures below will show the code for each of these algorithms. The dataset name is 'PUF' that we use to implement the DT algorithms is in python code that is 'CSV' extension. That's why, we implement for each algorithm separately in files using anaconda navigator and we assign for each file names that are similar to the algorithm name, for example for the ID3 algorithm, we assign ID3 as its filename.

The following is the steps of how to implement a decision tree in python language using anaconda navigator:

1. Importing of the Required Libraries (library related to DT implementation)

Firstly, we start with loading the required libraries as shown in the figure below (Figure 35).

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
import pandas
from sklearn import tree
from sklearn.metrics import confusion_matrix
import pandas
import seaborn as sns
```

Figure 35: Import libraries (ID3, Cart)

2. Loading Data

We load the required PUF dataset using pandas' read CSV function. Refer to figure below that is written in python code.

```
df = pandas.read_csv("Downloads/puftimedif_update.csv")
```

```
df.head()
```

	timedifference	dur	prot	sipnum	dipnum_	sport	dport	pack	byt	bps	pps	bpp	label
0	0	0.0	0	1	1	52524	53	1	60	0	0	60	0
1	0	0.0	0	1	1	50582	53	1	60	0	0	60	0
2	0	0.0	0	1	1	52784	53	1	60	0	0	60	0
3	0	0.0	0	1	1	53951	53	1	60	0	0	60	0
4	0	0.0	0	1	1	56270	53	1	60	0	0	60	0

```
df.shape
```

```
(298463, 13)
```

Figure 36: Import Data (ID3, Cart)

3. Feature Selection

In this step, we need to divide the given columns below into two types of variables: **dependent** variables (also they are called the **Target variable**) and **independent variable** (it is also called **Feature variables**).

```
df["label"].unique().tolist()
```

```
[0, 1]
```

```
X=df.drop('label',1)
Y=df['label']
```

Figure 37: Feature Selection (ID3, Cart)

4. Splitting Data

To understand the model performance concept, a good strategy is to divide the dataset into a training set and a test set. So, we decide to split the dataset by using a function called train_test_split (). We decided to pass to this function three parameters the (**features, the target, and the test_set size).**

```
X_train, X_test, y_train, y_test = train_test_split(X, Y,test_size=0.2)
```

Figure 38: Splitting Data (ID3, Cart)

This step before is common between all decision tree algorithms that were used in our thesis. This python code helping in the calculation of decision tree algorithms.

Let's start with the **ID3 algorithm** from step 5:

5. Building Decision Tree Model (ID3 algorithm)

We make a Decision Tree Model using Scikit-learn as shown in the figure below.

```
parameters = {'max_depth':range(3,25),
              'class_weight' : ["balanced"],
              'criterion':["entropy"],
              'splitter':["best", "random"]
             }

clf = GridSearchCV(tree.DecisionTreeClassifier(),parameters,cv = 3, n_jobs=4,verbose=1)
clf.fit(X_train,y_train)
tree_model = clf.best_estimator_
```

```
Fitting 3 folds for each of 44 candidates, totalling 132 fits
```

```
[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:   16.5s
[Parallel(n_jobs=4)]: Done 132 out of 132 | elapsed:   48.3s finished
```

Figure 39: Building DT (ID3)

6. Basic Values (ID3 algorithm)

We will see in the below figure the code of the Confusion matrix that is used to calculate Basic values.

```
X_pred = tree_model.predict(X_test)
y_true = y_train
cf_matrix = confusion_matrix(X_pred, y_test)
cm_df = pandas.DataFrame(cf_matrix)
sns.heatmap(cm_df, annot=True, fmt='g')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2679f8dd148>
```


```
def print_confusion_matrix(y_test,pred):
    cm = confusion_matrix(y_test,pred)
    print('True positive = ', cm[0][0])
    print('False negative = ', cm[0][1])
    print('False positive = ', cm[1][0])
    print('True negative = ', cm[1][1])
```

```
#red=clf.predict(X_test)
print_confusion_matrix(y_test,pred)
```

```
True positive =  51000
False negative =  1105
False positive =  508
True negative =  7080
```

Figure 40: Confusion matrix values (ID3)

7. Evaluating Model(ID3 algorithm)

In this step, we start with thinking of how accurately the classifier or model can predict the type of cultivars.

The Accuracy that may be computed by making a comparison between actual test set values and predicted values

```
from sklearn import metrics
pred=clf.predict(X_test)
print("precision",metrics.precision_score(y_test,y_pred=pred))
print("recall",metrics.recall_score(y_test,y_pred=pred))
print("accuracy:",metrics.accuracy_score(y_test,y_pred=pred))
print("f1_score", metrics.f1_score(y_test,y_pred=pred))
print(metrics.classification_report(y_test,y_pred=pred))
```

```
precision 0.8649969456322542
recall 0.9330521876647337
accuracy: 0.9729784061782789
f1_score 0.8977366385595638
              precision    recall  f1-score   support

           0       0.99      0.98      0.98     52105
           1       0.86      0.93      0.90      7588

    accuracy                           0.97     59693
   macro avg       0.93      0.96      0.94     59693
weighted avg       0.97      0.97      0.97     59693
```

Figure 41: Evaluation Values (ID3)

Let's write code to **CART algorithm** calculation in the step 5 (Building Decision Tree Model)

```
parameters = {'max_depth':range(3,25),
              'class_weight' : ["balanced"],
              'criterion':["gini"],
              'splitter':["best", "random"]
             }


clf = GridSearchCV(tree.DecisionTreeClassifier(),parameters,cv = 3, n_jobs=4,verbose=1)
clf.fit(X_train,y_train)
tree_model = clf.best_estimator_
```

```
Fitting 3 folds for each of 44 candidates, totalling 132 fits
[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:   17.0s
[Parallel(n_jobs=4)]: Done 132 out of 132 | elapsed:  1.0min finished
```

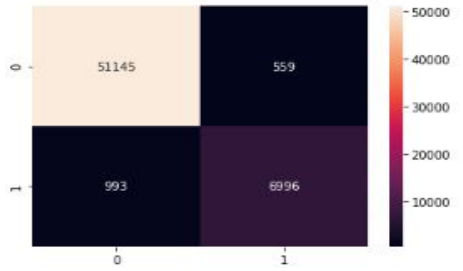
Figure 42: Building DT (CART

Basic Values (Cart algorithm)

We will see in the below figure the code of the Confusion matrix that is used to calculate Basic values.

```
X_pred = tree_model.predict(X_test)
y_true = y_train
cf_matrix = confusion_matrix(X_pred, y_test)
cm_df = pandas.DataFrame(cf_matrix)
sns.heatmap(cm_df, annot=True, fmt='g')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x12a24372c48>
```




```
def print_confusion_matrix(y_test,pred):
    cm = confusion_matrix(y_test,pred)
    print('True positive = ', cm[0][0])
    print('False negative = ', cm[0][1])
    print('False positive = ', cm[1][0])
    print('True negative = ', cm[1][1])
```

```
#red=clf.predict(X_test)
print_confusion_matrix(y_test,pred)
```

```
True positive =  51145
False negative =  993
False positive =  559
True negative =  6996
```

Figure 43: Confusion Matrix (CART)

Evaluating Model (Cart algorithm)

Here in this figure below the Evaluation Values calculated for the Cart algorithm.

```
from sklearn import metrics
pred=clf.predict(X_test)
print("precision",metrics.precision_score(y_test,y_pred=pred))
print("recall",metrics.recall_score(y_test,y_pred=pred))
print("accuracy:",metrics.accuracy_score(y_test,y_pred=pred))
print("f1_score", metrics.f1_score(y_test,y_pred=pred))
print(metrics.classification_report(y_test,y_pred=pred))
```

```
precision 0.8757040931280511
recall 0.9260092653871608
accuracy: 0.9740003015428945
f1_score 0.9001544004117344
              precision    recall  f1-score   support

           0       0.99      0.98      0.99     52138
           1       0.88      0.93      0.90      7555

    accuracy                           0.97     59693
   macro avg       0.93      0.95      0.94     59693
weighted avg       0.97      0.97      0.97     59693
```

Figure 44: Evaluation Value (CART)

Figures below will show all steps (from step1 to 7) used in calculating the C4.5 algorithm. There is only one difference in the code, here we use a folder called C4.5 that contains an additional code to calculate C4.5.

We may ask why we write special code for the C4.5 algorithm in the calculation of CART and ID3 decision tree algorithms, since there is a default parameter, we can use it to calculate easily the difference in C4.5. We use a separated file that are called in python code in the C4.5 file the file that its name is c45. So, in this folder (that named 'dec') we have two files: the basic one (C45_0.2.ipynb) and another file (c45) that we called in the basic one using the python code.

Figure 45 shows, the import of the C4.5 algorithm library used in python, loading of data, feature selection, and splitting of data.

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
import pandas
from sklearn import tree
from sklearn.metrics import confusion_matrix
import pandas
import seaborn as sns

from sklearn.model_selection import train_test_split
from c45 import c45
import pandas as pd
iris = pd.read_csv("C:/Users/acc/Downloads/puftimedif.csv")
attrNames=['timedifference','dur','prot','sipnum','dipnum_','sport','dport','pack','byt','bps','pps','bpp']
clf = c45.C45(attrNames)
```

Figure 45: Import libraries, loading data, feature selection, splitting of data

Figure 46 shows, the building of DT using the C4.5 algorithm

```
parameters = {'max_depth':range(3,25),
              'class_weight' : ["balanced"],
              'criterion':["gini"],
              'splitter':["best", "random"]
             }


clf = GridSearchCV(tree.DecisionTreeClassifier(),parameters,cv = 3, n_jobs=4,verbose=1)
clf.fit(X_train,y_train)
tree_model = clf.best_estimator_
X_pred = tree_model.predict(X_test)
y_true = y_train
cf_matrix = confusion_matrix(X_pred, y_test)
cm_df = pandas.DataFrame(cf_matrix)
sns.heatmap(cm_df, annot=True, fmt='g')
```

Figure 46: Building DT (C4.5)

Figure 47 shows, the python code to compute the basic values of the C4.5 DT algorithm.

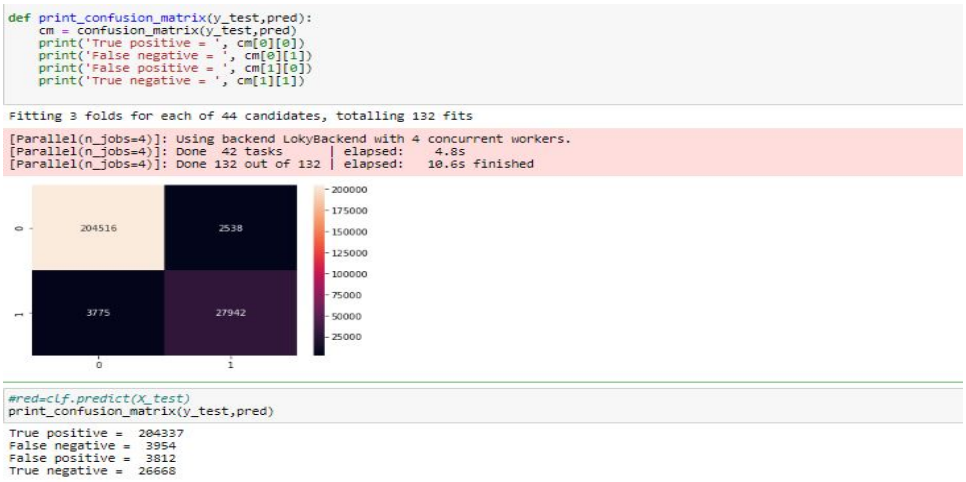


Figure 47: Basic Values (C4.5)

Figure 48 shows, how to compute the evaluation values **(Accuracy, Precision, Recall, and F1-score)** using python code

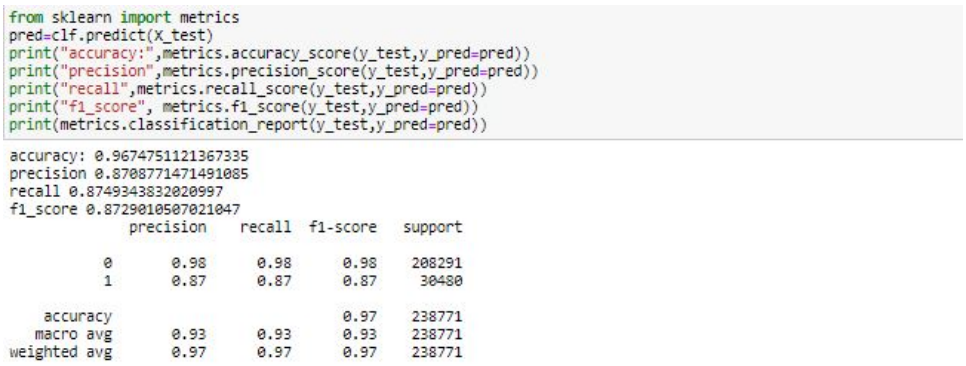


Figure 48: Evaluation values (C4.5)

In all the above figures, we use those two concepts **Grid Search parameter** and **KFlod Cross-Validation** that enhance the results of Evaluation and Basics values. **We can say that Grid searching** can be done across **machine learning** to compute the optimal or the best parameters to use for any given model. **Grid searching** is the process of analyzing the data to make optimal parameters for the given model. **We decide to use Grid search** to **get** the best hyperparameters

of a model we have which results in the most accurate predictions. The transaction **k** that related to the number of groups that a given data sample is to be split into that has a single parameter.

While **Cross-validation** is a resampling procedure **used** to evaluate machine learning models on a limited data sample.

ii. Neural network implementation:

Importing of the Required Libraries:

```
from numpy import loadtxt
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout
from google.colab import files #if the file is on device use this method
from keras import optimizers
from tensorflow import keras
import sklearn
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
```

Loading Data set (Upload dataset):

```
dataset = files.upload
df = pd.read_csv("puftimedif_update.csv")
data=df.to_numpy()
data = np.array(data)
# print(data.shape)
X= data[:,0:12]
y = data[:,12]
```

Splitting Data

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=
42)
```

Building Neural Network model 1 (Without weight & define keras) :

```
#without weight
# define the keras model
METRICS = [
      keras.metrics.TruePositives(name='tp'),
      keras.metrics.FalsePositives(name='fp'),
      keras.metrics.TrueNegatives(name='tn'),
      keras.metrics.FalseNegatives(name='fn'),
      keras.metrics.BinaryAccuracy(name='accuracy'),
      keras.metrics.Precision(name='precision'),
      keras.metrics.Recall(name='recall'),
      keras.metrics.AUC(name='auc'),]
model = Sequential()
model.add(Dense(12, input_dim=12, activation='selu'))#input layer
model.add(Dense(24, activation='selu'))
model.add(Dense(24, activation='selu'))
model.add(Dense(6, activation='softplus'))
model.add(Dense(1, activation='sigmoid'))
sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
adam=keras.optimizers.Adam(learning_rate=0.001,beta_1=0.9,beta_2=0.999,epsilon=1e-
07,amsgrad=True,name="Adam")
model.compile(loss='binary_crossentropy', optimizer=adam, metrics=METRICS)
model.fit(X_train, y_train, validation_data=(X_test,y_test), epochs=50, batch_size=128)
```

Evaluating model:

```
# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(X_test, y_test, batch_size=64)
print("test loss, test acc:", results)
```

Building Neural Network model 2 (Without weight & define keras) :

```
#without weight
# define the keras model
#batach size 64
```

```
METRICS = [
      keras.metrics.TruePositives(name='tp'),
      keras.metrics.FalsePositives(name='fp'),
      keras.metrics.TrueNegatives(name='tn'),
      keras.metrics.FalseNegatives(name='fn'),
      keras.metrics.BinaryAccuracy(name='accuracy'),
      keras.metrics.Precision(name='precision'),
      keras.metrics.Recall(name='recall'),
      keras.metrics.AUC(name='auc'),
]
model = Sequential()
model.add(Dense(12, input_dim=12, activation='relu'))
model.add(Dense(24, activation='relu'))
model.add(Dense(6, activation='softplus'))
model.add(Dense(1, activation='sigmoid'))
sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
adam=keras.optimizers.Adam(learning_rate=0.001,beta_1=0.9,beta_2=0.999,epsilon=1e-
07,amsgrad=True,name="Adam")

model.compile(loss='binary_crossentropy', optimizer=adam, metrics=METRICS)
model.fit(X_train, y_train, validation_data=(X_test,y_test), epochs=50, batch_size=128)
```

Evaluating Model:

```
# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(X_test, y_test, batch_size=64)
print("test loss, test acc:", results)
```

Building Neural Network model 3 (Without weight & define keras) :

```
#without weight
# define the keras model
#batach size 64
METRICS = [
      keras.metrics.TruePositives(name='tp'),
      keras.metrics.FalsePositives(name='fp'),
      keras.metrics.TrueNegatives(name='tn'),
      keras.metrics.FalseNegatives(name='fn'),
      keras.metrics.BinaryAccuracy(name='accuracy'),
      keras.metrics.Precision(name='precision'),
      keras.metrics.Recall(name='recall'),
      keras.metrics.AUC(name='auc'),
]
model = Sequential()
```

```
model.add(Dense(12, input_dim=12, activation='selu'))
model.add(Dense(24, activation='softplus'))
model.add(Dense(6, activation='softplus'))
model.add(Dense(1, activation='sigmoid'))
sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
adam=keras.optimizers.Adam(learning_rate=0.001,beta_1=0.9,beta_2=0.999,epsilon=1e-07,amsgrad=True,name="Adam")

model.compile(loss='binary_crossentropy', optimizer=adam, metrics=METRICS)
model.fit(X_train, y_train, validation_data=(X_test,y_test), epochs=50, batch_size=128)
```

Evaluating Model:

```
# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(X_test, y_test, batch_size=64)
print("test loss, test acc:", results)
```

Building Neural Network model 4 (Without weight & define keras) :

```
#without weight
      # define the keras model
      METRICS = [
            keras.metrics.TruePositives(name='tp'),
            keras.metrics.FalsePositives(name='fp'),
            keras.metrics.TrueNegatives(name='tn'),
            keras.metrics.FalseNegatives(name='fn'),
            keras.metrics.BinaryAccuracy(name='accuracy'),
            keras.metrics.Precision(name='precision'),
            keras.metrics.Recall(name='recall'),
            keras.metrics.AUC(name='auc'),
      ]
      model = Sequential()
      model.add(Dense(12, input_dim=12, activation='selu'))
      model.add(Dense(24, activation='selu'))
      model.add(Dense(24, activation='selu'))
      model.add(Dense(24, activation='selu'))
      model.add(Dense(24, activation='selu'))
      model.add(Dense(6, activation='softplus'))
      model.add(Dense(1, activation='sigmoid'))
      sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
      adam=keras.optimizers.Adam(learning_rate=0.001,beta_1=0.9,beta_2=0.999,epsilon=1e-07,amsgrad=True,name="Adam")

      model.compile(loss='binary_crossentropy', optimizer=adam, metrics=METRICS)
      model.fit(X_train, y_train, validation_data=(X_test,y_test), epochs=50, batch_size=128)
```

Evaluating Model:

```
      # Evaluate the model on the test data using `evaluate`
      print("Evaluate on test data")
```

```
126.        results = model.evaluate(X_test, y_test, batch_size=64)
127.        print("test loss, test acc:", results)
128.        # Generate predictions (probabilities -- the output of the last layer)
129.        # on new data using `predict`
130.        print("Generate predictions for 3 samples")
131.        predictions = model.predict(X_test[:3])
132.        print("predictions shape:", predictions.shape)
```

Building Neural Network model 5 (Without weight & define keras):

```
133.        #without weight
134.        # define the keras model
135.        METRICS = [
136.              keras.metrics.TruePositives(name='tp'),
137.              keras.metrics.FalsePositives(name='fp'),
138.              keras.metrics.TrueNegatives(name='tn'),
139.              keras.metrics.FalseNegatives(name='fn'),
140.              keras.metrics.BinaryAccuracy(name='accuracy'),
141.              keras.metrics.Precision(name='precision'),
142.              keras.metrics.Recall(name='recall'),
143.              keras.metrics.AUC(name='auc'),
144.        ]
145.
146.        model = Sequential()
147.        model.add(Dense(12, input_dim=12, activation='selu'))
148.        model.add(Dense(24, activation='selu'))
149.        model.add(Dense(12, activation='selu'))
150.        model.add(Dense(6, activation='selu'))
151.        model.add(Dense(3, activation='softplus'))
152.        model.add(Dense(1, activation='sigmoid'))
153.        sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
154.        adam=keras.optimizers.Adam(learning_rate=0.001,beta_1=0.9,beta_2=0.999,epsilon=1
    e-07,amsgrad=True,name="Adam")
155.
156.        model.compile(loss='binary_crossentropy', optimizer=adam, metrics=METRICS)
157.        model.fit(X_train, y_train, validation_data=(X_test,y_test), epochs=50, batch_si
    ze=10000)
```

Evaluating Model:

```
158.        # Evaluate the model on the test data using `evaluate`
159.        print("Evaluate on test data")
160.        results = model.evaluate(X_test, y_test, batch_size=5000)
161.        print("test loss, test acc:", results)
```

Building Neural Network model (With weight & define keras):

```
162.        #with weight
163.        # define the keras model
164.        METRICS = [
165.              keras.metrics.TruePositives(name='tp'),
166.              keras.metrics.FalsePositives(name='fp'),
167.              keras.metrics.TrueNegatives(name='tn'),
168.              keras.metrics.FalseNegatives(name='fn'),
169.              keras.metrics.BinaryAccuracy(name='accuracy'),
170.              keras.metrics.Precision(name='precision'),
171.              keras.metrics.Recall(name='recall'),
172.              keras.metrics.AUC(name='auc'),
173.        ]
174.        class_weight = {0: 1,
175.                        1: 2}
176.        model = Sequential()
177.        model.add(Dense(12, input_dim=12, activation='selu'))
178.        model.add(Dense(24, activation='selu'))
179.        model.add(Dense(6, activation='softplus'))
180.        model.add(Dense(1, activation='sigmoid'))
181.        sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
182.        adam=keras.optimizers.Adam(learning_rate=0.001,beta_1=0.9,beta_2=0.999,epsilon=1
    e-07,amsgrad=True,name="Adam")
183.
184.        model.compile(loss='binary_crossentropy', optimizer=adam, metrics=METRICS)
185.        model.fit(X_train, y_train, validation_data=(X_test,y_test), epochs=50, batch_si
    ze=10000,class_weight=class_weight)
```

Evaluating model (with weight):

```
186.        # Evaluate the model on the test data using `evaluate`
187.        print("Evaluate on test data")
188.        results = model.evaluate(X_test, y_test, batch_size=5000)
189.        print("test loss, test acc:", results)
190.
```

Chapter VI:

Comparison of Evaluation values (DT and NN)

I. Basic Values

The calculation of basic values had made up by using python language in anaconda navigator the basic values that are computes are **True Positive, True Negative, False Negative, and False Positive**. For each DT and NN, the basic values that have been calculated. However, we can calculate these basic values by writing python code to obtain a confusion matrix result (python code).

With the help of the Confusion matrix obtained these four values, where the Confusion matrix is the table that we can use to explain the performance of the classification model. Finally, the number of correct and incorrect predictions are brief with count values and divided by each class.

i. Basic Value's Explanation

The matrix is *m*x*m*, where *m* is the number of target values or classes. Whereas, the Performance of such models is commonly marked using the data in the matrix where the following table a 2x2 confusion matrix for two classes (Positive and Negative).

Numbers of (incorrect and correct) estimations or predictions made up by the classification model that makes a comparison to the actual outcomes (target value) in the data that we have all this locate under the confusion matrix concept.

True Positive (TP): Observation is positive and is estimate to be positive, the reality is positive, and testing of the variable estimates a positive.

For example, doctors make a PCR test for a group of people that are affected by corona virus and the test is accurately reported that.

False Negative (FN): Observation is positive, but is estimated negative, the truth is positive, but the test predicts a negative. For example, the groups of people that are sick in the corona make a PCR test, but the test inaccurately reports that they are not also it is called a type II error in statistics.

True Negative (TN): observation will give negative and it is estimates to be negative, the reality is showing negative, and the test estimates a negative.

For example, the group of people that do not have corona are not sick, and the PCR test accurately reports this

False Positive (FP): Observation is negative, but it is estimates positive, the reality is showing negative, but the test estimates a positive.

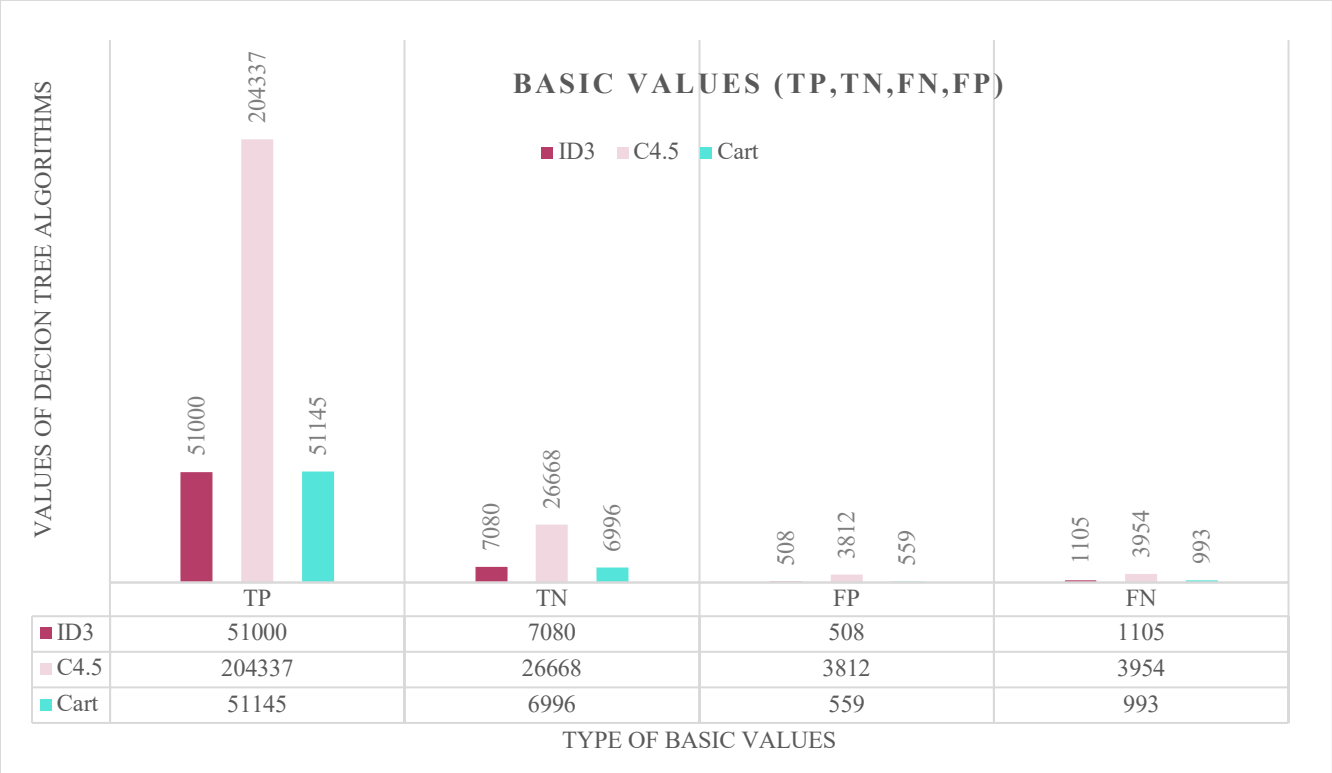
For example, the group of people do not have corona, but the test of PCR is an inaccurate reports that they are carrying the virus. It is also called a type I error in statistics

That's why, from the explanation above we can implies that (True Positive, True negative) must be greater than (False positive, False negative), then when this achieved so it will give better performance (using the dataset we have) and the concept called confusion matrix that give us the basic values that show in figures 44, 41, and 48 it's calculated for each algorithm in the decision tree.

The basic values that TP (True Positive) and TN (True Negative) should give us high values, since these two values means that the value of both the actual class and predicted class is Yes.

The Basic value of Decision Tree for each algorithm (CART, C4.5, and ID3 algorithm):

In this section, we start to calculate the basic values for each DT algorithms (ID3, C4.5, and CART) and NN these basic values will help us in the calculation of Evaluation Values which is very important in our work, since by depending on the evaluation values we will have the ability to compare values and allow us to know the best machine learning method that can be used in our task? Also, the display for those values in the figure below will facilitate the operation of analysis reaching the end result. From all that mentioned, allow the readers of this thesis to walk step by step in analysis operation and give the readers the ability to understand the calculation process.



	TP	TN	FP	FN
ID3	51000	7080	508	1105
C4.5	204337	26668	3812	3954
Cart	51145	6996	559	993

Figure 49: Decision tree algorithms basic values

Required the figure 49, the TP basic values should be the greatest value for the C4.5 DT algorithm that is equal 204337 that exceeds the Cart algorithm by 153192 and ID3 DT algorithm by 153337.

This means that C4.5 algorithm collected the highest value than Cart algorithm and ID3 algorithm respectively.

While, for TN basic values it gives the highest value for C4.5 algorithm too which is 26668 which surpassed the ID3 by 19588 and Cart algorithm by 19672.

This reveals that C4.5 algorithm recorded the highest value over ID3 algorithm and Cart algorithm orderly.

However, For TP basic values C4.5 algorithm presents the top one that is equal 3812 concerning other decision tree algorithm (ID3 algorithm and Cart algorithm) where it exceeds the Cart by 3253 and ID3 3304.

This implies that C4.5 algorithm surpassed Cart and ID3 orderly.

Furthermore, For FN basic values it gives the highest value for the C4.5 algorithm which is 3954 that surpassed ID3 algorithm by 2849 and Cart algorithm by 2961.

This consider that C4.5 algorithm recorded the greatest algorithm than ID3 and Cart algorithm respectively.

Therefore, the basic values' results showed that C4.5 algorithm is considered as the best algorithm over others.

Remark: The value of TP and TN reveals that it is the best algorithm.

Will C4.5 algorithm still the best one in Evaluation values as in basic values?

Basic value of Neural Network:

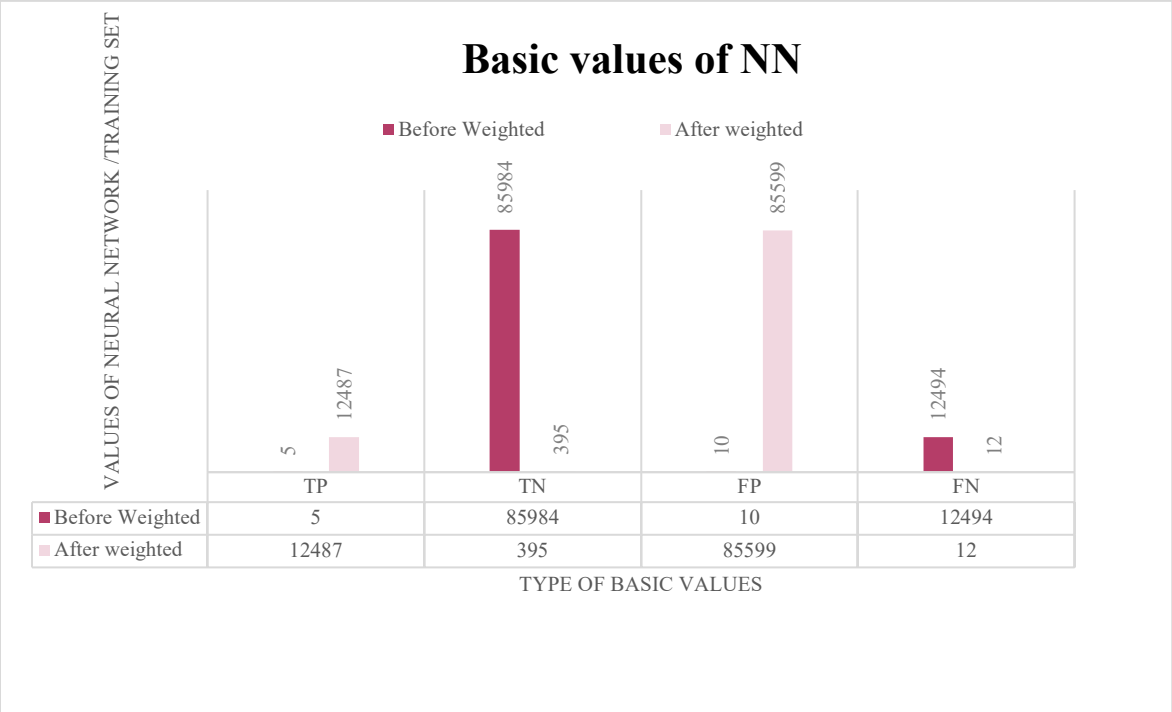


Figure 50: Neural Network basic values /Training set

This figure above shows the basic values with a neural network/Training set before and after weighted.

As shown in figure 50, in TP basic values of the neural network/ Training set before weighted was 5 increases highly to 12487 after weighted. But for TN, the basic values of the neural network/Training set were 85984 before weighted to decrease abruptly to 395 after weighted. Yet, For FP the basic values of the neural network/Training set were 10 before weighted to increase rapidly to 85599 after weighted.

While For FN the basic values of the neural network/Training set were 12494 before weighted to decrease fastly to 12 before weighted.

Thus, the weighted process increases the basic values of the neural network/Training set of TP and FP, but it decreases in FN and TN.

Therefore, the weighted process enhances the performance of the neural network/Training set. Weighted process with neural network / Training set play good role in increasing the performance.

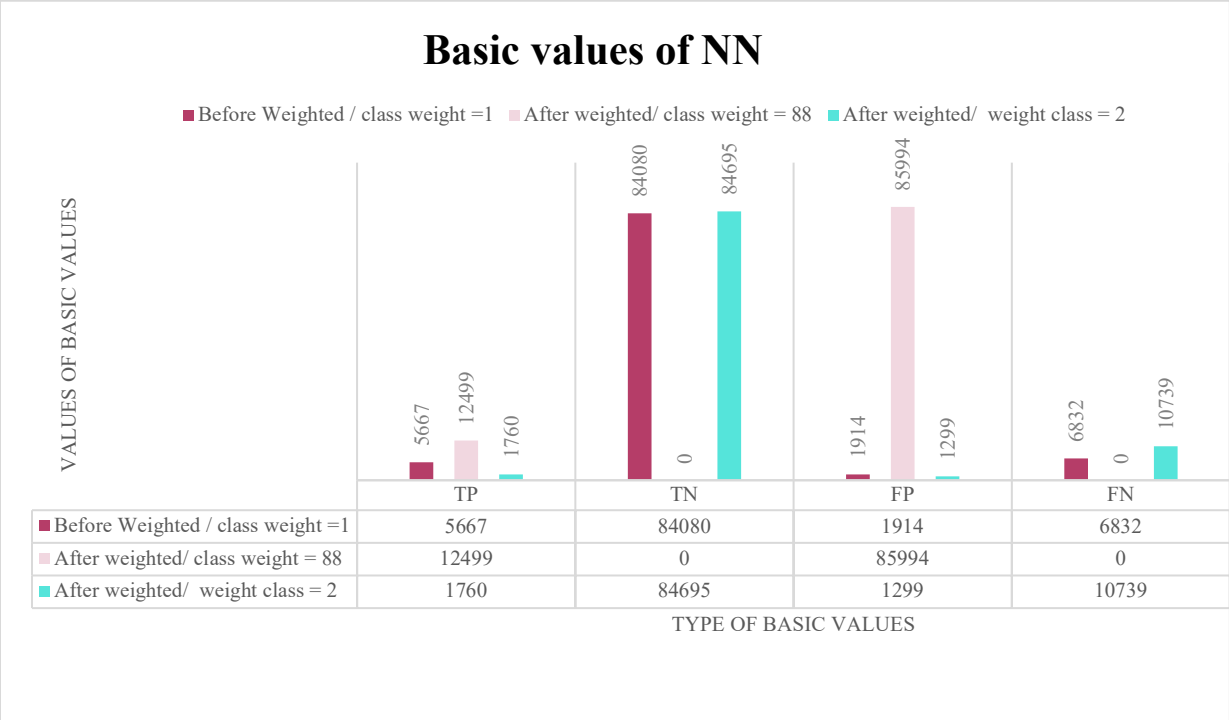


Figure 51: Neural Network basic values /Testing set

As shown in the figure 51, the values of neural network/Testing set records the greatest value of that after weighted/class = 88 in TP basic value which exceeds that of before weighted class weight =1 by 6832 and after weighted/class weight = 2 by 10739. However, the values of neural network/Testing set for that after weighted/class weight equal 2 takes the highest value 84695 in the TN basic value where it exceeds that before weighted class weight=1 by 615 and after weighted/class weight equal 88 by 84695.While, the values of neural network/testing of that after weighted class weight equal 88 records the greatest value which is equal to 85994 where it exceeds that before weighted class weight=1 84080 that after weighted class weight=2 by 84695. Furthermore, the values of neural network /testing set of that after weighted/class weight equal 2 gives the highest value in FN 10739 which exceeds that before weighted class weight=1 by 3907 and that of after /class weight equal 88 by 10739

Thus, the highest neural network/testing set values in TP is that of after weighted class weight equal 88, TN is that after weighted class weight equal 2, FP after weighted/class weight 88 and in

FN is that after weighted/class weight equal 2, also the zero values of that after weighted/class weight equal 88 in TN and after weighted of class weight equal 88 in FN.

Therefore, balancing data of neural network / Training set gives the highest performance when it balanced on training set. Moreover, on the test set it plays an opposite role (as in that after weighted/class weight equal 88 where TP, FP recorded highest score while, FN, TN recorded zero values) where the classifier starts induce wrong classification (due to weighting of instance in training set), that's why it gives the dominant for the highest weighted, and when new class introduced to make classification it automatically based on dominant class that induces wrong classification method.

- In conclusion, imbalanced of dataset in neural network is totally rejected because it misuses its role and gives negative results.

Tuning hyperparameter / Basic values:

Hyperparameter had already selected by the users via continuous testing to reach the best result value that gives the highest performance the calculation of basic values of hyperparameter (network size, activation function, and optimizer) will be shown in the below figures (52, 53, and 54).

Basic values of network size:

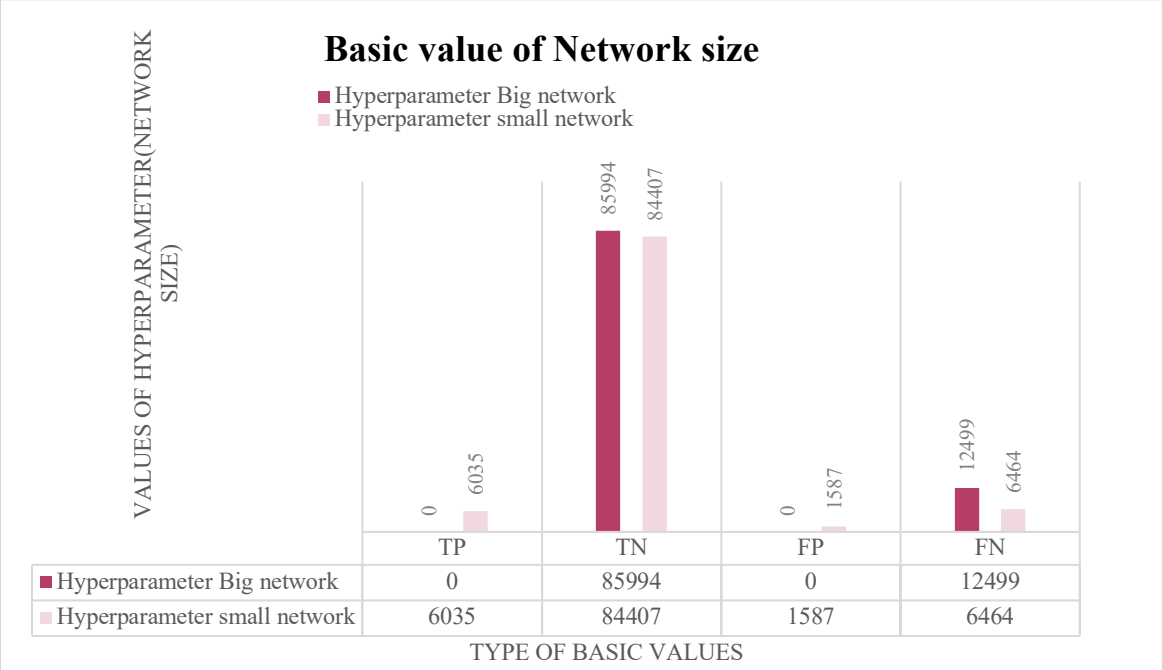


Figure 52: hyperparameter/ Network size (Bigger and small)

As shown in the above figure 52, we start testing the network size to reach the best result. First, depending on the increasing of network size the basic value results shown are TP and FP equal to zero, TN is 85994, and FN is equal to 12499. while, depending on the decrease of network size the result shown is higher in TP that is equal to 6035, and FP that is 1587, but lower in TN which is equal to 84407, and FN is 6464. Even though the big network gives higher values in TN, FN that may give higher accuracy than the small network, but the small network may give higher in the recall, precision, and accepted accuracy because it has higher TP.

Therefore, the basic value results of the small network are considered better than the basic value of big network results.

Basic values of Activation function:

To get the best activation function we made different tests and coded by change 1, change 2, and change 3

We depend on the try and error method for each activation function type to reach the best activation function as shown below. For example, in change 1 we use activation values type orderly relu, relu, softplus, and sigmoid, and then we made change 2 to get better than that results of change 1 and so on.

Change 1:

```
model.add(Dense(12, input_dim=12, activation='relu'))
model.add(Dense(24, activation='relu'))
model.add(Dense(6, activation='softplus'))
model.add(Dense(1, activation='sigmoid'))
```

change2:

```
model.add(Dense(12, input_dim=12, activation='selu'))
model.add(Dense(24, activation='selu'))
model.add(Dense(6, activation='softplus'))
model.add(Dense(1, activation='sigmoid'))
```

Change 3:

```
model.add(Dense(12, input_dim=12, activation='selu'))
model.add(Dense(24, activation='softplus'))
model.add(Dense(6, activation='softplus'))
model.add(Dense(1, activation='sigmoid'))
```

For each change, we compute the basic values and compare them.

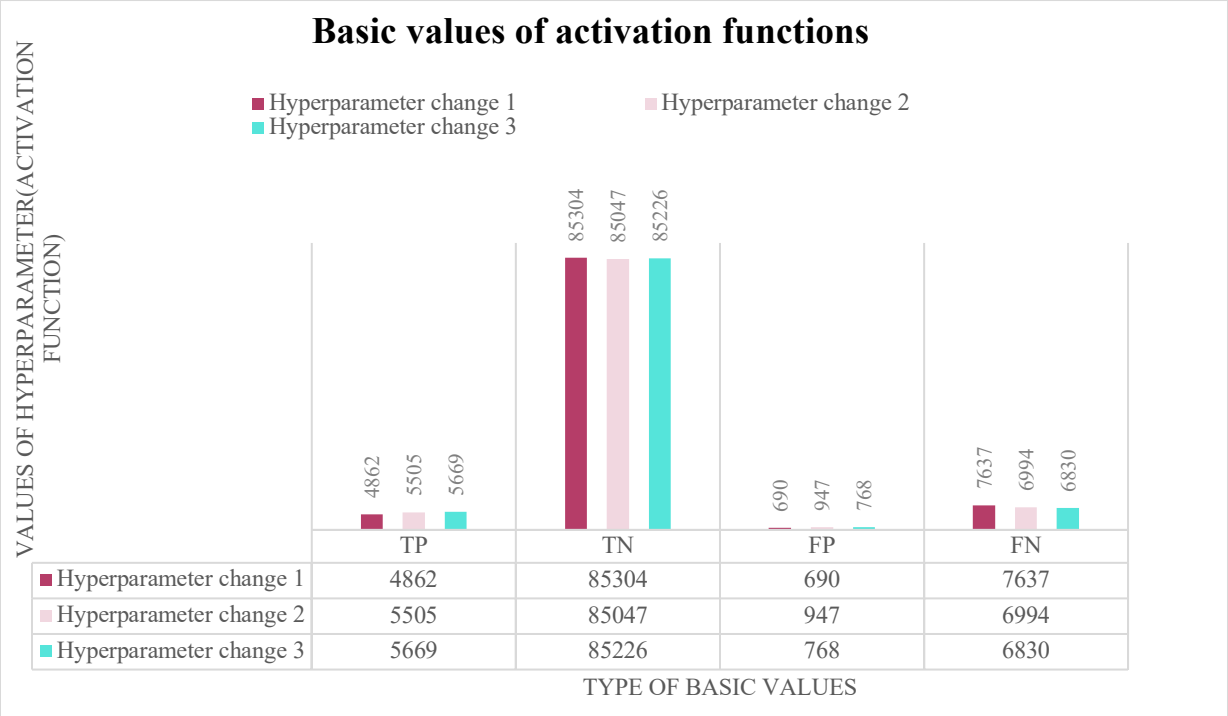


Figure 53 : hyperparameter/ Activation Function

As shown in the above figure 53, the change on gives in TP the lowest value 4862 and in FP which is equal 690, and the highest value in TN that is equal 85304 and in FN that is equal 7637. Moreover, change 2 recorded the intermediate value in TP that is equal to 5505 and in FN, and the highest value recorded in FP is equal to 947 and in FN that is equal to 6994, but the lowest value in TN that is equal to 85047. Although, change 3 recorded the highest value in TP that is equal to 5669, but the lowest value in FN that is equal to 6830, while the intermediate values in TN that is equal 85226 and in FP that is equal 768.

This means that we improve the fine activation function because change 3 gives an acceptable result.

Basic values of Optimizer:

We test SGD and ADAM optimizer and found out the basic value results for both.

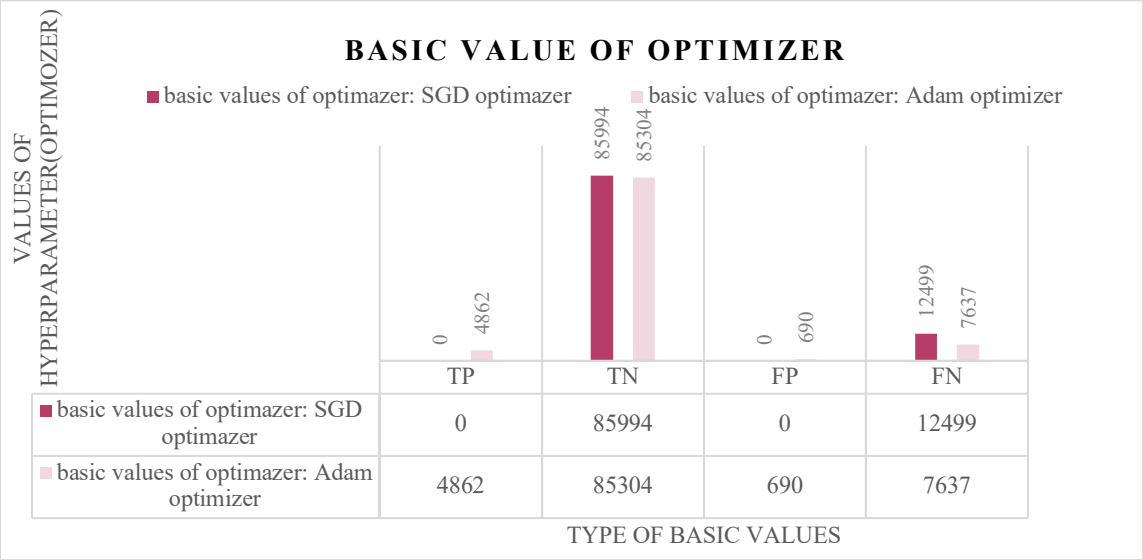


Figure 54: Tuning parameter / Optimizer

As shown in the above figure 54, the SGD optimizer recorded zero values in TP and FP while the highest value in TN that is equal to 85994 and in FN is equal to 12499. While, for Adam optimizer, the highest value recorded in TP is equal to 4862, and in FP that is equal to 690, but the lowest value in TN that is equal to 885304 FN that is equal to 7637.

This shows that by using Adam optimizer the results are more acceptable.

II. Evaluation values

The evaluation values have been calculated also using python. Whereby, these values can determine the performance of the datasets. Using our dataset (PUF.csv) we calculated **(RECALL, PRECISION, F1-SCORE, and ACCURACY),** all these value can determine the performance of the dataset we have.

i. Evaluation values explanation

Accuracy:

Accuracy is the most intuitive one and it is the ratio of the correctly labeled subjects to the whole group of subjects.

Accuracy comes to answer the following question:

- **How many students in the school did we correctly label out of all the students we have in the school?**

This is the formula of accuracy

Accuracy = (TP + TN) / (TP+ FP + FN + TN)

Precision:

This below is the formula of precision

Precision = TP / TP+ FP

The rate of correctly estimated or estimated positive values to the total estimated positive values this concept is called precision.

Recall (Sensitivity):

The recall or sensitivity is the rate of correctly estimated positive values to the actual positive values. Recall appears the sensitivity of the algorithm.

Formula:

Recall = TP / TP+ FN

F1 Score:

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

F1 score is the weighted average of both precision and recall. F1 might seem complicated it is a more developed metric than accuracy, since it takes into account both false positives and false negatives.by which, the concept of accuracy is good to use only when both false positives and false negatives have similar costs. Precision, Recall, and F1 Score offer a good alternative to the traditional accuracy metric and offer detailed vision about the algorithms that are under the analysis. [19]

Those values have been applied in decision tree algorithms and neural networks.

There are three decision tree algorithms; for each one the evaluation values had calculated by using four types of evaluation values (Recall, Accuracy, Precision, and F1-Score) as shown in figure 55.

Decision tree algorithms and neural network evaluation values figure:

In this section, we reach the last step that will allow us to take the right decision in choosing the machine learning method that should be suitable to detect the DNS Tunneling issue. So, we start to calculate the evaluation values (Accuracy, Precision, Recall, and F1_score) for each DT algorithms and NN. After that, we start to compare the results to end the decision all that will be shown in the figure below.

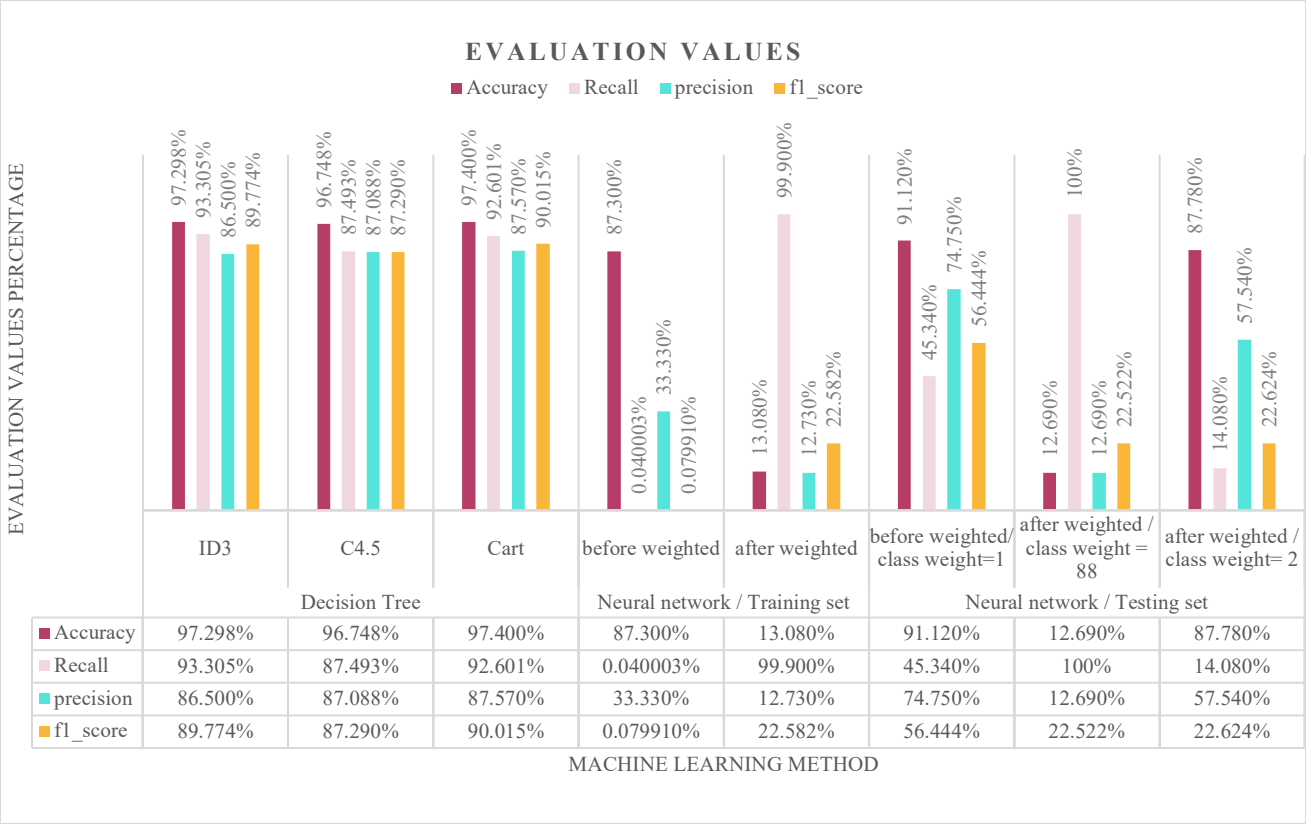


Figure 55: Evaluation values with decision tree algorithms and neural network

Figure 55 shows all the results that we calculate for it, and then we will start to make comparisons to the results at the end.

The Accuracy in red color, Recall in pink, precision in blue color, and F1_score in yellow color. Each algorithm has four values in four colors so each algorithm has its Evaluation values with its

percentage. At the x-axis, we have (Machine learning Methods) while on Y-axis we have the Evaluation values percentage. All the results above are in % and we make rounding for the decimal numbers. We have in the figure above the values of Decision tree algorithms (ID3, C4.5, and Cart), Neural Network in training set before we use the weighted process for values and after, and Neural Network in Testing set before and after the weighted process with the class weight that is changed from 1 that is before weighted to class weight =88, then we change class weight that equal to 2. So the class weights in NN change many times (all this showed in figure 53). Let's start with the analysis of the figure above in detail.

As shown in figure 54 above, for the decision tree algorithm the highest value of accuracy recorded is responsible for Cart that is equal 97.400% that outperform 0.102% of the ID3 algorithm, and in the C4.5 algorithm by 0.652%. Recall that the greatest value recorded is for the ID3 algorithm equal 93.305% that surpasses that of the Cart algorithm by 0.704% and C4.5 by 5.812%. Yet, for precision, the top value recorded is in the Cart algorithm is equal to 87.570% that outperforms that in the C4.5 algorithm by 0.482% and of the ID3 algorithm by 1.07%. However, for that f1_score the highest value recorded that is equal to 90.015% is responsible for the Cart algorithm that exceeds that of the ID3 algorithm by 0.241% and the C4.5 algorithm by 0.725%. Moreover, In the neural network /training set recorded the highest value of accuracy in before weighted which is equal 87.300% that outperform after weighted by 74.22%, same as that in precision where before weighted neural network /training set which is equal 33.330% outperform that in after weighted by 20.6%. On the other hand, it recorded that the greatest value in after weighted that is equal to 99.900% and surpass before weighted by 99.859997% also for that of f1_score where the highest score recorded in after weighted that is equal to 22.582% and exceeds that in before weighted by 22.50209%.

Furthermore, for neural network /testing set for that of accuracy, it recorded the top value in before weighted /class weight equal 1 that is equal 91.120% that exceeds that in after weighted /class weighted equal 2 by 3.34% and that in after weighted/class weight equal 88 by 78.43%, the same as that in precision and f1_score but in precision the highest percentage recorded is responsible for that before weighted/ class weight equal 1 that is equal 74.750% that surpasses that of after weighted /class weight equal 2 by 17.12% and after weighted /class weight equal 88 by 62.06%

and for f1_score the greatest percentage recorded is in that of before weighted /class weight equal 1 that is equal 56.444% that outperform that after weighted /class weight equal 2 by 33.82% and that of after weighted /class weight equal 88 by 33.922%

What about the decision tree and neural network /testing set?

By referring to what is shown in the decision tree and neural network/Testing set, we can compare them accurately. But in the decision tree, the percentage of accuracy in the Cart algorithm recorded is equal to 97.400% more than that in the neural network/Testing set of after weighted /class weight equal to 2 by 9.62%, the same as that of precision and f1_score but in the precision the highest percentage recorded in the Cart algorithm equal 87.570% that exceeds that in neural network /Testing set in after weighted /class weight equal 2 by 30.03% and for f1_score the top recorded percentage in the Cart algorithm that is equal 90.015% more than that in neural network /Testing set in after weighted /class weight equal 2 by 67.391%. On the other hand, for the recall the highest percentage recorded for the ID3 algorithm which is equal to 93.305% and outperforms that of the neural network /Testing set of after weighted /class weight equal 2 by 79.225%.

In our example (DNS tunneling detection), the **Precision** is **more** important than **Recall**. **Precision** tells when we predict something positive and how many times they were positive. Moreover, remember that this tells us from the actual positive data, how many times we correctly expected it. Updating precision often affects recall and vice versa. **F-Score** gives a single **score** that balances both **precision** and **recall** in one number. The recall and precision were affected if the basic values (FP and FN) respectively increase or decrease. Maximizing precision will minimize the number of false positives, whereas maximizing the recall will minimize the number of false negatives.

Thus, in the decision tree, the Cart algorithm recorded the highest accuracy, precision, and f1_score and a low percentage in the recall. While low percentages recorded in the neural network/ Testing set as a whole.

Therefore, the decision tree is the most convenient machine learning method for DNS tunneling detection using our dataset (PUF).

Tuning hyperparameter / Evaluation values:

In this section, we compute the evaluation values of the tuning hyperparameter (Size of the network, Activation function, and optimizer) to know if the changes in values are good or not all this will show in the figures below.

Evaluation value of Network size:

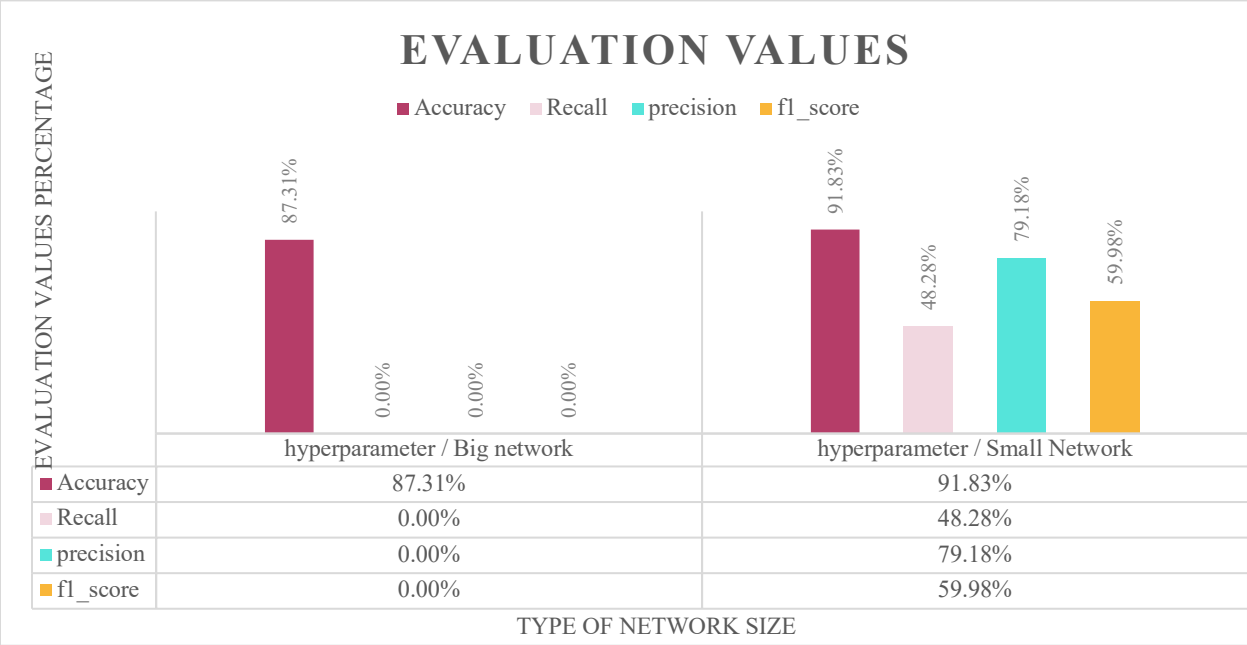
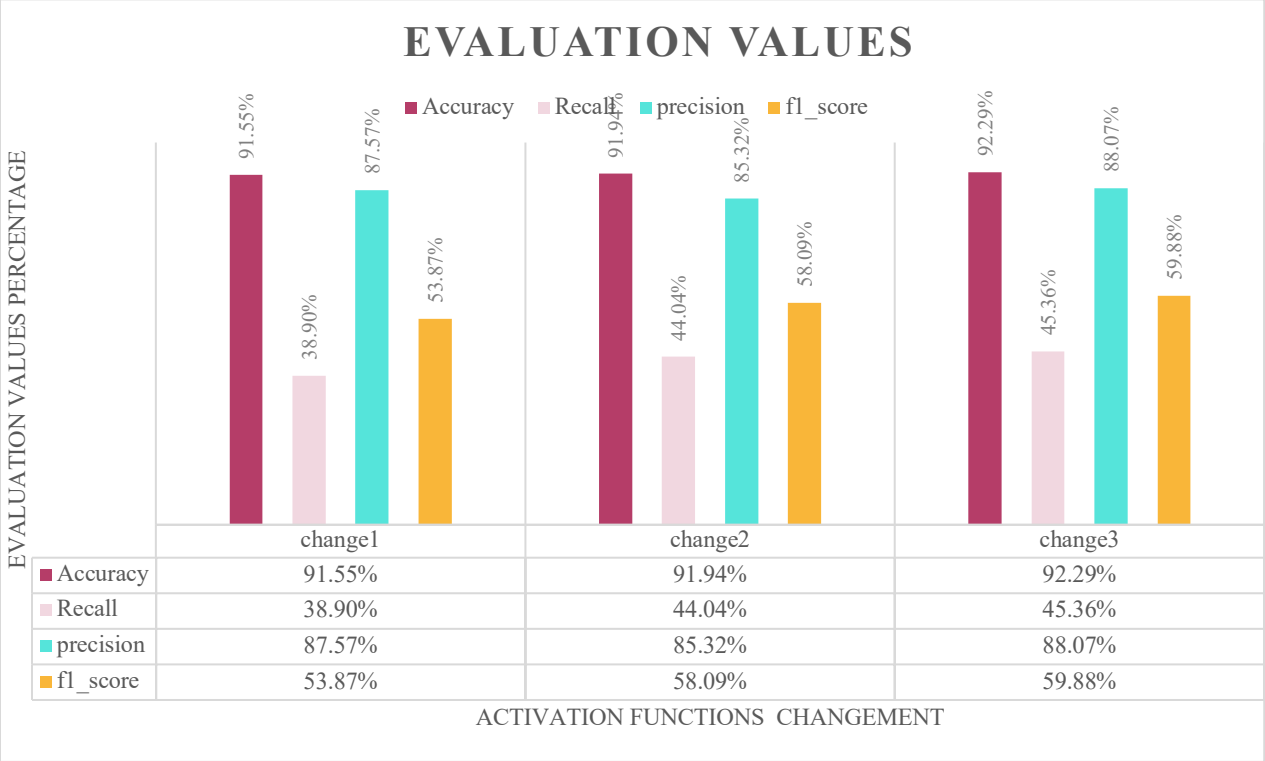


Figure 56: evaluation values of hyperparameter/size of network.

As shown in figure 56, the small network recorded the highest value in all the evaluation values (accuracy 91.83%, recall 48.28%, precision 79.18%, and f1_score 95.98%). while the big network gives an accuracy of 87.31% and zero value for the others.

Therefore, the decrease in network size gives better results.

Evaluation value of Activation function:



	change1	change2	change3
Accuracy	91.55%	91.94%	92.29%
Recall	38.90%	44.04%	45.36%
precision	87.57%	85.32%	88.07%
f1_score	53.87%	58.09%	59.88%

Figure 57: Evaluation values of hyperparameter/activation function.

As shown in the above figure 57, the change 3 of activation function gives the highest evaluation value with respect to change 1(accuracy 91.55%, recall 38.90% , precision 87.57%, and f1_score 53.87%)and change 2 (accuracy 91.94%, recall 44.04%, precision 85.32%, and f1_score 58.09%)where accuracy equal 92.29%, recall equal 45.36%, precision equal 88.07%, and f1_score 59.88%.

Thus, change 3 shows the best evaluation values percentages concerning change 1 and change 2.

Therefore, change 3 allow the achievement of the best activation function.

Evaluation value of Optimizer:

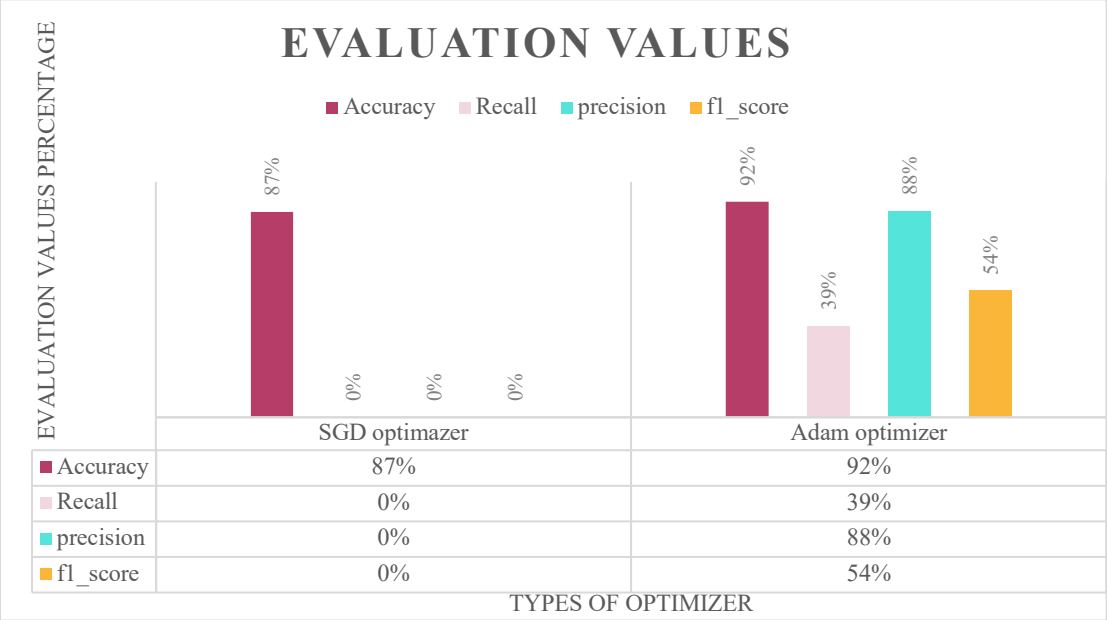


Figure 58: Hyperparameter optimizer's evaluation values

As shown in the above figure 58, Adam optimizer recorded the highest evaluation values (accuracy 92%, recall 39%, precision 88%, and f1_score 54%) with respect to SGD optimizer (accuracy 87% and zero values for recall, precision, f1_score). Adam optimizer is better than SGD because in our case there are binary values.

Thus, Adam optimizer shows the best evaluation values percentage results.

Therefore, after testing Adam optimizer is better than that of SGD, which gives the best results.

CONCLUSION:

As a matter of fact and what already mentioned in this report, the audience can now with thoroughness express more about the detection of DNS Tunneling by using the studied machine learning methods, decision tree algorithms, and deep machine learning (Neural Network) to get in touch with the difference of decision tree algorithms and the calculation methods. Whereby, we test the whole concept of neural network and enhancement of its performance by practicing different ways of improvement on the network and dataset.

The above simulation figures drive us hardly to reach the accurate results such as, the C4.5 is consider the best algorithm than others (ID3, Cart) according to the basic values showed where C4.5 recorded the highest basic value than ID3, Neural Network, and Cart, while depending on Evaluation value the Cart is considered the best algorithm.

Moreover, the final results are depend basically on the Evaluation value, that allow us to recognize Cart as the best decision tree algorithm than ID3, Neural Network, and C4.5, also according to the weighted process, in addition the statistical results demonstrate that it enhances the performance of the neural network/Training set and play a significant role on it that induces the increasing the performance.

Also, this trigger the highest performance of the neural network/Training set, that is why the classifier starts to induce wrong classification (due to the weighting of instance in the training set). For that reason, dominant results are given to the highest weighted, and when a new class is introduced to make classification, it is automatically based on the dominant class that triggers the wrong classification. This allows the total rejection of the imbalanced dataset in the neural network due to the misusing of its role and the given of negative results.

From what is preceded and depending on the statistical results, the decision tree is recommended as the best machine learning that must be used in DNS tunneling detection using our data set (PUF).

For the announcement, all the above results shown in this thesis is not the standard results that we can depend on them, as a rule, it specifically depends on the dataset given (for example, Cart may not be the best algorithm in all cases). Last but not least, the notion of DNS tunneling must be

taken into consideration all over the world because it induces a dangerous attack that is called a cyber-attack, which practically harms user's personal information and owner's sensitive data over the DNS domain.

REFERENCES

[1] https://link.springer.com/chapter/10.1007/978-3-642-24212-0_15

[2]https://www.researchgate.net/publication/289980169_An_Overview_of_Machine_Learning_and_its_Applications

[3] http://ceur-ws.org/Vol-2343/paper13.pdf

[4] https://expertsystem.com/machine-learning-definition/

[5] https://www.sas.com/en_us/insights/analytics/machine-learning.html

[6] https://www.saedsayad.com/decision_tree.htm

[7]https://www.researchgate.net/publication/322525657_Comparative_Analysis_for_Detecting_DNS_Tunneling_Using_Machine_Learning_Techniques

[8] https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

[9] https://www.datacamp.com/community/tutorials/decision-tree-classification-python

[10] https://deepai.org/machine-learning-glossary-and-terms/neural-network

[11]: https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a

[12]:https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c

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

[14]:https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/

[15]:https://datascience.stackexchange.com/questions/25485/are-neural-networks-able-to-deal-with-non-normalised-inputs

[16]:https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029

[17]: Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. 2014. arXiv:1412.6980v9

[18] https://www.saedsayad.com/model_evaluation_c.htm

[19]https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124