UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERIA DE TECNOLOGIAS Y SERVICIOS DE TELECOMUNICACION

TRABAJO FIN DE GRADO

DEVELOPMENT OF INTRUSION DETECTION MODELS FOR CYBERSECURITY IN COMPUTER NETWORKS APPLYING MACHINE LEARNING ALGORITHMS.

FERNANDO LORO VELARDO

2019

TRABAJO DE FIN DE GRADO

Título:	Desarrollo de Modelos de Detección de Intrusiones en Redes de Ordenadores aplicando Algoritmos de Machine Learning.
Título (inglés):	Development of Intrusion Detection Models for CyberSecurity in Computer Networks applying Machine Learning Algorithms.
Autor:	Fernando Loro Velardo
Tutor:	Álvaro Carrera Barroso
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO DE FIN DE GRADO

DEVELOPMENT OF INTRUSION DETECTION MODELS FOR CYBERSECURITY IN COMPUTER NETWORKS APPLYING MACHINE LEARNING ALGORITHMS.

FERNANDO LORO VELARDO

JULIO 2019

Resumen

Internet crece a diario en términos de usuarios y aplicaciones. Hoy, más de la mitad de la población mundial utiliza Internet. Debido al gran número de usuarios y dispositivos conectados a la red, el volumen de información almacenada crece de forma exponencial. Estos datos son más valiosos cada día y su protección se convierte en una prioridad. Pero no solo aumenta la cantidad de datos sino también el número de amenazas. A diario aparecen nuevas vulnerabilidades en redes y sistemas. Protegerse de estos peligros es un gran reto debido a su rápido surgimiento y evolución. Esto supone crear medidas de protección para ataques que incluso hoy no son conocidos. Es por ello que, tanto la comunidad científica como empresas privadas trabajan mano a mano para desarrollar herramientas de defensa y protección.

Tanto los Intrusion Detection Systems (IDS) como los Intrusion Prevention Systems (IPS) son herramientas diseñadas para detectar y prevenir ciberataques. Esto lo consiguen analizando el tráfico entrante y saliente de la red protegida. Estas herramientas basan su comportamiento en modelos o conjuntos de reglas que determinan si una comunicación es o no peligrosa. Este proyecto tiene como objetivo desarrollar modelos basados en Machine Learning destinados a IDSs e IPSs. Estos modelos deben ser capaces de detectar ciberataques. Para conseguirlo, los modelos son entrenados con grandes colecciones de datos denominadas datasets que registran el tráfico de red etiquetando cada flujo como benigno o malicioso.

Los ciberataques analizados en este proyecto para los cuales hemos propuesto un conjunto de soluciones son: DDoS, Port Scan, Infiltración, Botnets, Web Attacks y Fuerza Bruta. Estos son los ataques más comunes dirigidos a redes privadas que se llevan a cabo a día de hoy.

En cuanto al procedimiento seguido para generar modelos se divide en tres partes: preprocesado de la información, en el que se adecúa y corrige la información contenida en los datasets de tal forma que sea apropiada para los algoritmos. A continuación, comienza el entrenamiento, este es un proceso iterativo en el que generamos un gran abanico de modelos. Esto se consigue variando el algoritmo utilizado (k-Nearest Neighbors, Logistic Regression, Gaussian Naive Bayes y Multilayer Perceptron) así como su configuración y

número de variables utilizadas. En este proyecto, hemos desarrollado en total unos 1350 modelos. Por último, se evalúa cada modelo con diferentes medidas estadísticas para su posterior comparación.

Finalmente, comparamos los resultados obtenidos tras evaluar todos los modelos generados. A partir de estos datos podemos concluir cuáles son los mejores algoritmos para cada tipo de ataque así como la información que requieren. Los modelos obtenidos alcanzan una exactitud a la hora de clasificar flujos de entorno al 98%. Estos valores son realmente buenos ya que el porcentaje de falsas alarmas y flujos malignos no detectados es de tan solo un 2%. Por ello podemos concluir en que aplicar técnicas de Machine Learning a herramientas de ciberseguridad es una interesante propuesta con un gran rendimiento.

Palabras clave: Machine Learning, Cybersecurity, IDS, IPS, Dataset, Intrusion.

Abstract

The Internet grows daily in terms of users and applications. Today, more than half of the global population is using the Internet. Due to the huge number of users and devices connected to the network, the volume of information stored grows exponentially. This data are more valuable each day and its protection becomes a top priority. Not only the amount of data increase but also the number of threats. Every day, new exploits and vulnerabilities in systems come up. That is why protection measures need to be created even for unknown attacks. To be protected from these threats is hard labour due to the rapid emergence and evolution of the threats. Both the scientific community and companies work together in order to create new defensive tools.

Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) are tools designed to detect and prevent cyber-attacks. This goal is achieved analyzing the inbound and outbound traffic of the protected network. These tools are based on models or a set of rules that classify flows as benign or malicious. The goal of this project is to develop Machine Learning based models designed for IDSs and IPSs. These models must be able to detect cyber-attacks. To achieve that, models are trained with large data collections named datasets. Datasets log the network traffic labelling every flow as benign or malicious.

In this project, we propose a set of solutions to the next cyber-attacks analyzed: DDoS, Port Scan, Infiltración, Botnets, Web Attacks and Brute Force. Nowadays, these are the most common attacks directed to private networks.

The methodology followed to elaborate models is divided into three parts. The first part is named preprocessing, it consists of making sure that information stored in datasets is correct and appropriate for training. Datasets usually have incorrect or missed pieces of information that have to be fixed. Next, the training begins, this is an iterative process repeated for each developed model. We generate a wide range of models varying the algorithm applied (k-Nearest Neighbors, Logistic Regression, Gaussian Naive Bayes and Multilayer Perceptron), its configuration and the number of used features. We have trained a total of about 1350 models. The last part consists of evaluating each model with different statistical scores based on the test performance and compared with other solutions.

Finally, we compare the results obtained after all models are evaluated. With this information, we can conclude which algorithm is the best for each attack type and the information required. The models developed reach an accuracy of about 98% classifying flows. These scores are successful because the false alarms ratio and undetected malicious flows are only 2%. That is why we can conclude that applying Machine Learning techniques to Cybersecurity is an interesting proposal with a great performance.

Keywords: Machine Learning, Cybersecurity, IDS, IPS, Dataset, Intrusion.

Agradecimientos

En primer lugar, dar las gracias a mi tutor, Álvaro por todo el tiempo y la paciencia dedicada. Por ofrecerme su ayuda cuando la necesitaba.

También dar las gracias a Michel, Manuel, Sergio y Beatriz, porque ellos tienen tanto mérito como yo en los pasos que me han hecho llegar hasta aquí.

Y por supuesto, a mi familia porque ellos han hecho posible que pueda continuar con mi educacuón de la que siempre se han preocupado.

Contents

List of Figures

Glossary

CIC Canadian Institute for Cybersecurity

CSV Comma-separated Values

DM Data Mining

DDoS Distributed Denial of Service

FTP File Transfer Protocol

GNB Gaussian Naive Bayes

GE Gigabit Ethernet

IDS Intrusion Detection System

IPS Intrusion Prevention System

k-NN k-Nearest Neighbors

LR Logistic Regression

ML Machine Learning

MLP Multilayer Perceptron

NGIPS Next-Generation Intrusion Prevention System

SSH Secure Shell

CHAPTER 1

Introduction

This chapter gives a brief introduction to the project. To get a better vision and understanding, we explain the context in which is situated this work. With this purpose, we explain the two main terms implied in this work: Intrusion Detection System and Machine Learning. Furthermore, we review the main reasons that motivate this project and the project goals. Finally, we give a short explanation of the document structure.

1.1 Context

The Internet grows daily in terms of users and applications. Today, more than half of the global population is using the Internet [14]. Due to the huge number of users and devices connected to the network, the volume of information stored grows exponentially. This data are more valuable each day and its protection becomes a top priority. Not only the amount of data increase but also the number of threats. Every day, new exploits and vulnerabilities in systems come up. That is why protection measures need to be created even for unknown attacks. To be protected from these threats is hard labour due to the rapid emergence and evolution of the threats. Both the scientific community and companies work together in order to create new defensive tools.

Cybersecurity as Cisco describes is "The practice of protecting systems, networks, and programs from digital attacks. These cyber attacks are usually aimed at accessing, changing, or destroying sensitive information; extorting money from users; or interrupting normal business processes" [13]. Today, there are as many protection tools as types of cyber-attacks. However, the fast progress turns effective protection applications into deprecated software as cyber attackers become more innovative. In this context IDS or Intrusion Detection Systems becomes a useful and powerful tool.

IDS or Intrusion Detection System is a software application that monitors network traffic looking for malicious activity. There are two main types of IDS: Signature-Based and Anomaly Based. The Signature Based ones are a good option to detect known cyber-attacks because they detect specific patterns or signatures in the malicious packets. However, this IDS type has a bad performance with zero-day vulnerabilities. The Anomaly-Based Intrusion Detection Systems base their behaviour in a set of rules that define the normal activity of the network. Any event that differs from the normal activity is classified as a potential attack. Once an attack is detected, IDS can act as a warning or actively modifying the network configuration. The Anomaly Based IDS can detect zero-day attacks however, the number of false positives is its biggest drawback. In this project, we focus on Anomaly Based IDS since they can be based on Machine Learning models.

Machine Learning is a branch of Artificial Intelligence that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead. The fact of not being based on explicit instructions makes Machine Learning a versatile and powerful tool. It is especially helpful to solve unknown and complex problems. That is why Machine Learning is applied to IDSs. Some IDSs base their behaviour in a Machine Learning model. Models are made from large data collections, an algorithm

and a training process. Once the model is trained, it can make predictions from new input data.

The information needed to train IDS models comes from the packet captured from the network. The network traffic is logged on the different machines it passes through. This traffic generates large amounts of data that need to be processed. In order to train new models, we need to distinguish benign from malicious flows in the logged data. Once the model is trained, the IDS can predict benign and malicious packets simply monitoring the network.

1.2 Motivation

There are several reasons that motivate the realization of this project. First of all, the continuous growth of the Internet. Each day the number of users, data and applications increase. Apart from that, the dangers of the network raise. The protection of data is a continuous and highly valued labour that implies companies and scientific communities all over the world. Every day, companies take cybersecurity more seriously and spend bigger amounts of money on protection tools. According to the 2019 Official Anual Cybercrime Report sponsored by Herjavec Group, cybercrime will cost the world 6 trillion $ annually by 2021 [8]. In fact, the previous predictions made by this group have proven this trend.

In the same line as the previous point, the fast evolution of cyber-threats requires a fast adaptation and creation of new defensive tools, even to unknown attacks. The cyber-threats of tomorrow are unpredictable but we have to face them. Intrusion Detection Systems are a good solution to this kind of problems.

Finally, there are not too many studies about Intrusion Detection Datasets in part because the existing were practically obsolete. However, today we can collect immense amounts of data what make possible to apply Data Mining and Machine Learning technologies. Furthermore, thanks to the big progress of computing capacities, the cost of these technologies is much lower.

1.3 Project goals

This project is aimed to develop new and functional Machine Learning models able to classify benign and malicious traffic. We can list the next main project goals:

- Train a wide range of models using different information and algorithms (Nearest

Neighbors Logistic Regression or Artificial Neural Network). We train an elevated number of models with the objective of finding those with the best performance.

- Evaluate the set of solutions developed with different statistical scores and determine the best models and algorithms for each attack type.
- Draw a set of conclusions about the utility of applying Machine Learning to Intrusion Detection and the best solution for each cyberattack.

1.4 Structure of this document

This document is divided into six chapters:

Chapter 1. The first chapter gives a brief introduction, explaining the context in which is situated the project subject. Furthermore, we explain the reasons that motivate this work and project goals. Finally, we describe the document structure.

Chapter 2. In this chapter, we review the main technologies implied in the development of this project with a special focus on the current picture of the machine learning applications to Cybersecurity. We also give a short description of the Dataset used, the Anaconda distribution and the Machine Learning algorithms.

Chapter 3. In this chapter, we carry out a deep description and analysis of the Dataset used.

Chapter 4. In this chapter we describe the methodology which we have worked. We explain the process of creating Machine Learning models, from the import of the Dataset to the validation of the model.

Chapter 5. This chapter summarizes all the results obtained from training the Machine Learning models.

Chapter 6. The last chapter explains the conclusions obtained from this project. We also review the main lines to keep working on this subject.

CHAPTER 2

Enabling Technologies

This chapter reviews the main technologies implied in the development of this project. Firstly, we explain the current picture of the machine learning applications to Cybersecurity. Then, we analyze the Intrusion Detection Evaluation Dataset (CICIDS2017) provided by the Canadian Institute for Cybersecurity (CIC) as well as the Python distribution used to write the code, Anaconda. Finally, we describe the different algorithms used to train the Machine Learning models.

2.1 Introduction

The need to protect information due to its high value has led both scientific community and companies to invest large amounts of money to improve protection measures of computers networks. New technologies have arisen to solve these complex problems, such as Big Data and Artificial Intelligence. These technologies have existed for a long time already. But now, thanks to the significant progress of computing, Big Data and Artificial Intelligence have meant a technological revolution.

In this chapter, we review the current scene of Cybersecurity. Furthermore, we explain the main technologies and tools used in this project. In section 2.2, we review the current situation of Cybersecurity and IDS analyzing the most popular IDSs and IPSs. In section 2.3, we review the Python distribution used to write the code, Anaconda, and its components. Finally, in section 2.4, we explain the Machine Learning algorithms which we have developed this work.

2.2 Cybersecurity and IDS

The Cybersecurity market is a business with a short existence but an expanding sector. According to Gartner[1], the Cybersecurity industry represented a global turnover of 62,540 million euros in 2015, a net profit up about 15% compared to the previous year [5].

Cybersecurity solutions are offered by many technology companies and each day this number increase. The set of software tools and devices tended by the business is wide and varied: Firewalls, Network Analysis, Vulnerability Scanners, Intrusion Detection Systems, Access Control... The trend to join these devices in order to simplify networks architecture result in Intrusion Prevention Systems. An IPS is a cybersecurity system designed to prevent intrusions that acts as a firewall blocking malicious packets. The IPS bases his behaviour on an Intrusion Detection System criteria. These devices can also act as a traditional firewall based on static rules. The IPS is placed in the edge of the company network acting as a gateway. All the inbound and outbound traffic passes through the IPS. Thus, the IPS checks the packet flow protecting the local network from the rest of the world.

The commercial IPS are physical devices designed and optimized to protect corporate networks. These tools are able to analyze large packet flows from some Gbps to hundred Gbps. The most popular and biggest companies offer IPS as one of their key products. These products are sold under the commercial name Next-Generation Intrusion Prevention

[1]Gartner: `https://www.gartner.com/en`

System (NGIPS). Some of the most popular IPS are the next:

- Fortinet FortiGate IPS
- Forcepoint NGFW
- IBM QRadar
- Trend Micro TippingPoint
- Palo Alto Networks NGFW
- Cisco Firepower

These products have a wide range of characteristics depending on necessities. Their two main features are the analyzed traffic volume and the available ports. The analyzed bandwidth goes from hundred Mbps for most simple IPSs, up to 200 Gbps for the most equipped ones. The number and type of ports are varied, normally IPSs include GE and RJ45 connectors.

The price of IPSs is very varied depending on their characteristic, furthermore, most cybersecurity and consultant companies rent the device for a period of time. The physical device cost from some thousands of dollars for the basic ones (i.e. Cisco Firepower 2110[2]) to several hundred thousand dollars for the best equipped. The rent of these devices plus protection services are in the same prices range crossing the million dollars for the most expensive options (i.e. Fortinet FortiGate 7060E[3]). Here we can observe a comparative analysis of some of these products by NSS Labs [10]

Thanks to the evolution and good performance of Machine Learning Algorithms and Data Mining, these technologies have become a really useful tool used in Intrusion Prevention Systems. ML and DM are really useful for anomaly detection. Plenty of studies about applying these techniques have been made in the last years. For example, [6] shows how to implement a Java-based IDS.

The software architecture under IDS is based on three main parts:

- **Packet capturer:** Able to capture all the flow packets of the network. For a simple IDS, the packet captured can come from a mirroring port. However, for IPS all the traffic must cross the device and no other path.

- **Database:** Usually a MySQL Database where packet flow information is stored. This database can be also used to save alarms and logs.

[2]Cisco Firepower 2110: https://www.connection.com/product/cisco-firepower-2110-ngfw-appliance-1ru/fpr2110-ngfw-k9/33873297

[3]Fortinet FortiGate 7060E: https://www.zones.com/site/product/index.html?id=105244238

- **Intrusion Detector:** The main actor of the IDS that implements the logic. The detector is based on a set of rules that can be defined by a Machine Learning model.

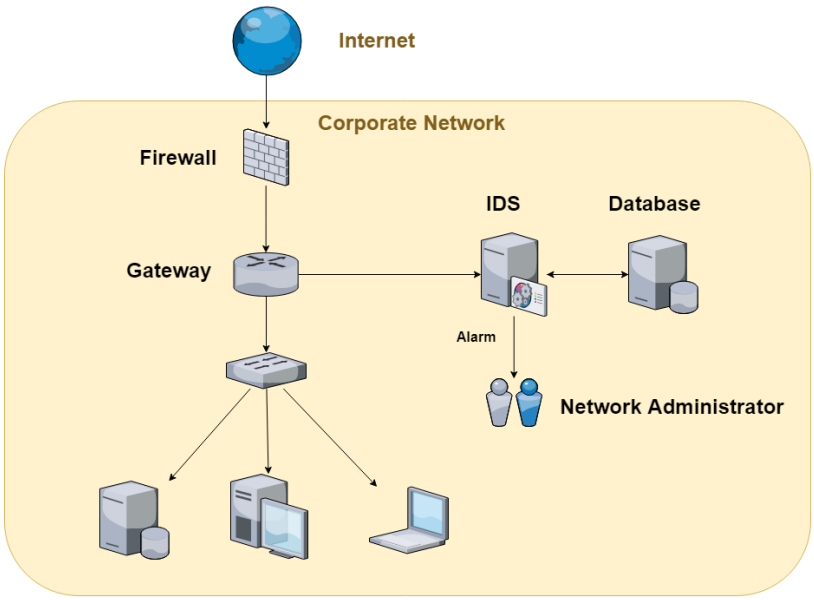

Figure 2.1: Intrusion Detection System in a Corporate Network

There are some comparative studies about the existing software Intrusion Detection Systems. In the next article [9], we can observe a comparison of most common IDSs such as Bro, Snort, Network Flight Recorder (NFR), Suricata or The Dragon IDS. As we can observe, most IDSs are signature based, only Bro (Zeek) is anomaly based. Signature based IDSs are effective against known attacks, however, zero-day exploits are invisible to them. To detect zero-day attacks we need an IDS based on anomalous behaviours.

2.3 Anaconda Distribution

Anaconda[4] is a free open source Python/R distribution oriented to Machine Learning and Data Science. This distribution allows to train and develop Machine Learning models. In order to manage packets and libraries, Anaconda uses Conda. Conda allows to easily install different packets and libraries from repositories. In the development of this project, we use the next software packages for Anaconda:

[4]Anaconda Distribution: `https://www.anaconda.com/distribution/`

2.3.1 Jupyter Notebooks

Jupyter[5] is a web application that allows to write code and plain text and see visualizations in a web browser. The code is written and executed in short blocks and the output is directly shown inline. Jupyter supports more than 40 programming languages including Python and R. It is also oriented to Machine Learning and Data Mining with the ability to manage large amounts of data.

2.3.2 Pandas

Pandas[6] is an open source Python Data Analysis Library. This library is designed to analyze and model data with good performance and productivity. The DataFrame object is the main element where Datasets are loaded. The DataFrame can be created by reading from other data structures such as CSV, Microsoft Excell and text files or SQL databases. Once imported, it can be modified, sliced, merged and joined with other datasets, grouped by columns and rows allowing to easily and quickly reshape the data. Pandas also includes statistical tools in order to acquire information from the DataFrame.

2.3.3 Scikit-learn

Scikit-learn[7] is an open source Python library oriented to Machine Learning. This library includes several classification and regression algorithms such as Nearest Neighbors, Naive Bayes, Gaussian Process Regression and Decision Trees. Furthermore, this library gives some useful tools for preprocessing data, dimensionally reduction, and model selection, like Feature Selection, Cross Validation or Stratified Selection. Scikit-learn is designed to interoperate with SciPy.

2.3.4 SciPy and NumPy

SciPy[8] is a python eco-system oriented to scientific computing. One of its core packages used for this project is NumPy. NumPy[9] is a mathematical library designed and optimized to work with high dimensional arrays. That is why this library is really useful for Machine Learning applications.

[5]Jupyter Notebooks: `https://jupyter.org/`
[6]Pandas: `https://pandas.pydata.org/`
[7]Scikit-learn: `https://scikit-learn.org/stable/`
[8]SciPy: `https://www.scipy.org/`
[9]NumPy: `http://www.numpy.org/`

2.4 Machine Learning Algorithms

The algorithms used in this project to train models are the most common applied up-to-day. In this section, we give a short explanation of the different Machine Learning Algorithms. Normally, these algorithms are classified into three different groups: supervised, unsupervised and semi-supervised. For this project, we use the supervised methods due to the Dataset structure. Supervised algorithms are trained with two data groups, the input X and the output Y. The target of the training is to find the best approach to the function f that meets Y = f (X).

2.4.1 k-Nearest Neighbors Algorithm

The k-Nearest Neighbors is a supervised classification learning method. This algorithm is based on obtaining the k nearest points to the new input that has to be classified. The input is classified with the same label of the k majority nearest points. Each row of the Dataset represents a point, and each cell or feature a coordinate of the position vector. In order to get the distance between points, we can use different algorithms, Euclidean Distance is the most common. Besides its apparent simplicity, the k-NN performance is quite good even when the points distribution is very irregular. However, when the volume of the Dataset growth, the computational cost increases considerably, that is the main drawback of this algorithm.

2.4.2 Logistic Regression Algorithm

The Logistic Regression Algorithm is another supervised classification learning method designed for binary classification. Regression is a statistical process that estimates the relationship between the independent variables and the dependent variable. The logistic function or sigmoid assigns the odds of belonging to a class to the input data. Sigmoid is the reverse function of the logit function. The logit function is the logarithm of the odds of belonging to a class:

$$logit(p_i) = ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 \cdot x_{1i} + ... + \beta_k \cdot x_{ki}$$

Where p_i is the probability of belonging to a group. As we can observe the logit function can be approximated to a linear function. That is why Logistic Regression is a linear model. The objective of the algorithm is to find the values of $\beta_0, ..., \beta_k$. This algorithm draws the k dimensional plane that better separates the elements of the binary class, where k is the number of independent variables.

2.4.3 Naive Bayes Algorithm

The Naive Bayes is a supervised classification learning algorithm based on the Bayes theorem. The Bayes Theorem relates the probability of an event with the occurrence of another related event. This theorem is expressed mathematically as follows:

$$p(C/F_1, ..., F_n) = \frac{p(C) \cdot p(F_1,...,F_n/C)}{p(F_1,...,F_n)}$$

Where the conditional probability P(C/F1, ..., Fn) is the probability of occurring C (the class) given that F1, ..., Fn has occurred and F1, ..., Fn are the features of the input data. If the volume of data increase, the computational cost growths exponentially, that is why this theorem is simplified under above the naive or independence assumption:

$$p(C/F_1, ..., F_n) = \frac{1}{Z} \cdot p(C) \prod_{i=1}^{n} p(F_i/C)$$

Where Z is a constant value which depends on F1, ..., Fn.

2.4.4 Multilayer Perceptron Algorithm

The Multilayer Perceptron or MLP is a class of Artificial Neural Network. Artificial Neural Networks are a computational model inspired by its biological analogous. A Neural Network is based in a set of neurons interconnected through links. Neurons are grouped in layers. The input data are introduced into the first layer and processed by neurons. Then, the modified data goes to the next layer. The information goes through the Neural Network generating at the output layer an outcome as a result. The layers between the input and the output layers are named hidden.

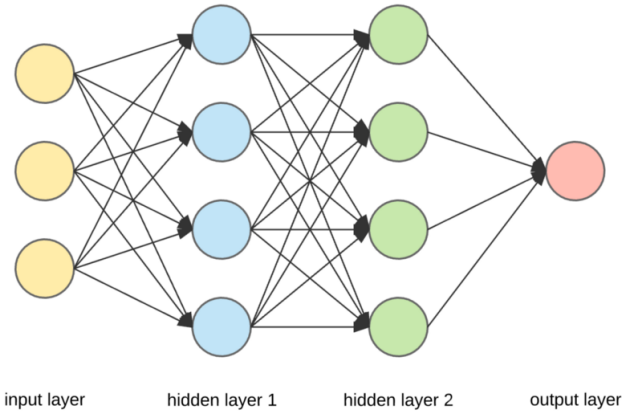

Figure 2.2: Neural Network Architecture [2]

Each neuron has a number of entries. The information that gets in the neuron comes from the input data or from the previous layer. The value of each input is multiplied by a weight, then, the weighted inputs are added. Finally, a function is applied to the weighted sum. This function, also named activation function, modifies and limits the value giving as result an output. The outcome is sent to the next group of neurons.

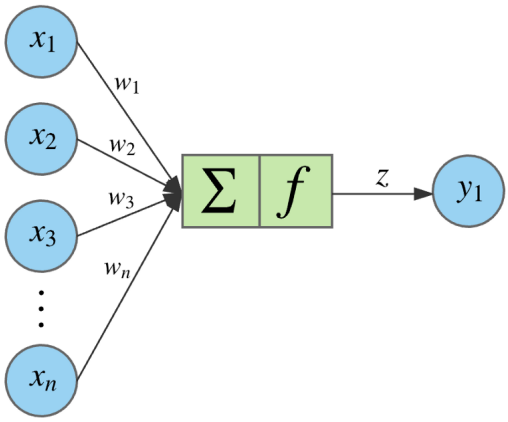

Figure 2.3: Operation of a Neuron [2]

Neurons have the ability to change the input weights. This way the Neural Network can learn if we train it with a set of input data and the expected output. This process is named backpropagation. Herein lies the power of Neural Networks and implies that it does not need to be preprogrammed to operate correctly.

The Multilayer Perceptron consists of at least three neural layers (input, hidden and output layer). It is designed to supervised learning with backpropagation training. With a labelled dataset we can train Neural Networks to predict the label of new inputs. In this project, we use four different activation function: Identity function, Logistic function or Sigmoid, Hyperbolic tangent and Rectified Linear unit function.

CHAPTER 3

Dataset Analysis

In this chapter, we analyze in depth the Dataset used in this project. This is the most important element when it comes to training Machine Learning Models. This Dataset is the Intrusion Detection Evaluation Dataset [3] (CICIDS2017) provided by the Canadian Institute for Cybersecurity[1] *(CIC). Additionally, we review the different attacks performed and how they are structured. Finally, we carry out a exploratory data analysis.*

[1]Canadian Institute for Cybersecurity: `https://www.unb.ca/cic/about/index.html`

3.1 Introduction

The data used to train models in Machine Learning processes is the most important element because algorithms learn from the available information. Cybersecurity Datasets designed to detect anomalous behaviours are elaborated with the packets captured from the network traffic flow.

There are two types or levels of traffic information abstraction: Packet level and flow level. The packet level or first level collects every packet interchanged between machines. The flow level or second level, defined by Cisco as Netflow is a network protocol that provides the ability to collect IP network traffic as it enters or exits an interface [15]. Netflow defines a flow as a sequence of packets that have the same seven attributes: ingress interface, source IP address, destination IP address, IP protocol, source port, destination port, and IP type of service.

There are only a few Datasets for Cybersecurity purpose and most of them are outdated or its data volume is too small, so they are practically useless. However, the Canadian Institute for Cybersecurity developed in 2017 a Dataset designed to train Intrusion Detection Systems Models.

The Canadian Institute for Cybersecurity[2] (CIC) is a multidisciplinary institution dedicated to research, development and divulgation in social sciences, business, computer science, engineering, law and science. The institute is part of the University of New Brunswick. In this project, we use the Intrusion Detection Evaluation Dataset (CICIDS2017) [3] provided by the CIC.

In this chapter, we review the Dataset used in this project. In section 3.2, we describe how the Dataset is generated and the information collected. In section 3.3, we explain the Dataset structure. Finally, in section 3.4, we perform an exploratory data analysis with the purpose of understanding the data and finding the most relevant information. Due to the limited space, we only show the analysis of one attack type. In Appendix C we show the list of features collected from the traffic flow.

3.2 Dataset Description

As we mentioned before, the Dataset used for this project is the Intrusion Detection Evaluation Dataset [3] (CICIDS2017). This flow level Dataset consists of benign traffic and the most common actual cyber attacks.

[2]Canadian Institute for Cybersecurity: `https://www.unb.ca/cic/about/index.html`

The traffic is captured and labelled with CICFlowMeter, a network traffic biflow generator and analyzer developed by the Canadian Insitute for Cybersecurity. This analyzer logs all the information of every packet flow.

The human traffic, in order to generate realistic behaviours, is simulated according to Gharib et al. 2016 [7]. The Dataset collects the traffic generated by the abstract behaviour of 25 humans.

In order to create a real network with real attacks, the CIC Dataset elaboration is based on the next criteria:

- **Complete networking configuration:** with Firewalls, Routers, Switches and Modems and three different Operating Systems, Ubuntu, Windows and OS X.
- **Complete traffic, interactions, and protocols:** 12 machines in the victim network in two different LANs with Internet access. The use of the most common protocols such as HTTP, HTTPS, FTP, SSH, and email protocols.
- **Complete capture:** All flow packets are captured by using mirroring ports.
- **Labelled Dataset:** Every flow is labelled with Bening or Attack type.
- **Attack Diversity:** With the most common actual cyber attacks, Web-based, Brute force, DoS, DDoS, Infiltration, Bot and Port Scan.

3.3 Dataset Structure

The collected information covers a period of time of a working week (from Monday to Friday). The capture gathers the normal activity of 25 workers plus the different attacks. The structure of the attacks performed is the next:

- **Monday:**
  - **Normal Activity**
- **Tuesday:**
  - **Brute Force attacks:**
    - FTP-Patator (9:20–10:20 a.m.)
    - SSH-Patator (14:00–15:00 p.m.)
  - **Normal Activity.**

- **Wednesday:**
  - **DoS / DDoS attacks:**
    - DoS slowloris (9:47 – 10:10 a.m.)
    - DoS Slowhttptest (10:14 – 10:35 a.m.)
    - DoS Hulk (10:43 – 11 a.m.)
    - DoS GoldenEye (11:10 – 11:23 a.m.)
  - **Normal Activity.**
- **Thursday:**
  - **Web attacks:**
    - Web Attack – Brute Force (9:20 – 10 a.m.)
    - Web Attack – XSS (10:15 – 10:35 a.m.)
    - Web Attack – Sql Injection (10:40 – 10:42 a.m.)
  - **Infiltration and Dropbox download:**
    - Meta exploit Win Vista (14:19 and 14:20-14:21 p.m.) and (14:33 -14:35)
    - Infiltration – Cool disk – MAC (14:53 p.m. – 15:00 p.m.)
    - Win Vista (15:04 – 15:45 p.m.)
  - **Normal Activity.**
- **Friday:**
  - **Botnet attacks:**
    - Botnet ARES (10:02 a.m. – 11:02 a.m.)
  - **PortScan (Nmap):**
    - Firewall Rules on (13:55–13:57, 13:58–14:00, 14:01–14:04, 14:05–14:07, 14:08-14:10, 14:11–14:13, 14:14–14:16, 14:17–14:19, 14:20–14:21, 14:22–14:24, 14:33–14:33, 14:35-14:35)
    - Firewall rules off (14:51-14:53, 14:54-14:56, 14:57-14:59, 15:00-15:02, 15:03-15:05, 15:06-15:07, 15:08-15:10, 15:11-15:12, 15:13-15:15, 15:16-15:18, 15:19-15:21, 15:22-15:24, 15:25-15:25, 15:26-15:27, 15:28-15:29)
  - **DoS / DDoS attacks:**
    - DDoS LOIT (15:56 – 16:16)
  - **Normal Activity.**

The information generated is saved in eight different files, one per attack. Each file is a set of rows and columns filled with values. A row represents a flow. A flow as we already

defined is a sequence of packets that have the same seven attributes: ingress interface, source IP address, destination IP address, IP protocol, source port, destination port, and IP type of service. It means every interaction between two computers in order to carry out a service. Each column represents an attribute of a flow. The complete features list is displayed in Appendix C.

The files are CSV format which implies they are ready to be imported by the Pandas Library. The dataset is made up of eight files:

- Monday-WorkingHours-Benign
- Tuesday-WorkingHours-BruteForce
- Wednesday-WorkingHours-DDos
- Thursday-WorkingHours-Afternoon-Infiltration
- Thursday-WorkingHours-Morning-WebAttacks
- Friday-WorkingHours-Morning-Botnet
- Friday-WorkingHours-Afternoon-DDos
- Friday-WorkingHours-Afternoon-PortScan

3.4 Exploratory Data Analysis

Collecting information from a Dataset before the training process is especially helpful because it helps us to understand the Dataset. This task also allows us to find the most relevant features, redundant information and patterns. With this analysis, we can balance the Dataset properly. Furthermore, with this information, we can fix and correct missed and wrong values.

To achieve this, we select the most correlated features to the label. Then we group rows by variables and label. Furthermore, we plot histograms for those features with a wide range of values or continuous values This way, we can see the information distribution.

In this document, we only show the exploratory data analysis for only one Dataset due to the limited space. We analyze the Thursday-WorkingHours-Morning-WebAttacks Dataset with the purpose of obtaining information before preprocessing it.

First of all, we obtain the number of rows, this Dataset has originally the next number of entries or flows:

Label	Number of rows
Benign	168.186
Web Attack	2.180

Now, we group the rows by Destination Port Number and Label. We only display the Destination Ports of Benign flows that appear more than 200 times due to the limmited space:

Label	Destination Port	Number of rows
Benign	22	1035
	53	76.545
	80	18.682
	88	514
	123	3.746
	137	689
	443	35.833
Web Attack	80	2.180

As we can observe, all Web Attacks are directed to port 80. Therefore we can select only those flows directed to port 80. This way we simplify the Dataset without losing relevant information. Now, the dataset has the next number of flows:

Label	Destination Port	Number of rows
Benign	80	18.682
Web Attack	80	2.180

Next, we analyze the most correlated features to the Label. We display the distribution of benign and malicious flows depending on each variable. The distribution of variables with discrete values is shown in a table (PSH Flag Count, Down/Up Ratio, Min Seg Size Forward). Those variables with continuous or a wide range of possible values are displayed in histograms (Init Win Bytes Backward, Init Win Bytes Forward):

- **Init Win Bytes Backward:** The number of bytes sent in the initial window in the

backward direction. It is the most relevant feature, next we can see the distribution of benign and malicious flows depending on Init Win Bytes Backward value:

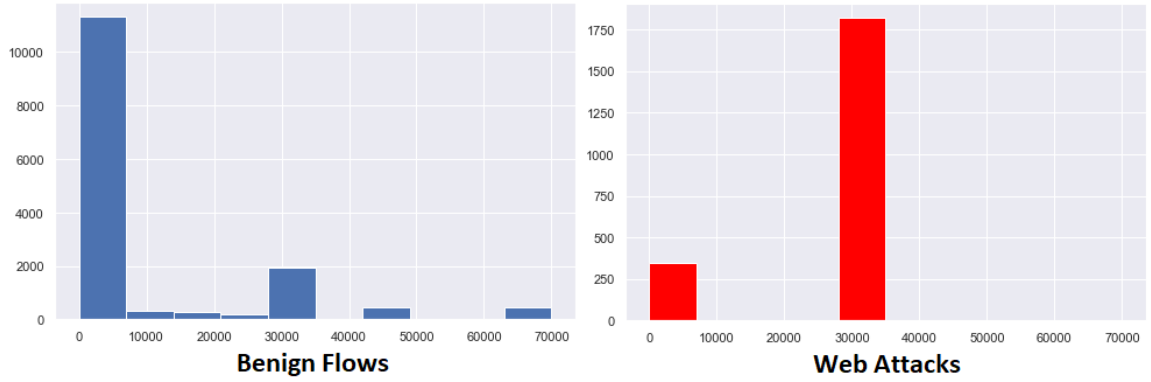

Figure 3.1: Init Win Bytes Backward Histogram

- **PSH Flag Count:** The number of packets with PUSH flag. In the next table we can see the PSH Flag Count distribution related to the Label:

Label	PSH Flag Count	Number of rows
Benign	0	6.051
	1	12.631
Web Attack	0	175
	1	2.005

- **Init Win Bytes Forward:** The number of bytes sent in the initial window in the forward direction. Next we display its histogram:

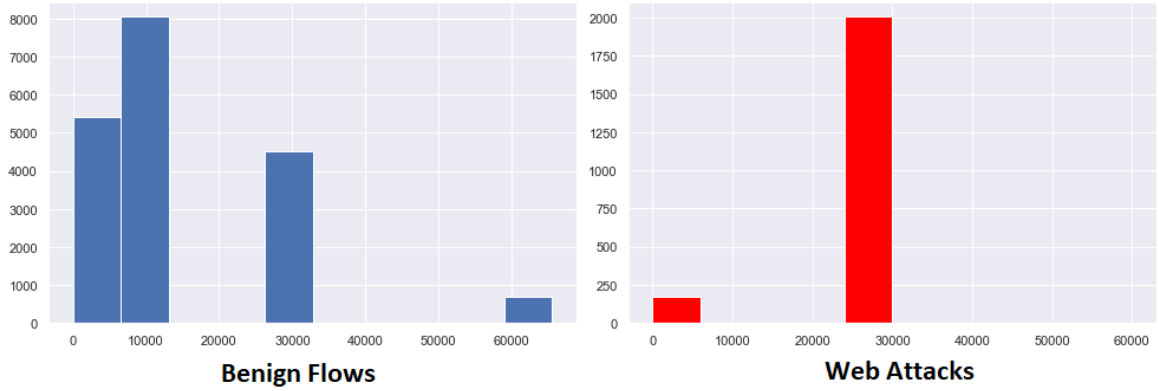

Figure 3.2: Init Win Bytes Forward Histogram

- **Down/Up Ratio:** Download and upload ratio. This ratio is rounded to the unit. Next, we show in a table the distribution of benign and malicious flows related to Down/Up Ratio:

Label	Down/Up Ratio	Number of rows
Benign	0	11.241
	1	7.348
	2	89
Web Attack	0	1.942
	1	238

- **Min Seg Size Forward:** Minimum segment size observed in the forward direction. In the next table, we display the number of flows grouped by Min Seg Size Forward value:

Label	PSH Flag Count	Number of rows
Benign	20	12.916
	28	13
	32	5.747
	40	6
Web Attack	32	2.180

As we can observe with these features we gather some interesting information. Grouping rows by variables and the label bring useful patterns and distributions. With the purpose of balancing the Dataset, we can delete irrelevant information based on the Destination Port. As we expected, all malicious flows are directed to port number 80 (HTTP Port), therefore, we can delete the flow directed to other ports. This way we also simplify the bandwidth analyzed directing to the classifier only the HTTP traffic.

We have analyzed most correlated features to the label. These variables are the most useful in the classification task. In fact, with only Init Win Bytes Backward we can improve significantly the accuracy of the classifier due to the separated distribution of benign and malicious flows. The rest of the features also show a high polarization depending on the label.

CHAPTER 4

Methodology

In this chapter, we explain the work methodology followed to develop this project. We explain the main phases in which is divided a normal Machine Learning research. First of all, we outline the data import and preprocessing. Once the data is ready, we carry out the training process with different algorithms and configurations. Finally, we explain the different metrics used to evaluate models.

4.1 Introduction

The methodology followed in a Machine Learning research is divided into three main parts: Preprocessing, Training and Evaluation. The first part is the preparation of data for the training process. It implies that data must be correct and appropriate for the training algorithms. Next, the training process consists of the development of Machine Learning models with different algorithms and configurations in order to generate a big range of prototypes. It means that the training process is iterative. Once we get enough models, we must evaluate them with the goal of obtaining the best configuration for each one and the model that best solves the problem.

In section 4.2, we describe how to prepare the original information for training. In section 4.3, we explain how we reduce the volume of data by selecting only the most relevant variables. In section 4.4, we expose the methodology followed to train models. Then, in section 4.5 we explain the scores used to rate the model performance.

4.2 Preprocessing

Preprocessing as we said before is the first part of any Machine Learning project. We must prepare the data for training. Datasets usually have missed or incorrect values. We need to check that all data are appropriate for the training algorithms. With the Intrusion Detection Evaluation Dataset (CICIDS2017) we faced four preprocessing tasks:

- **Filling empty or null cells:** Although this is an uncommon incident, some cells of the Dataset are empty. If a cell is empty, we cannot use that row in the training process. In these cases, we fill the empty feature with the median obtained from the rest of the elements. By doing so, we do not need to delete an entire flow, but the row is valid to train with the empty cell as blank information.

- **Fixing object types:** Variation of objects types is another issue found in the original data. More than an error in the original information, this is due to the combination of numbers and words in the same column or feature. This problem is solved by setting a single object type for the column. Thus we need to change the elements or cells with a different object type to the chosen type, usually numeric (float type).

- **Balancing:** Some Datasets are unbalanced. It means that there is a big difference between the number of elements of each class. Next, we describe the actions carried out in the Datasets with the objective of balancing benign and malicious flows:

- **Friday-WorkingHours-Afternoon-DDos Dataset:** This Dataset is overall balanced, the number of flows of each class is the next:
  - Benign 97.718
  - DDoS 128.027
- **Friday-WorkingHours-Afternoon-PortScan Dataset:** The PortScan Dataset is also balanced. Next, we show the number of malicious and benign flows:
  - Benign 127.537
  - PortScan 158.930
- **Thursday-WorkingHours-Afternoon-Infiltration Dataset:** The Infiltration record has only 36 Attack flows and 288.566 Benign. However, there is a simple solution to solve this difference. Due to all Infiltration attacks are directed to 444 port, we delete all flows which their Destination Port is different from port 444. This way, the Dataset reads as follows:
  - Benign: 98
  - Infiltration: 36
- **Friday-WorkingHours-Morning-Botnet Dataset:** This Dataset has originally: 189.067 Benign flows and 1.966 Bot flows. However, in this case, there is no feature repeated in the Bot flows. Therefore, in order to balance the Dataset, we randomly select two thousand Bening rows. This way, the final Dataset has the next number of rows:
  - Benign: 2.000
  - Bot: 1.966
- **Thursday-WorkingHours-Morning-WebAttacks Dataset:** This Dataset has originally 168.186 Benign flows and 2.180 Web Attack flows. In order to balance it, we have followed both strategies described in the previous cases. First, we choose only the flows which their Destination Port is 80 due to all Web Attacks are directed to that port. Next, as the Dataset still has more benign than malicious communications, we randomly select a number of Benign rows that matches with the number of Web Attacks. By doing this, the Dataset reads as follows:
  - Benign: 2.200
  - Web Attack: 2.180
- **Tuesday-WorkingHours-BruteForce Dataset:** This Dataset has 189.890 Benign flows and 10.109 malicious flows (FTP-Patator and SSH-Patator). The procedure followed is the same as the previous Dataset. We choose only the

flows directed to 21 and 22 Ports. There are 12.122 flows directed to those ports. Next, to balance the final Dataset, we select only 2.000 malicious flows.

  * Benign: 2.014
  * BruteForce: 2.000

- **Checking:** Finally, once we have filled empty cells, corrected object types and balanced the dataset, we check that the data is appropriate to train if so, we go to the next phase.

4.3 Dimensionality Reduction

Before we start the training process, we have to choose which features from the original data we will use. In many Machine Learning projects, the amount of data and variables is immense. This could cause two issues. Firstly, the computational cost is too high and secondly, some variables could introduce noise or confusion in the classifier. In order to avoid these problems, the number of features selected is reduced. We choose the most relevant features and discard those that do not have information. In one hand, the chosen variables are highly correlated with the Label and in the other hand, the discarded variables are almost constant or highly correlated with other feature (it means that two variables have the same information).

Scikit-learn library has a feature selection or dimensionality reduction utility. We use the SelectKBest tool from this module. The SelectKBest utility chooses the k most relevant variables from a Dataset based on univariate statistical tests.

4.4 Training

The training process is the longest in terms of time. It consists of creating a set of models with different algorithms in order to find the best configuration for each attack type. The number of models trained in this project is about 25 for each algorithm configuration. It means a total of 1350 models. A model is a classifier based on a mathematical representation of the problem. This set of mathematical relations is generated processing the original data with a machine learning algorithm.

4.4.1 K-fold cross-validation

Before we start training, we have to split the Dataset into training and test parts. This division is done with the objective of training and test data are different. If test and training information are equal, the model would know the Label of test input a priori. The training segment is usually between 60 and 90 per cent of the Dataset and the test section is between 40 and 10 per cent of the data respectively. In our project, the training set is 90 per cent and the test set is 10 per cent of the original Dataset.

If we train and validate with only one division we can obtain an incorrect result due to the distribution of data in the training and test sections. To avoid this problem we use cross-validation. Cross-validation process consists of the next 5 steps:

- The Dataset is split into n parts also known as folds.
- Every iteration a model is generated with n - 1 sections of the original data
- The model is evaluated with the test part or remaining section.
- The training and test steps are repeated n times with the next fold.
- Once all iterations have finished, the average performance scores of every iteration are calculated.

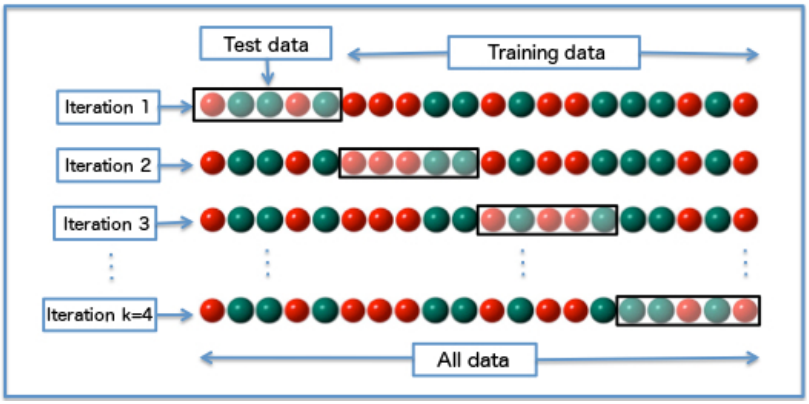

Figure 4.1: Diagram of K-Fold cross-validation [4]

In spite of the number of iterations or folds is not a standardized parameter, the 10 fold cross validation is the most commonly used [11]. In our project, we use the StratifiedKFold utility from Scikit-learn as cross-validator with 10 folds.

4.4.2 Algorithm configuration

In order to generate a wide range of models, we use four different algorithms with different configurations. The algorithms used to train the models are the next:

- **k-Nearest Neighbors:** This algorithm as we described in section 2.4.1, obtains the k-nearest points to the new element to be classified. These elements are labelled with the same label of the k majority nearest points. The number of neighbours used is 3, 5 and 10. We use the sklearn.neighbors.KNeighborsClassifier to train models with the k-NN algorithm.

- **Logistic Regression:** This algorithm described in section 2.4.2, assigns the odds of belonging a class based on a logistic function or sigmoid after a linear equation is applied to the input data. The target of the training is to find the best parameters of the linear equation. We use the sklearn.linear_model.LogisticRegression to apply this algorithm.

- **Naive Bayes:** As we explained in section 2.4.3, this algorithm is based on the Bayes theorem. To train models with this algorithm we use the sklearn.naive_bayes.GaussianNB module.

- **Multilayer Perceptron:** This algorithm was described in section 2.4.4. The Multilayer Perceptron is a Neural Network that has the ability to learn. It is possible thanks to the Neural Network changes its configuration. This adaptation is produced by the processing of input data and its expected result. The activation function defines the output of a neuron given an input. In our project we use four different activation functions. Next, we describe the different activation functions where x is the value at the neuron input:

  - **Identity function:** Is the function that returns as result the same value as the input. The equation is:

$$f(x) = x$$

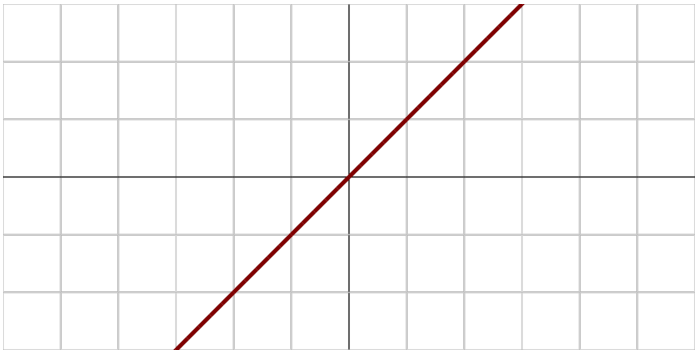

Figure 4.2: Identity function [1]

– **Logistic function or sigmoid:** The general equation of the logistic function is:

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

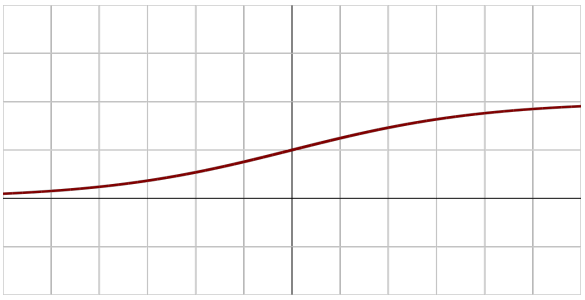

Figure 4.3: Logistic function or sigmoid [1]

– **Hyperbolic tangent:** The mathematical expression of hyperbolic tangent is:

$$f(x) = tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

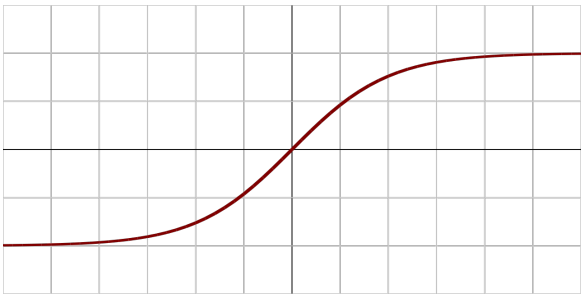

Figure 4.4: Hyperbolic tangent function [1]

– **Rectified Linear unit:** This function returns the positive part of the input:

$$f(x) = \begin{Bmatrix} 0 \ for \ x < 0 \\ x \ for \ x \geq 0 \end{Bmatrix}$$

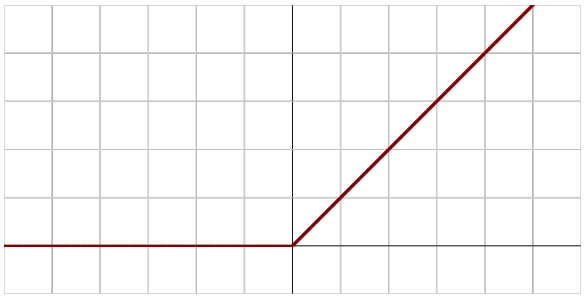

Figure 4.5: Rectified Linear function [1]

4.4.3 Feature selection

With the objective of finding the best models, we generate them with a wide range of variables using the SelectKBest described previously. In order to find the most proper number of features, we divide the training process into two phases.

- In the first part, we generate a model for each set of variables, from 5 to 75, five-by-five. This allows us to find the features range where the algorithm reaches its best result.
- Once located, we train again around the maximum discovered before with one variable jumps. This way we find the number of features with the best performance for each algorithm.

4.5 Evaluation

The evaluation of the model performance is done with the test part. This set is passed to the model as input data without the Label. The model classifies every input or row into Benign or Attack. This classification is based on the logic learnt in the training process. Once the model has labelled the test part the result is checked with the original Labelled information. The model performance is evaluated with four different metrics based on the next statistical results:

- **True Positive (TP):** The element is correctly identified.
- **True Negative (TN):** The element is correctly rejected
- **False Positive (FP):** The element is incorrectly identified
- **False Negative (FN):** The element is incorrectly rejected.

Note that in a binary classification these statistical measures are cross-connected:

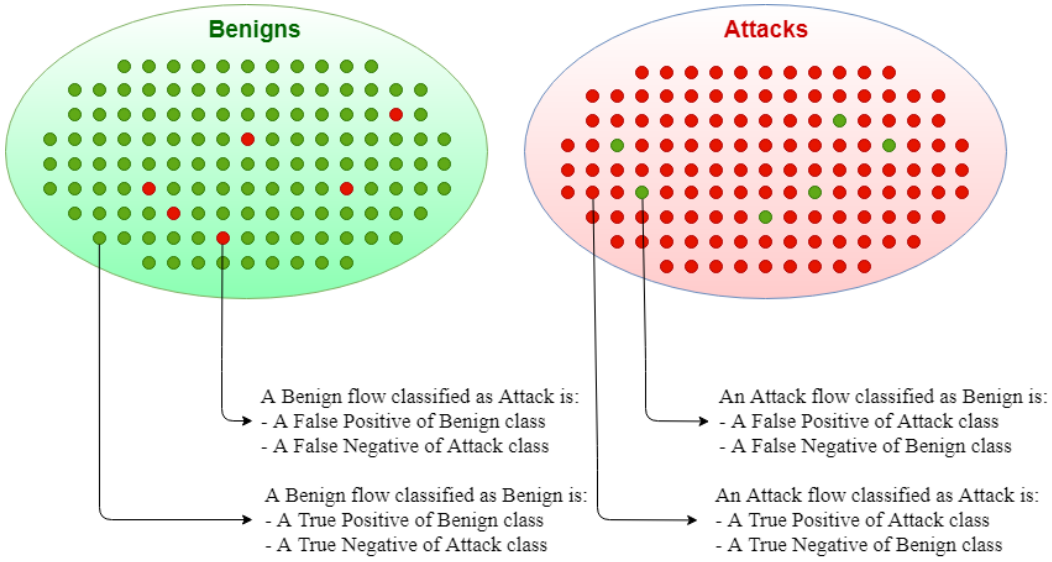

Figure 4.6: Statistical results

The metrics used to evaluate the model performance are based on the statistical results previously described. In our project we employ the next metrics:

- **Accuracy:** Accuracy is defined as the sum of true positives plus true negative divided by the total population. Intuitively is the number of well-predicted elements divided by the number of elements:

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

- **Precision or True Positive Rate:** Precision is calculated as the number of true positives divided by true positives plus false positives. In other words, the number of well-predicted elements divided by the total predicted elements:

$$Precision = \frac{tp}{tp + fp}$$

- **Recall:** Recall is described as the number of true positives divided by the sum of true positives plus false negative. It means the number of well-identified elements divided by the total positive elements:

$$Recall = \frac{tp}{tp + fn}$$

- **F1 Score:** It is a statistical measure that combines both precision and recall. The f1 score is defined as the harmonic average of precision and recall. It is calculated as follows:

$$f1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

In our model evaluation, we use the f1 score as the most significant rating. We try to optimize the model in order to get the best f1 score, where 1 is the best value and 0 the worst.

Once training has finished, we obtain a visual representation of all scores. This graphic representation shows in a spider chart the model scores (precision, recall and f1-score) of each algorithm configuration and the number of features used. Then, we compare the best models of each algorithm based on the number of variables and the f1-score achieved. Then we show in a table all scores obtained by the best models and finally we select the best algorithm. A deep analysis of the models can be found in Appendix C.

The training process requires a long computing time. It takes from several minutes for simple Datasets to hours for long Datasets to train one model. The algorithm used also has a big impact on computing time. While the fastest algorithms are Logistic Regression and Naive Bayes, the k-Nearest Neighbors and Multilayer Perceptron are heavier. That is why we save the models, scores, configuration and graphic representation in order to optimize and save time.

CHAPTER 5

Results

In this chapter, we show the results obtained in this project. For each attack type (Dataset) we have trained a wide collection of models. Once the training process is finished, we evaluate the models following the methodology described before. In this section, we show the configuration with which the algorithm performance reaches its best values. Furthermore, we display the most relevant variables for each attack type.

5.1 Introduction

In this chapter, we show the results obtained in this project. These results are the consequence of training and evaluating a broad set of models. We have developed classifiers able to detect benign and malicious flows. The malicious flows are generated by the attacks performed and collected in the Datasets: DDoS, Port Scan, Infiltration, Botnet, Web Attacks and Brute Force. As we described in chapter 4, we generate a wide range of models in the training process with the purpose of finding the best algorithm configuration. Each configuration breeds a different model. Next, we describe the set of algorithm configurations used depending on two parameters:

- **Number of features:** The number of variables used for training is determined in two phases:
  - Firstly, we generate a model for each set of variables, from 5 to 75, five-by-five. This way, we locate the range of features where the algorithm reaches its best performance.
  - Next, we train again around the maximum discovered before with one variable jumps. This allows us to find the number of features where the algorithm obtains the best scores.

  The features are selected with the SelectKBest. This function is based on the correlation between the variables and the Label. It means that the number of features is selected in the same order for each attack type regardless of the algorithm used.
- **Algorithm parameters:** In this project, we use four algorithms (k-Nearest Neighbors, Logistic Regression, Gaussian Naive Bayes and Multilayer Perceptron). K-Nearest Neighbors and Multilayer Perceptron have one parameter to be adjusted. We have to determine the number of neighbours in the k-NN algorithm. In our project, we train with 3,5 and 10 neighbours. In MLP we have to choose the activation function. We use four different activation functions: Identity function, Sigmoid, Hyperbolic tangent and Rectified Linear unit. These two attributes are applied to every set of variables. This way, the number of combinations is multiplied.

We have trained an overall of 1350 models, covering this way the entire spectrum of information. In this chapter, we only show the sets with the best performance based on the f1-score and the number of features used. In Appendix D we provide a deeper performance analysis of the models developed. Next, we stipulate which model is the best and the list of most important features. Finally, we draw a short conclusion.

5.2 DDoS

DoS or Denial of Service is a cyber-attack directed to a computer or computer network in which the attacker tries to deny a service or a resource in the targeted network. This way, other users can not access the service. This attack is typically executed performing an elevated number of requests to the committed server. This causes the attacked network failure because of the high bandwidth consumed or the committed computer overload. Thus, the resource or service is denied.

DDoS or Distributed Denial of Service is a DoS attack type characterized by the number of malicious computers implied. In Distributed DoS, the requests directed to the assaulted network came from different computers. These machines also called bots are usually infected by the original attacker. A DDoS is harder to defend because of the elevated number of malicious computers implied.

5.2.1 Results

The models developed in this project have obtained the performance described below detecting DDoS attacks.

- **k-Nearest Neighbors**: K-NN algorithm algorithm reaches its best performance with 3 neighbours and 12 features.

- **Logistic Regression**: The best configuration for Logistic Regression algorithm is using 6 variables.

- **Gaussian Naive Bayes**: Gaussian Naive Bayes algorithm obtains its best scores with 6 features.

- **Multilayer Perceptron**: The best configuration for Multilayer Perceptron algorithm is using 6 variables with Logistic activation function.

The next graphic representation compares the different algorithms performance using its best configuration. We display the f1-score obtained and the number of features needed to reach that score:

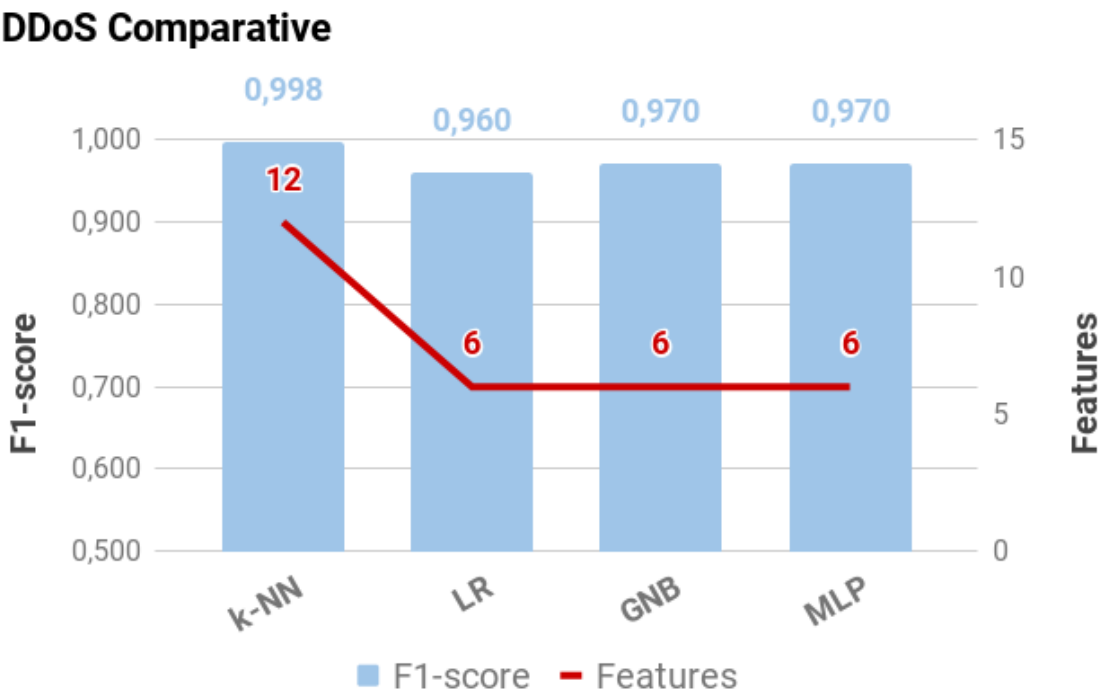

Figure 5.1: DDoS Algorithm Performance Comparative

Down below, we show a comparative table with the best rating obtained for each algorithm. The displayed scores are Accuracy, Precision, Recall and f1-score. In addition, the number of features to reach those scores is also shown.

Algorithm	Features	Accuracy	Precision	Recall	f1-score
k-NN	12	0.998 (+/- 0.002)	0.998 (+/- 0.002)	0.998 (+/- 0.002)	0.998 (+/- 0.002)
LR	6	0.96 (+/- 0.04)	0.96 (+/- 0.04)	0.96 (+/- 0.04)	0.96 (+/- 0.04)
GNB	6	0.93 (+/- 0.09)	0.94 (+/- 0.07)	0.93 (+/- 0.09)	0.93 (+/- 0.09)
MLP	6	0.97 (+/- 0.15)	0.97 (+/- 0.10)	0.97 (+/- 0.15)	0.96 (+/- 0.17)

Finally, we show the list of 10 most relevant Features in DDos detection:

- Destination Port
- Total Length of Fwd Packets
- Fwd Packet Length Max
- Bwd Packet Length Max
- Bwd Packet Length Min
- Bwd IAT Total
- Bwd IAT Std
- Bwd IAT Max
- Min Packet Length
- Max Packet Length

5.2.2 Conclusions

The best algorithm detecting DDoS attacks is the k-Nearest Neighbors using only 12 features with 0.998 points in f1-score and an accuracy of 99.8%. This is a great performance with a ratio of only 0.2% of false alarms plus undetected malicious flows.

It is important to highlight that destination port and packet size (Total Length of Fwd Packets, Fwd Packet Length Max, Bwd Packet Length Max...) are as expected the most important information detecting DDoS attacks.

5.3 Port Scan

A Port Scan is a cyber-attack whose purpose is to find vulnerable ports in the committed computer. With the purpose of discovering exploitable services running on opened ports, the attacker scans the largest possible number of ports. The scanning strategies are varied because of the elevated number of ports, 65536. Most common services are between port 0 and 1023.

Once the scan is completed, the attacker obtains a list of analyzed ports and their state (opened or closed). An opened port has a service running on it that can be vulnerable. Port Scan attacks are usually directed to a computer network with the purpose of analyzing an elevated number of computers.

5.3.1 Results

The models developed in this project have obtained the performance described below detecting Port Scan attacks.

- **k-Nearest Neighbors**: K-NN algorithm algorithm reaches its best performance with 3 neighbours and 4 features.
- **Logistic Regression**: The best configuration for Logistic Regression algorithm is using 15 variables.
- **Gaussian Naive Bayes**: Gaussian Naive Bayes algorithm obtains its best scores with 12 features.
- **Multilayer Perceptron**: The best configuration for Multilayer Perceptron algorithm is using 42 variables with Rectified Linear unit activation function.

The next graphic representation compares the different algorithms performance using its best configuration. We display the f1-score obtained and the number of features needed to reach that score.

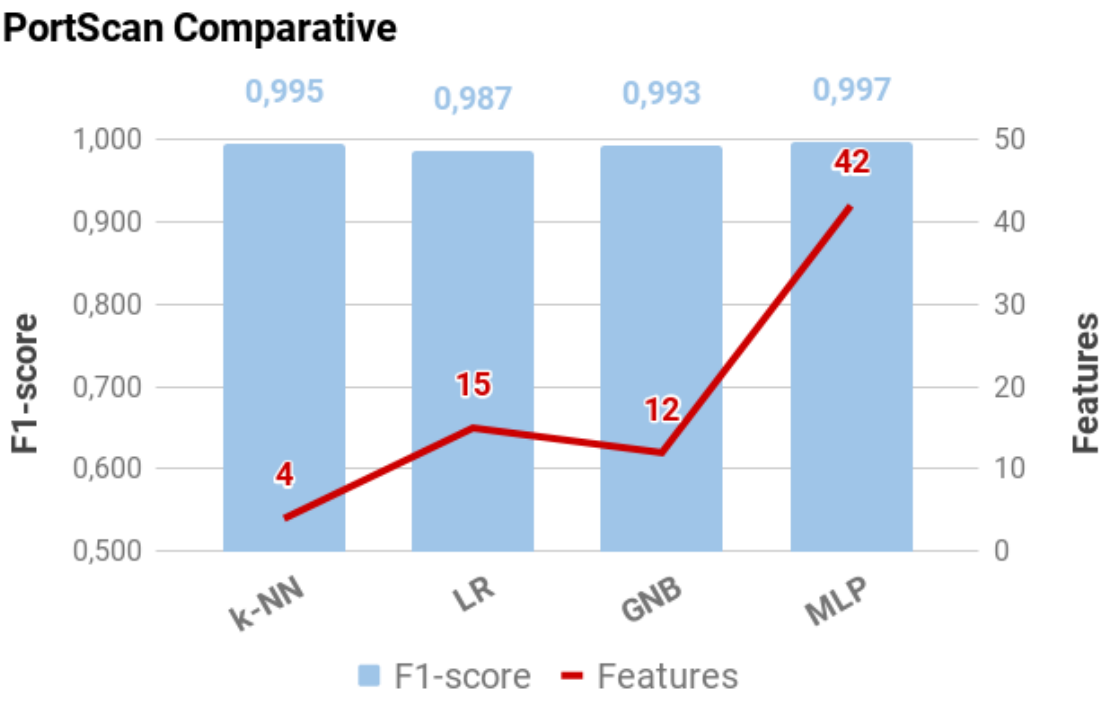

Figure 5.2: PortScan Algorithm Performance Comparative

Next, we show a comparative table with the best rating obtained for each algorithm. The displayed scores are Accuracy, Precision, Recall and f1-score. In addition, the number of features to reach those scores is also shown.

Algorithm	Features	Accuracy	Precision	Recall	f1-score
k-NN	4	0.995 (+/- 0.003)	0.995 (+/- 0.003)	0.995 (+/- 0.003)	0.995 (+/- 0.003)
LR	15	0.987 (+/- 0.016)	0.987 (+/- 0.015)	0.987 (+/- 0.016)	0.987 (+/- 0.016)
GNB	12	0.993 (+/- 0.007)	0.993 (+/- 0.007)	0.993 (+/- 0.006)	0.993 (+/- 0.007)
MLP	42	0.997 (+/- 0.003)	0.997 (+/- 0.003)	0.997 (+/- 0.003)	0.997 (+/- 0.003)

Finally, we show the list of 10 most relevant Features in Port Scan flows detection:

- Fwd Packet Length Min
- Bwd Packet Length Min
- Bwd Packet Length Mean
- Min Packet Length
- Packet Length Mean
- PSH Flag Count
- ACK Flag Count
- Average Packet Size
- Avg Bwd Segment Size
- Min Seg Size Forward

5.3.2 Conclusions

The best algorithm detecting Port Scan flows is Multilayer Perceptron with Rectified Linear unit activation function, however, the number of features required to reach those scores is too elevated, 42. However, k-Nearest Neighbors obtains a similar rating (f1-score of 0.995 and 99,5% of Accuracy) with a much lower number of features, only 4. Therefore, k-NN using 3 neighbors is the best algorithm detecting Port Scan attacks.

In this case, the size of interchanged packet (Fwd and Bwd Packet Length Min, Bwd Packet Length Mean, Min Packet Length...) is the most useful information.

5.4 Infiltration

Infiltration attacks consist of sending malicious files via email to the target computer. This file contains an application that exploits a vulnerability of the system. If the victim executes the file, a backdoor is opened and the attacker obtains access to the committed computer and its network.

5.4.1 Results

In our Infiltration Dataset, we have almost 300.000 flows, however, only 36 are malicious flows. The limited number of infiltration flows makes difficult to detect them. The models developed have obtained the performance described below detecting Infiltration attacks.

- **k-Nearest Neighbors**: K-NN algorithm algorithm reaches its best performance with 3 neighbours and 5 features.
- **Logistic Regression**: The best configuration for Logistic Regression algorithm is using 36 variables.
- **Gaussian Naive Bayes**: Gaussian Naive Bayes algorithm obtains its best scores with 38 features.

- **Multilayer Perceptron**: The best configuration for Multilayer Perceptron algorithm is using 51 variables with Logistic activation function.

The next graphic representation compares the different algorithms performance using its best configuration. We display the f1-score obtained and the number of features needed to reach that score:

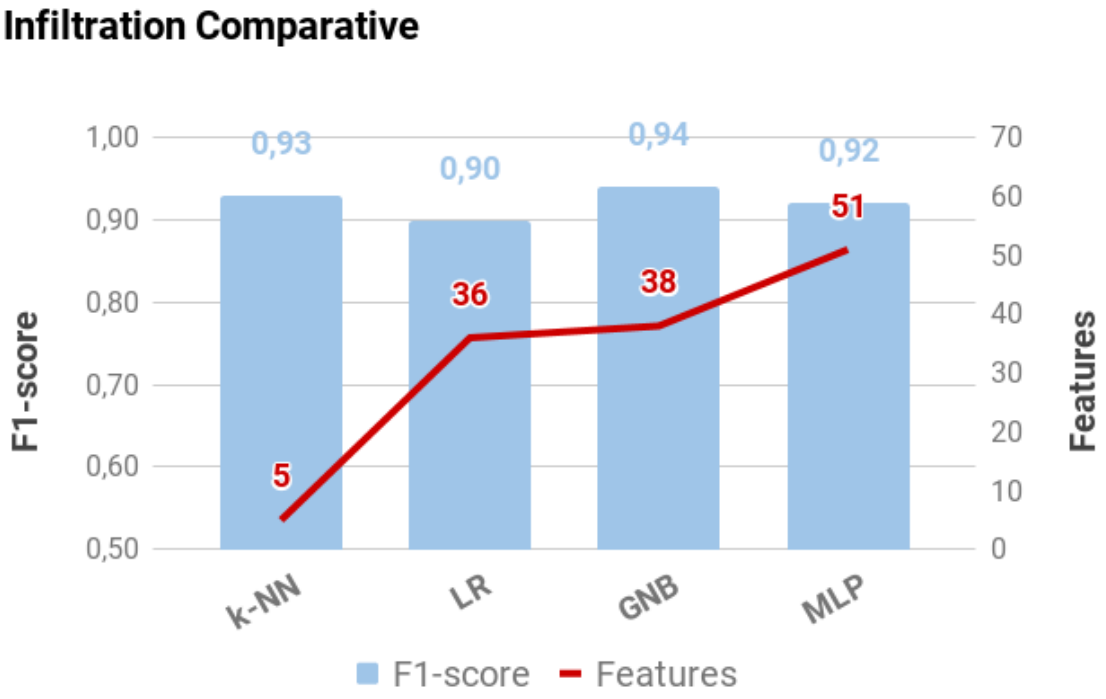

Figure 5.3: Infiltration Algorithm Performance Comparative

Down below, we show a comparative table with the best rating obtained for each algorithm. The displayed scores are Accuracy, Precision, Recall and f1-score. In addition, the number of features to reach those scores is also shown.

Algorithm	Features	Accuracy	Precision	Recall	f1-score
k-NN	5	0.93 (+/- 0.11)	0.94 (+/- 0.10)	0.93 (+/- 0.11)	0.93 (+/- 0.11)
LR	36	0.91 (+/- 0.13)	0.92 (+/- 0.12)	0.91 (+/- 0.13)	0.90 (+/- 0.15)
GNB	38	0.94 (+/- 0.09)	0.95 (+/- 0.06)	0.94 (+/- 0.09)	0.94 (+/- 0.09)
MLP	51	0.92 (+/- 0.10)	0.94 (+/- 0.07)	0.92 (+/- 0.10)	0.92 (+/- 0.10)

Finally we show the list of 10 most relevant Features in Infiltration attacks detection:

- Flow Duration
- Fwd Packet Length Max
- Fwd Packet Length Min
- Fwd Packet Length Std
- Flow IAT Max
- Fwd IAT Total
- Bwd IAT Total
- Min Packet Length
- PSH Flag Count
- ACK Flag Count

5.4.2 Conclusions

The best algorithms detecting Infiltration attacks with a similar f1-score are Gaussian Naive Bayes and k-Nearest Neighbors. However, the k-NN algorithm only requires 5 features while GNB needs 38 to reach that score. Therefore, the k-NN algorithm using 5 features and 3 neighbors obtains the best performance with an f1-score of 0,93 and an Accuracy of 93%. In spite of the number of false alarms is elevated, about 7% (1 - Recall) with regard to the flows classified as malicious it is still a suitable performance since the potential danger of this attack in a few flows. It is interesting to underline that Flow duration is the most useful feature in this attack type.

5.5 Botnet

The attack performed in the CICIDS2017 is Botnet ARES, an open source trojan. This attack takes control of every infected computer also named bot. Once a computer is infected, the trojan has the ability to run command shells, download and upload files, take screenshots and keylogging.

5.5.1 Results

The models developed in this project have obtained the performance described below detecting Botnet flows.

- **k-Nearest Neighbors**: K-NN algorithm algorithm reaches its best performance with 3 neighbours and 46 features.
- **Logistic Regression**: The best configuration for Logistic Regression algorithm is using 18 variables.
- **Gaussian Naive Bayes**: Gaussian Naive Bayes algorithm obtains its best scores with 10 features.

- **Multilayer Perceptron**: The best configuration for Multilayer Perceptron algorithm is using 63 variables with Logistic activation function.

The next graphic representation compares the different algorithms performance using its best configuration. We display the f1-score obtained and the number of features needed to reach that score:

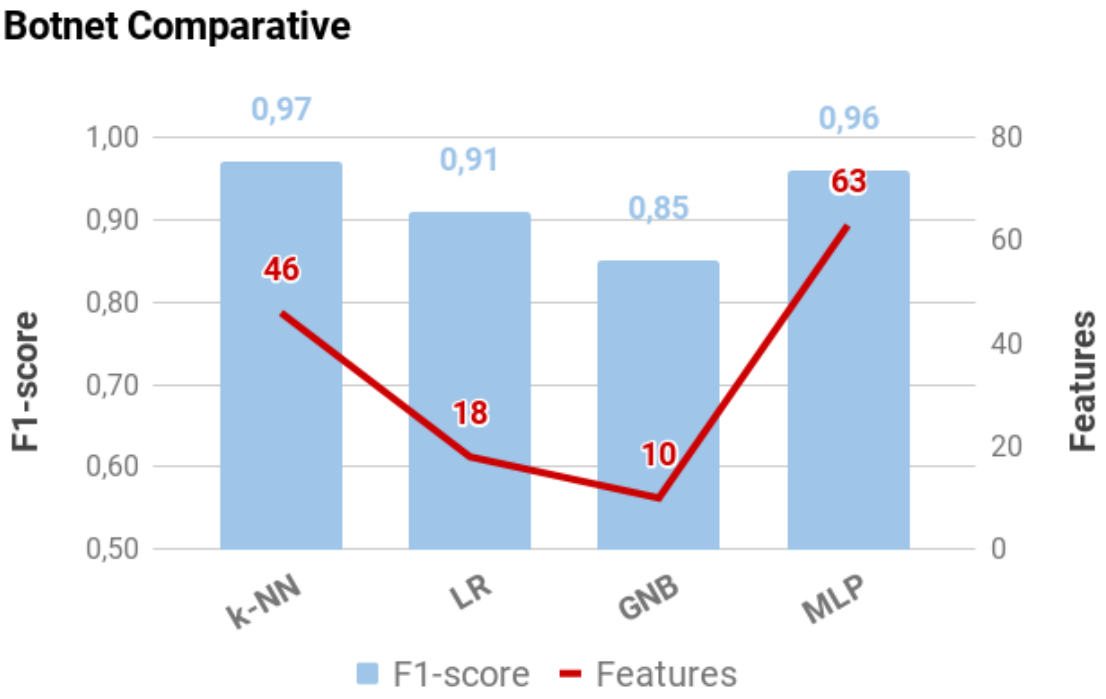

Figure 5.4: Botnet Algorithm Performance Comparative

Down below, we show a comparative table with the best rating obtained for each algorithm. The displayed scores are Accuracy, Precision, Recall and f1-score. In addition, the number of features to reach those scores is also shown.

Algorithm	Features	Accuracy	Precision	Recall	f1-score
k-NN	46	0.97 (+/- 0.02)	0.97 (+/- 0.02)	0.97 (+/- 0.02)	0.97 (+/- 0.02)
LR	18	0.91 (+/- 0.03)	0.92 (+/- 0.02)	0.91 (+/- 0.03)	0.91 (+/- 0.04)
GNB	10	0.86 (+/- 0.04)	0.89 (+/- 0.02)	0.86 (+/- 0.04)	0.85 (+/- 0.04)
MLP	63	0.96 (+/- 0.02)	0.96 (+/- 0.02)	0.96 (+/- 0.02)	0.96 (+/- 0.02)

Finally we show the list of 10 most relevant Features in Botnet attacks detection:

- Destination Port
- Flow Duration
- Fwd Packet Length Min
- Bwd Packet Length Max
- Bwd Packet Length Min
- Bwd Packet Length Mean
- Min Packet Length
- PSH Flag Count
- URG Flag Count
- Avg Bwd Segment Size

5.5.2 Conclusions

The best algorithm detecting Botnet attacks is k-Nearest Neighbors using 46 features and 3 neighbors obtaining an f1-score of 0,97 and an Accuracy of 97%. Nevertheless, Logistic Regression algorithm obtains an f1-score of 0.91 with only 18 features. If the computing resources are enough and the criticality of the protected infrastructure is elevated, the k-NN is the best algorithm. Otherwise, Logistic Regression is a good model. In this attack type, Destination Port and Flow Duration are the most useful variables.

5.6 Web Attacks

The Web Attacks Dataset includes Cross-site scripting, SQL Injection and Web Brute Force attacks. These cyber-attacks are performed through web applications. Cross-site scripting or XSS consists of injecting code in JavaScript or a similar language into a web page. An SQL injection is similar to XSS but the injection is directed to a database. If a web site is vulnerable to SQL injection, the attacker can perform SQL queries such as drop tables or modify rows. Finally, Web Brute Force attack consists of accessing or login in restricted web sites by brute force. It means trying combinations of user and passwords.

5.6.1 Results

The models developed in this project have obtained the performance described below detecting Web Attacks.

- **k-Nearest Neighbors**: K-NN algorithm algorithm reaches its best performance with 3 neighbours and 61 features.

- **Logistic Regression**: The best configuration for Logistic Regression algorithm is using 1 variables.

- **Gaussian Naive Bayes**: Gaussian Naive Bayes algorithm obtains its best scores with 53 features.
- **Multilayer Perceptron**: The best configuration for Multilayer Perceptron algorithm is using 11 variables with Logistic activation function.

The next graphic representation compares the different algorithms performance using its best configuration. We display the f1-score obtained and the number of features needed to reach that score:

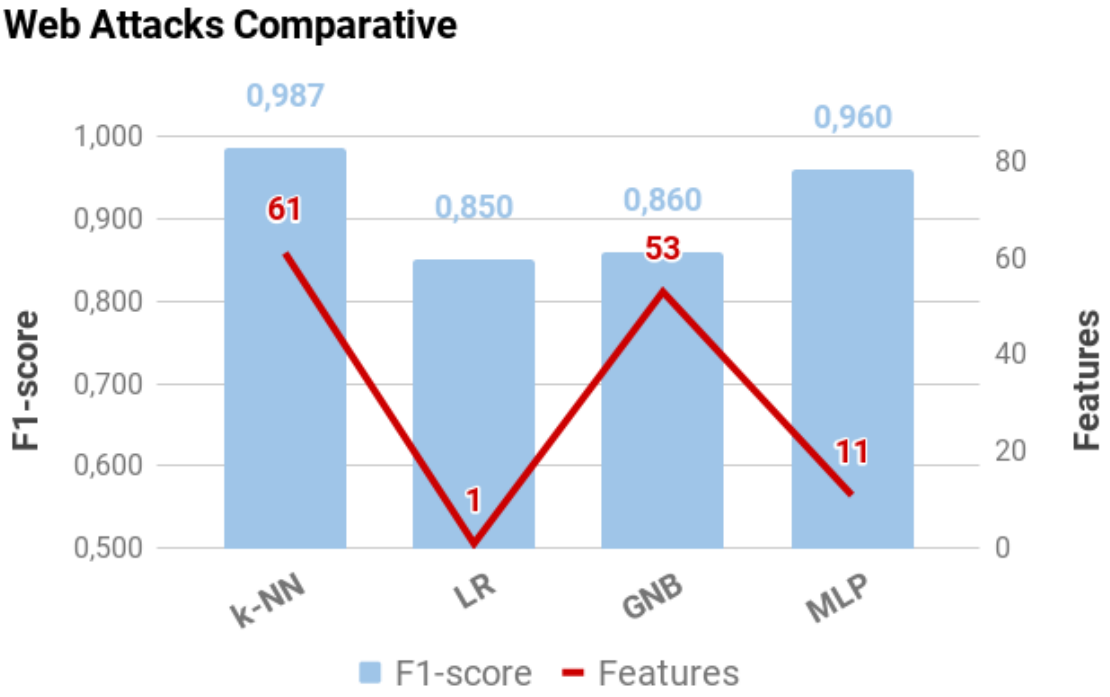

Figure 5.5: Web Attacks Algorithm Performance Comparative

Down below, we show a comparative table with the best rating obtained for each algorithm. The displayed scores are Accuracy, Precision, Recall and f1-score. In addition, the number of features to reach those scores is also shown.

Algorithm	Features	Accuracy	Precision	Recall	f1-score
k-NN	61	0.987 (+/- 0.023)	0.987 (+/- 0.022)	0.987 (+/- 0.023)	0.987 (+/- 0.023)
LR	1	0.86 (+/- 0.03)	0.89 (+/- 0.02)	0.86 (+/- 0.03)	0.85 (+/- 0.03)
GNB	53	0.86 (+/- 0.04)	0.89 (+/- 0.03)	0.86 (+/- 0.04)	0.86 (+/- 0.05)
MLP	11	0.96 (+/- 0.03)	0.96 (+/- 0.03)	0.96 (+/- 0.03)	0.96 (+/- 0.03)

Finally we show the list of 10 most relevant Features in Web Attacks detection:

- Bwd IAT Total
- Bwd IAT Mean
- Bwd IAT Std
- Bwd IAT Max
- Init Win Bytes Forward
- Init Win Bytes Backward
- Min Seg Size Forward
- Idle Mean
- Idle Max
- Idle Min

5.6.2 Conclusions

Similarly to Botnet Dataset, there are two algorithms with a suited performance. k-Nearest Neighbors obtains an f1-score of 0,987 using 61 features while Multilayer Perceptron obtains an f1-score of 0,96 using only 11 variables. In this situation, the decision depends on the nature of the network to be protected. If the infrastructure is critical, we would choose k-NN with 3 neighbors in spite of the elevated computing cost. Otherwise, we would choose the MLP with Logistic activation function.

The most important variable in Web Attacks is Destination Port since all malicious flows are directed to that port. However, the model is trained to analyze only flows directed to port 80. Besides this variable, the most important features detecting Web Attacks are the time interval between two packets in the backward direction (Bwd IAT Total, Bwd IAT Mean, Bwd IAT Std...)

5.7 Brute Force

Brute Force attacks consist of repetitive attempts of user and passwords combination with the purpose to access a restricted service. Due to the number of possible combinations is very high, hacking tools use user and password dictionaries. In the CICIDS2017 the Brute Force attacks are performed Patator Python script against FTP and SSH services.

5.7.1 Results

The models developed in this project have obtained the performance described below detecting Brute Force Attacks.

- **k-Nearest Neighbors**: K-NN algorithm algorithm reaches its best performance with 3 neighbours and 34 features.

- **Logistic Regression**: The best configuration for Logistic Regression algorithm is using 39 variables.

- **Gaussian Naive Bayes**: Gaussian Naive Bayes algorithm obtains its best scores with 3 features.

- **Multilayer Perceptron**: The best configuration for Multilayer Perceptron algorithm is using 14 variables with Logistic activation function.

The next graphic representation compares the different algorithms performance using its best configuration. We display the f1-score obtained and the number of features needed to reach that score:

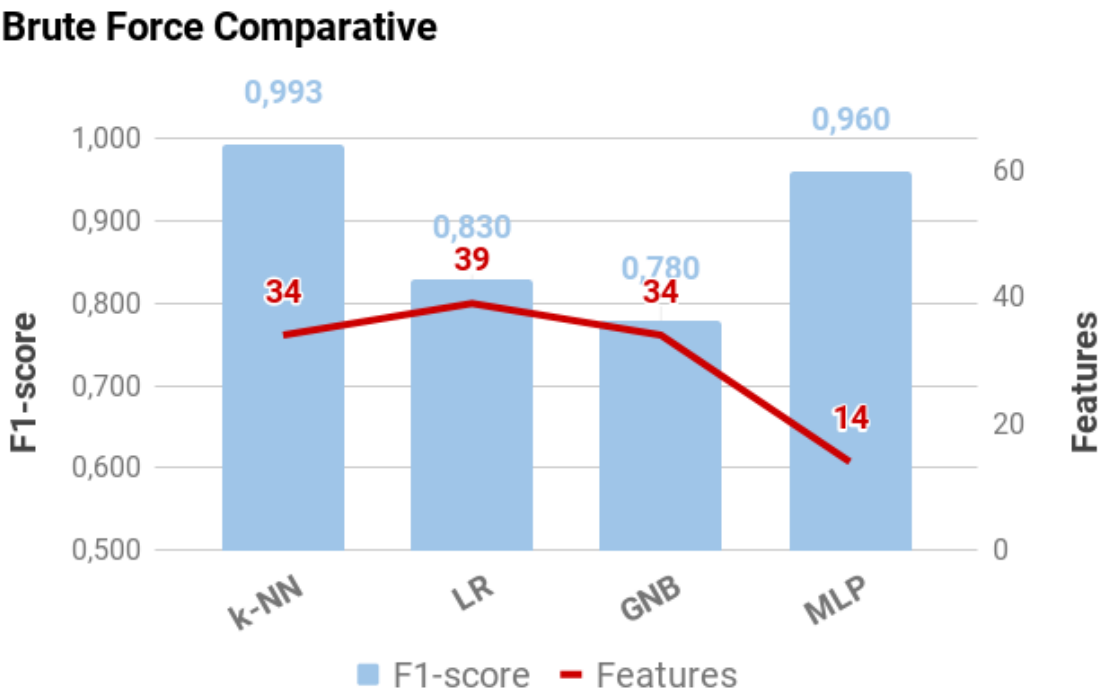

Figure 5.6: Brute Force Algorithm Performance Comparative

Down below, we show a comparative table with the best rating obtained for each algorithm. The displayed scores are Accuracy, Precision, Recall and f1-score. In addition, the number of features to reach those scores is also shown.

Algorithm	Features	Accuracy	Precision	Recall	f1-score
k-NN	34	0.993 (+/- 0.011)	0.993 (+/- 0.011)	0.993 (+/- 0.011)	0.993 (+/- 0.011)
LR	39	0.83 (+/- 0.09)	0.87 (+/- 0.06)	0.83 (+/- 0.09)	0.83 (+/- 0.09)
GNB	34	0.79 (+/- 0.04)	0.85 (+/- 0.02)	0.79 (+/- 0.04)	0.78 (+/- 0.04)
MLP	14	0.96 (+/- 0.05)	0.96 (+/- 0.05)	0.96 (+/- 0.05)	0.96 (+/- 0.05)

Finally, we show the list of 10 most relevant Features in Brute Force attacks detection:

- Destination Port
- Flow Duration
- Fwd IAT Total
- Fwd IAT Std
- Fwd IAT Max
- Bwd IAT Total
- Bwd IAT Std
- Fwd PSH Flags
- SYN Flag Count
- Min Seg Size Forward

5.7.2 Conclusions

In Brute Force detection we have two algorithms with a suitable performance: k-Nearest Neighbors with 34 features and 3 neighbors obtains an f1-score of 0,993 and Multilayer Perceptron using 14 variables with Logistic activation function obtains and f1-score of 0,96. The k-NN model is more precise but it requires more information while the MLP algorithm needs less information but its performance slightly decrease.

The most important information detecting Brute Force attacks is Destination Port due to all attacks are directed to 21 and 22 ports. In addition, the flow duration and the time interval between packets in both directions is the most useful information in detecting these attacks.

CHAPTER 6

Conclusions and future work

In this last chapter, we draw the conclusions and achievements obtained in this project. Furthermore, we explain the problems faced during the development of the work. Finally, we show some possible lines of future work and applications of this project.

In section 6.1, we explain the conclusions obtained in this project, the problems faced during the development of the work and the limitations of the results. Finally, in section 6.2, we outline some future work lines to continue the project.

6.1 Conclusions

In Chapter 5 we discussed briefly the conclusions obtained from the evaluation and comparison of the algorithms used in this project for each attack type. As of this comparison, we can conclude that k-Nearest Neighbors is the best algorithm for attack detection problems. This algorithm is the best option in four different attacks out of six. However, the rest of the algorithms (GNB, LR, and MLP) obtain a suitable performance and sometimes even more useful because they require less information than k-NN.

In this project, we have faced the problem of a binary classification. A binary classifier must beat the accuracy of a random classifier (50%). The best models developed in this project for each attack type have obtained an average accuracy of 98%. This is a high performance, however, due to the elevated number of flows in computer network communications, false alarm ratio becomes a problem to solve. For this purpose, we can modify the threshold of positive malicious flows. This way, the number of false alarms decrease, though, the amount of malicious flows undetected raise.

The features or information contained in the Datasets can be obtained just capturing the protected network traffic. Therefore, the models developed are applicable to attacks coming from new machines because they do not need any particular attacker information such as IP directions. As we can observe in Chapter 5, the most useful variables detecting malicious flows are:

Destination Port, Packet Length (minimum, maximum, average and standard deviation) in the forward and backward direction, Time Interval between packets and the number of times that PSH, ACK and URG flags appear.

This is the expected result because these features characterize the cyber-attacks analyzed.

We divide into two types the problems faced in this project. Firstly, we describe the limitations of the results. Next, we review the problems dealt with developing this project.

We must outline the next limitations of the developed models:

– Models require the complete flow information since some training data are based on the entire communication such as the maximum, minimum and average packet length or the flag count. In the case of IDS applications, this is not a problem because the system alerts once the flow is completed. However, IPS applications act in real time, consequently generating flow information is a problem to solve. Obtaining flows statistics in real time, it means while the communication is still active, would be the

solution. However, this requires a more elevated computing capacity.

- The tested models have achieved high scores, nevertheless, these are not strict result because models are tested only with the Dataset. More accurate a clarifying results would be accomplished with a simulation of an entire network and an IDS implementing the models developed.
- Finally, as we explained in Chapter 1, the fast-changing and evolution of cyber-attacks become efficient models into deprecated in short periods of time. That is why training and generating new models must be a continuous process with the purpose of being up-to-date. This implies an additional waste of money.

To conclude, the most notable problem faced during the development of this project is the computing capacity required due to the elevated volume of information stored in datasets. However, the investigation group together I develop this project (Grupo de Sistemas Inteligentes[1]) provides me the computing resources needed to train the models.

6.2 Future Work

There are a few lines to keep working on this project. In the first place, we could continue developing models for IPS. The traffic stored in the Datasets is gathered by the CICFlowMeter tool developed by the CIC. This tool labels each flows once the communication finishes. However, IPSs require traffic information in real time. Therefore, we need to create an application able to capture packets and generate the statistical traffic values needed for models in real time. Furthermore, this would also require new models adapted to the new information flow. This would be the most interesting line to keep working due to the high interest not only detecting but preventing intrusions.

On the other hand, another appealing future work would be testing the developed models in real situations with the purpose of checking their real behaviour. It requires creating a platform able to capture and generate the information stored in datasets. We could use the CICFlowMeter, the open source traffic flow processor developed by CIC. Once the packet capturer and processor is ready, we just need to send the information to the classifier which determines if the flow is benign or malicious.

[1]Grupo de Sistemas Inteligentes: `https://www.gsi.upm.es/`

Bibliography

[1] Activation function - Wikipedia. https://en.wikipedia.org/wiki/Activation_function. (Accessed on 06/09/2019).

[2] Applied Deep Learning: Artificial Neural Networks. https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6. (Accessed on 06/05/2019).

[3] IDS 2017 - Datasets - Research - Canadian Institute for Cybersecurity - UNB. https://www.unb.ca/cic/datasets/ids-2017.html. (Accessed on 06/04/2019).

[4] K-fold Cross-validation - Wikipedia. https://en.wikipedia.org/wiki/Cross-validation_(statistics). (Accessed on 06/09/2019).

[5] Market Trends in Cybersecurity. https://www.incibe.es/sites/default/files/estudios/cybersecurity_market_trends.pdf, July 2016. (Accessed on 06/04/2019).

[6] Eugène Ezin and Hervé Djihountry. Java-based intrusion detection system in a wired network. *International Journal of Computer Science and Information Security*, 9:33–40, 11 2011.

[7] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. An evaluation framework for intrusion detection dataset. In *2016 International Conference on Information Science and Security (ICISS)*, pages 1–6, Dec 2016.

[8] Herjavec Group. 2019 Official Annual Cybercrime Report. https://www.herjavecgroup.com/wp-content/uploads/2018/12/CV-HG-2019-Official-Annual-Cybercrime-Report.pdf. (Accessed on 05/31/2019).

[9] Ashish Kumar, Srikant Chandak, and Rita Dewanjee. Recent advances in intrusion detection systems: An analytical evaluation and comparative study. *International Journal of Computer Applications*, 86(4), 2014.

[10] NSS LABS. Next Generation Intrusion Prevention System (NGIPS). https://www.fortinet.com/content/dam/fortinet/assets/certifications/nss-labs-2018-dcips-svm.pdf, September 2018. (Accessed on 06/04/2019).

[11] J. D. Rodriguez, A. Perez, and J. A. Lozano. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):569–575, March 2010.

[12] J. Schonwalder, M. Bjorklund, and P. Shafer. Network configuration management using netconf and yang. *IEEE Communications Magazine*, 48(9):166–173, Sept 2010.

[13] Cisco Systems. What is cybersecurity? `https://www.cisco.com/c/en/us/products/security/what-is-cybersecurity.html`. (Accessed on 05/31/2019).

[14] International Telecommunication Union. Statistics. `https://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx`. (Accessed on 05/31/2019).

[15] Ed. YB. Claise. Cisco Systems NetFlow Services Export Version 9. RFC 3954, RFC Editor, October 2004.

APPENDIX A

Impact of the project

In this appendix, we analyze the professional responsibility of engineering project development. Furthermore, we describe the project impacts at the social, economic and environmental level. Finally, we discuss the ethical implications and the responsibility of the practical application of engineering.

A.1 Introduction

The Internet has changed the way we live and relate. Today, we carry out a big number of task and activities through connected devices. All these services accessed through Internet require protection. That is why cybersecurity is a growing and developing field due to the powerful tool that the Internet has become.

This project is situated in the cybersecurity area in the field of software development. As we described before, the objective of this work is to improve network security by proposing a set of solutions based on Machine Learning. The models developed are designed to be implemented in intrusion detection and prevention systems, IDSs and IPSs.

As with all engineering projects, this work has a series of impacts and responsibilities. This project aims to contribute to society with new scientific knowledge. The objective of this work is to make a safer and more protected world through investigation. Next, we describe the project impact at different levels.

A.2 Social Impact

It is expected that this project improves network security. That would increase the user trust in the Internet. With better confidence and network protection, the use of the Internet for all kind of activities in special those that require privacy and security will increase as well as new applications and features. Both IPSs and IDSs are designed to protect private networks, consequently, they are mainly used at the enterprise level. However, since a large proportion of user information is stored in the cloud, protecting the privacy of users implies protecting the services they use. That is why better corporate network protection guarantees better user privacy and safety reducing cybercrime.

A.3 Economic Impact

At the economic level, cybersecurity and cybercrime are two sectors that move large amounts of money. That is why any advance in this area has important economic implications. The consequence of improving IDSs and IPSs is the increase in their use and marketing. The applications of new technologies cybersecurity like Machine Learning can improve protection tools and even develop new products. These products are highly demanded by companies due to their vital importance in protecting networks.

A.4 Environmental Impact

Since this project is a software development based, its direct environmental impact is practically zero. Nevertheless, like any other human activity, this work has an environmental footprint because its application implies the deployment of infrastructures and technological devices. These equipment require pollutant materials (batteries, heavy metals, plastics...) and their manufacture also contaminates. We may also mention the use of electricity in electronic devices. In spite of these negative impacts, there are no other adverse effects on the environment.

A.5 Ethical and Professional Implications

There are a series of ethical implications due to the improvement of cybersecurity. Like any other technological advance, it can be used for better or worse. Unfortunately, the use of knowledge and technology does not depend on the people who developed it. On the one hand, the improvement of network security guarantees greater safety for those who use that technology. However, those users that cannot afford the cost of these devices will become more vulnerable.

On the other hand, improving protection and privacy could increase and facilitate illegal activities through the Internet. Furthermore, due to cybercriminals can also access to protection tools, they can develop better and more complex attacks able to bypass actual cybersecurity systems.

APPENDIX B

Cost of the System

In this appendix, we show the economic costs of this project. The costs are generated by two main factors: physical resources and human resources. We also provide a table where we gather all the expenses.

B.1 Cost of the System

In this appendix, we analyze the economic cost of the project. We only consider the expenses of developing Machine Learning models.

There are two types of costs:

- **Labour cost:** The development of this project has required a total of 300 working hours. We estimate the average salary for a Software Engineer about Euros 15 per hour. The total expense in human resources is 4.500 €.
- **Cost of materials:** To develop this project we need a personal computer for working with text editors, write the code and evaluate models and a training computer to execute the code. The personal computer does not need any special features however the training computer requires appropriate hardware. We estimate the cost of the personal computer about 1.000 € and 2.500 € for the training computer. Finally, we must attach the waste of energy of these two computers. We estimate in about 720 hours number of computer working hours. The power consumed by a computer during 720 hours is about 720kWh. We obtain the energy cost multiplying the energy consumed by the price of kWh (average price of 0.18 € /kWh) for a total of about 130 €.

The total budget is: 7.345,45 €

Next, we gather all cost in a descriptive table:

		Hours	Price/hour	TOTAL
LABOR COST (direct cost)		300	15 €	**4.500 €**
COST OF MATERIAL RESOURCES (direct cost)	**Purchase pirce**	**Use in months**	**Depretiation in years**	**TOTAL**
Personal computer (Software included)	1.000,00 €	6	5	100,00 €
Trainign computer (Software included)	2.500,00 €	6	5	250,00 €
Energy waste	130,00 €	-	-	130,00 €
COST OF MATERIAL RESOURCES				**480,00 €**
GENERAL EXPENSES (indirect cost)	15%	due to the direct cost		**747,00 €**
INDUSTRIAL BENEFIT	6%	due to the direct cost + indirect cost		**343,62 €**
SUBTOTAL BUDGET				**6.070,62 €**
VAT APPLICABLE			21%	**1.274,83 €**
TOTAL BUDGET				**7.345,45 €**

APPENDIX C

Dataset Features Description

In this appendix, we describe every feature of the Dataset. We display in a table the feature name and its description. The Dataset is labelled with 78 features and the tag determining if a flow is benign or malicious.

C.1 Dataset Features

This dataset collects a total of 78 features and the tag determining if the flow is benign or an attack. Next, we describe the list of features:

Feature name	Description
Destination Port	Destination Port
Flow Duration	Flow duration
Total Fwd Packets	Total packets in the forward direction
Total Backward Packets	Total packets in the backward direction
Total Length of Fwd Packets	Total size of packet in forward direction
Total Length of Bwd Packets	Total size of packet in backward direction
Fwd Packet Length Max	Maximum size of packet in forward direction
Fwd Packet Length Min	Minimum size of packet in forward direction
Fwd Packet Length Mean	Mean size of packet in forward direction
Fwd Packet Length Std	Standard deviation size of packet in forward direction
Bwd Packet Length Max	Maximum size of packet in backward direction
Bwd Packet Length Min	Minimum size of packet in backward direction
Bwd Packet Length Mean	Mean size of packet in backward direction
Bwd Packet Length Std	Standard deviation size of packet in backward direction
Flow Bytes/s	Flow byte rate that is number of bytes transferred per second
Flow Packets/s	Flow packets rate that is number of packets transferred per second
Flow IAT Mean	Mean time between two flows
Flow IAT Std	Standard deviation time two flows
Flow IAT Max	Maximum time between two flows
Flow IAT Min	Minimum time between two flows
Fwd IAT Total	Total time between two packets sent in the forward direction
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction

Feature name	Description
Fwd IAT Max	Maximum time between two packets sent in the forward direction
Fwd IAT Min	Minimum time between two packets sent in the forward direction
Bwd IAT Total	Total time between two packets sent in the backward direction
Bwd IAT Mean	Mean time between two packets sent in the backward direction
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
Bwd IAT Max	Maximum time between two packets sent in the backward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction
Fwd PSH Flags	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
Bwd PSH Flags	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
Fwd URG Flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
Bwd URG Flags	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
Fwd Header Length	Total bytes used for headers in the forward direction
Bwd Header Length	Total bytes used for headers in the backward direction
Fwd Packets/s	Number of forward packets per second
Bwd Packets/s	Number of backward packets per second
Min Packet Length	Minimum length of a flow
Max Packet Length	Maximum length of a flow
Packet Length Mean	Mean length of a flow
Packet Length Std	Standard deviation length of a flow
Packet Length Variance	Minimum inter-arrival time of packet
FIN Flag Count	Number of packets with FIN
SYN Flag Count	Number of packets with SYN

Feature name	Description
RST Flag Count	Number of packets with RST
PSH Flag Count	Number of packets with PUSH
ACK Flag Count	Number of packets with ACK
URG Flag Count	Number of packets with URG
CWE Flag Count	Number of packets with CWE
ECE Flag Count	Number of packets with ECE
Down/Up Ratio	Download and upload ratio
Average Packet Size	Average size of packet
Avg Fwd Segment Size	Average size observed in the forward direction
Avg Bwd Segment Size	Average size observed in the backward direction
Fwd Header Length.1	Total bytes used for headers in the forward direction
Fwd Avg Bytes/Bulk	Average number of bytes bulk rate in the forward direction
Fwd Avg Packets/Bulk	Average number of packets bulk rate in the forward direction
Fwd Avg Bulk Rate	Average number of bulk rate in the forward direction
Bwd Avg Bytes/Bulk	Average number of bytes bulk rate in the backward direction
Bwd Avg Packets/Bulk	Average number of packets bulk rate in the backward direction
Bwd Avg Bulk Rate	Average number of bulk rate in the backward direction
Subflow Fwd Packets	The average number of packets in a sub flow in the forward direction
Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction
Subflow Bwd Packets	The average number of packets in a sub flow in the backward direction
Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
Init Win Bytes Forward	Number of bytes sent in initial window in the forward direction
Init Win Bytes Backward	Number of bytes sent in initial window in the backward direction
Act Data Pkt Fwd	Number of packets with at least 1 byte of TCP data payload in the forward directio

Feature name	Description
Min Seg Size Forward	Minimum segment size observed in the forward direction
Active Mean	Mean time a flow was active before becoming idle
Active Std	Standard deviation time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle
Active Min	Minimum time a flow was active before becoming idle
Idle Mean	Mean time a flow was idle before becoming active
Idle Std	Standard deviation time a flow was idle before becoming active
Idle Max	Maximum time a flow was idle before becoming active
Idle Min	Minimum time a flow was idle before becoming active

APPENDIX D

Algorithm Performance Visualization

In this appendix, we provide a deeper performance analysis of the models developed. For each attack type, we analyze the four algorithms used and their configuration. For each algorithm, we display the interval of features where the scores reach the maximum value. Finally, we show the scores achieved by the best algorithm configuration and the number of features used.

D.1 DDoS

D.1.1 k-Nearest Neighbors

The k-NN algorithm obtains its best scores in the range of 7 to 15 variables with 3 neighbours:

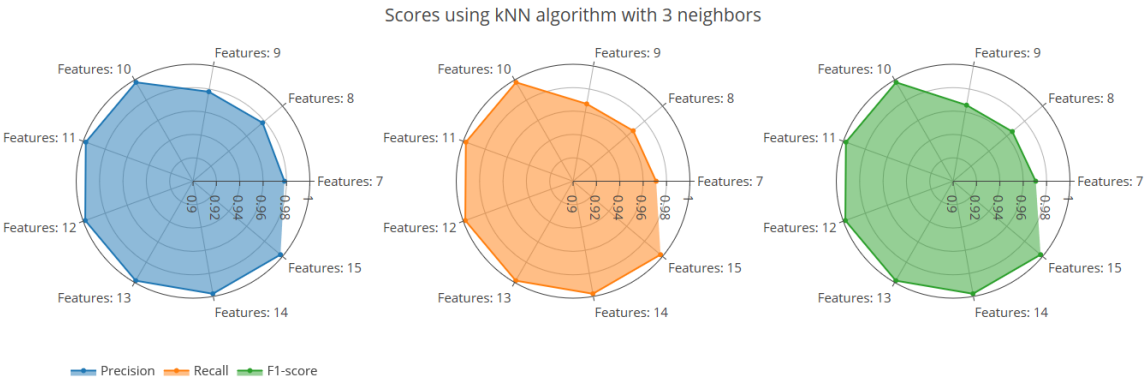

Figure D.1: k-NN scores

Note that the radial axis is delimited between 0.9 and 1 in the previous graphic representation.

The k-Nearest Neighbors algorithm reaches its best performance with 3 neighbours and 12 features obtaining the next scores:

- Accuracy: 0.998 (+/- 0.002)
- Precision: 0.998 (+/- 0.002)
- Recall: 0.998 (+/- 0.002)
- f1-score: 0.998 (+/- 0.002)

D.1.2 Logistic Regression

Logistic Regression algorithm obtains its best scores in the range of 3 to 11 variables:

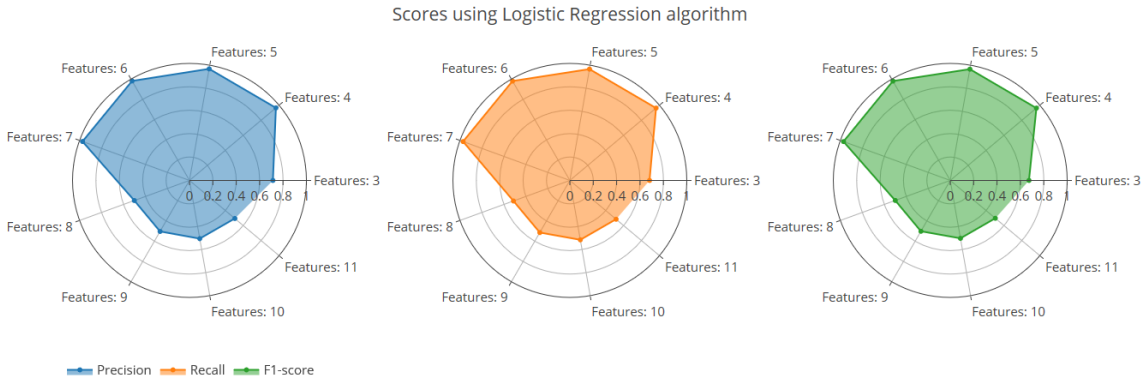

Figure D.2: Logistic Regression scores

The best configuration for Logistic Regression algorithm is using 6 variables. . With this configuration, we get the next performance:

- Accuracy: 0.96 (+/- 0.04)
- Precision: 0.96 (+/- 0.04)
- Recall: 0.96 (+/- 0.04)
- f1-score: 0.96 (+/- 0.04)

D.1.3 Gaussian Naive Bayes

Logistic Regression algorithm obtains its best scores in the range of 3 to 11 variables:

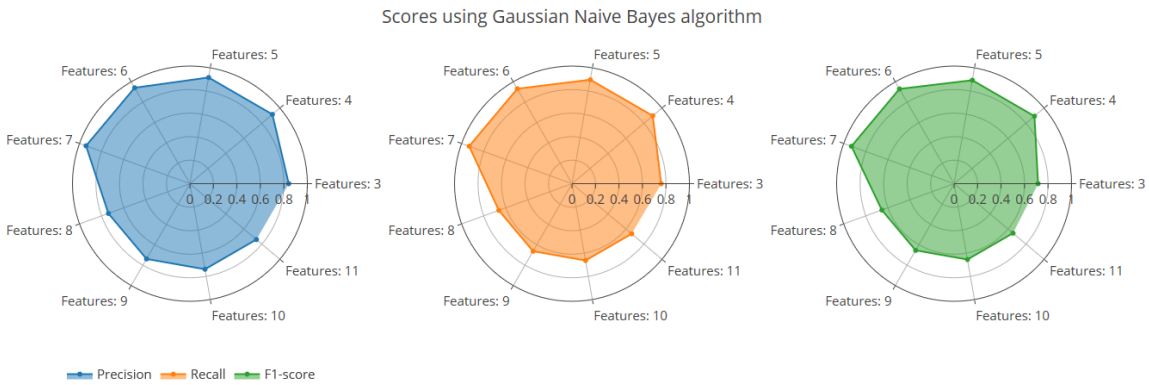

Figure D.3: Gaussian Naive scores

Gaussian Naive Bayes algorithm reaches its best performance with 6 features obtaining the next scores:

- Accuracy: 0.93 (+/- 0.09)
- Precision: 0.94 (+/- 0.07)

- Recall: 0.93 (+/- 0.09)
- f1-score: 0.93 (+/- 0.09)

D.1.4 Multilayer Perceptron

The Multilayer Perceptron obtains its best scores in the range of 3 to 11 variables with the Logistic activation function:

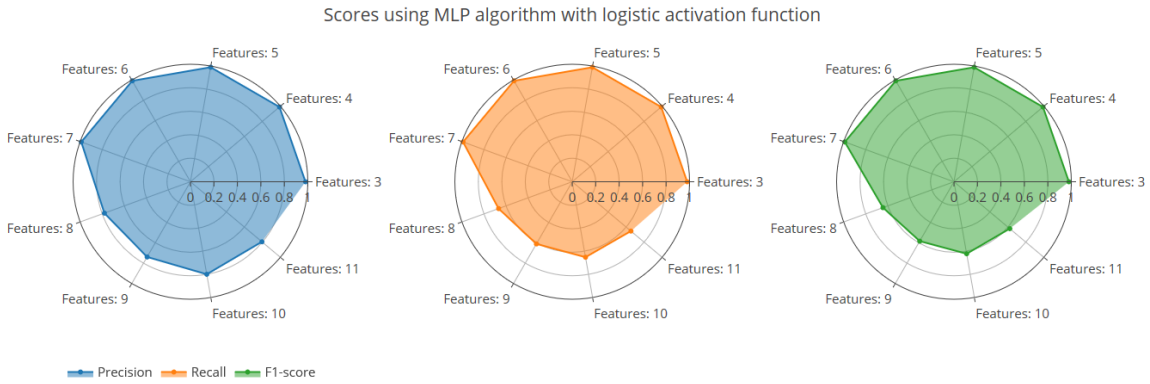

Figure D.4: Multilayer Perceptron scores

The best configuration for Multilayer Perceptron is using 6 variables with Logistic activation function. With this configuration we get the next scores:

- Accuracy: 0.97 (+/- 0.15)
- Precision: 0.97 (+/- 0.10)
- Recall: 0.97 (+/- 0.15)
- f1-score: 0.96 (+/- 0.17)

D.2 Port Scan

D.2.1 k-Nearest Neighbors

The k-NN algorithm obtains its best scores in the range of 1 to 5 variables with 3 neighbours:

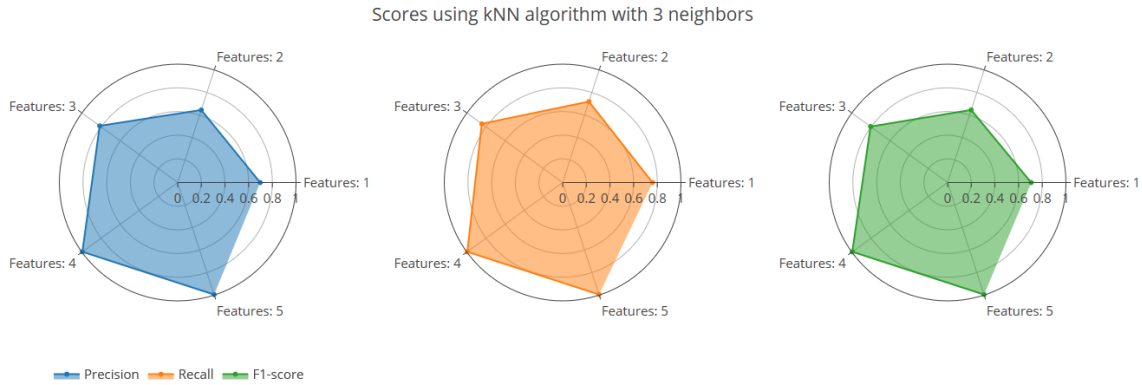

Figure D.5: k-NN scores

The k-Nearest Neighbors algorithm reaches its best performance with 3 neighbours and 4 features obtaining the next scores:

- Accuracy: 0.995 (+/- 0.003)
- Precision: 0.995 (+/- 0.003)
- Recall: 0.995 (+/- 0.003)
- f1-score: 0.995 (+/- 0.003)

D.2.2 Logistic Regression

Logistic Regression algorithm obtains its best scores in the range of 10 to 18 variables:

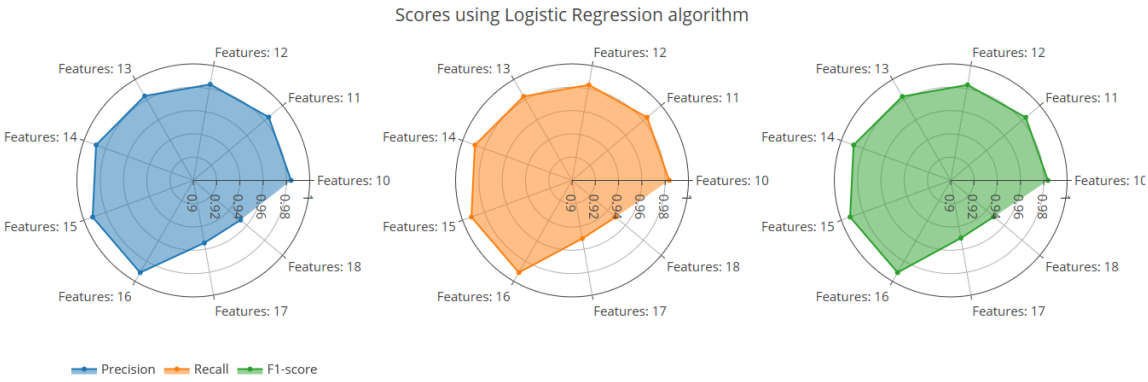

Figure D.6: Logistic Regression scores

Note that the radial axis is delimited between 0.9 and 1 in the previous graphic representation.

The best configuration for Logistic Regression algorithm is using 15 variables. With this configuration, we get the next performance:

- Accuracy: 0.987 (+/- 0.016)
- Precision: 0.987 (+/- 0.015)
- Recall: 0.987 (+/- 0.016)
- f1-score: 0.987 (+/- 0.016)

D.2.3 Gaussian Naive Bayes

Logistic Regression algorithm obtains its best scores in the range of 8 to 16 variables:

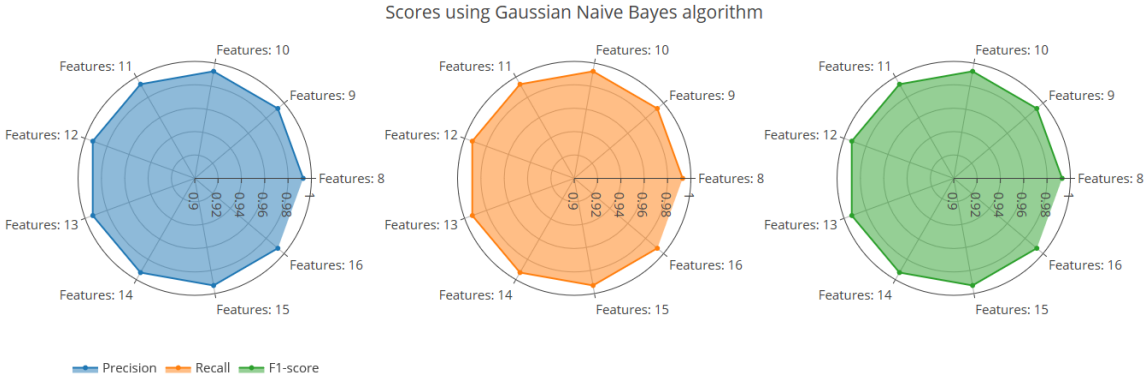

Figure D.7: Gaussian Naive scores

Note that the radial axis is delimited between 0.9 and 1 in the previous graphic representation. Gaussian Naive Bayes algorithm reaches its best performance with 12 features obtaining the next scores:

- Accuracy: 0.993 (+/- 0.007)
- Precision: 0.993 (+/- 0.006)
- Recall: 0.993 (+/- 0.007)
- f1-score: 0.993 (+/- 0.007)

D.2.4 Multilayer Perceptron

The Multilayer Perceptron obtains its best scores in the range of 37 to 45 variables with the Rectified Linear unit function:

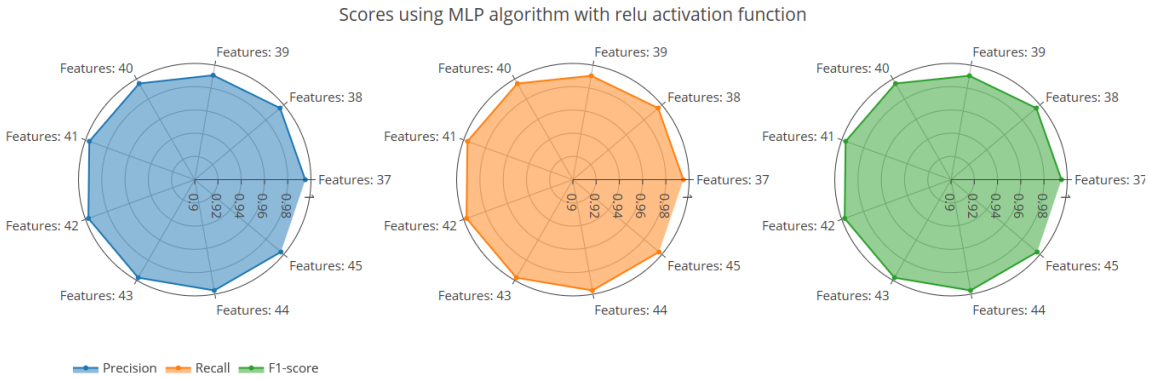

Figure D.8: Multilayer Perceptron scores

Note that the radial axis is delimited between 0.9 and 1 in the previous graphic representation. The best configuration for Multilayer Perceptron is using 42 variables with Rectified Linear unit activation function. With this configuration we get the next scores:

- Accuracy: 0.997 (+/- 0.003)
- Precision: 0.997 (+/- 0.003)
- Recall: 0.997 (+/- 0.003)
- f1-score: 0.997 (+/- 0.003)

D.3 Infiltration

D.3.1 k-Nearest Neighbors

The k-NN algorithm obtains its best scores in the range of 1 to 9 variables with 5 neighbours:

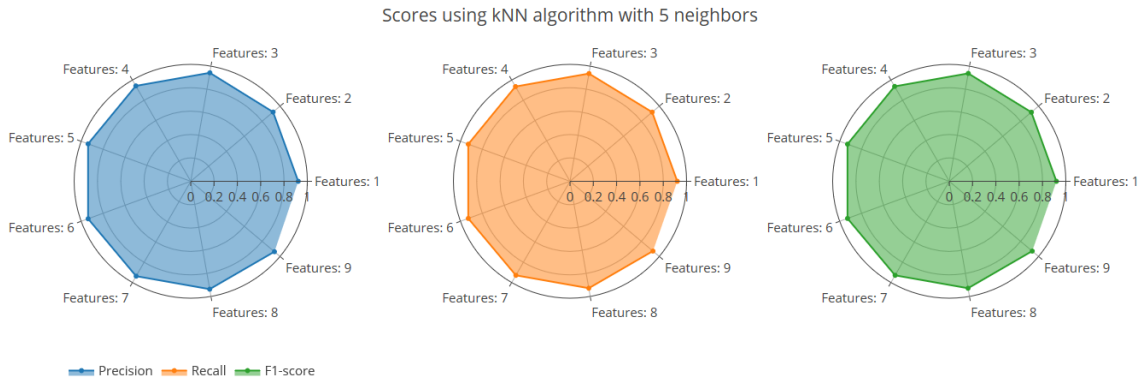

Figure D.9: k-NN scores

The k-Nearest Neighbors algorithm reaches its best performance with 5 neighbours and 3 features obtaining the next scores:

- Accuracy: 0.93 (+/- 0.11)
- Precision: 0.94 (+/- 0.10)
- Recall: 0.93 (+/- 0.11)
- f1-score: 0.93 (+/- 0.11)

D.3.2 Logistic Regression

Logistic Regression algorithm obtains its best scores in the range of 32 to 40 variables:

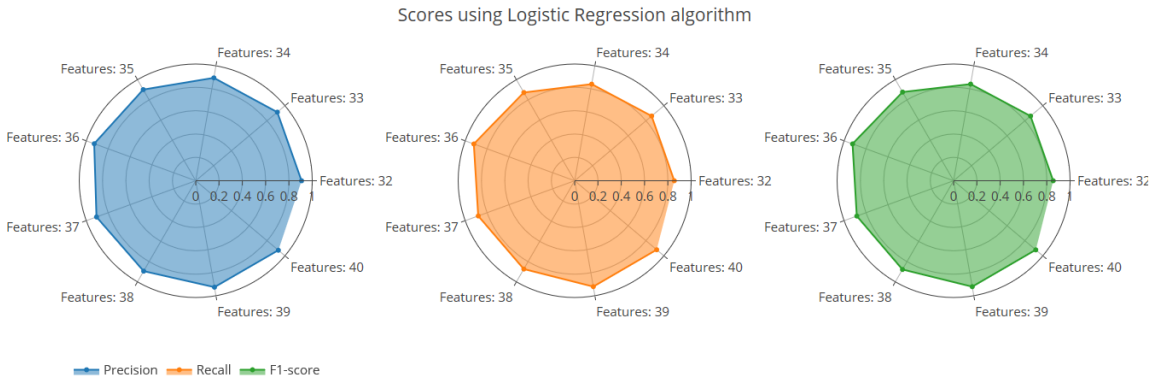

Figure D.10: Logistic Regression scores

The best configuration for Logistic Regression algorithm is using 36 variables. With this configuration, we get the next performance:

- Accuracy: 0.91 (+/- 0.13)
- Precision: 0.92 (+/- 0.12)
- Recall: 0.91 (+/- 0.13)
- f1-score: 0.90 (+/- 0.15)

D.3.3 Gaussian Naive Bayes

Logistic Regression algorithm obtains its best scores in the range of 35 to 43 variables:

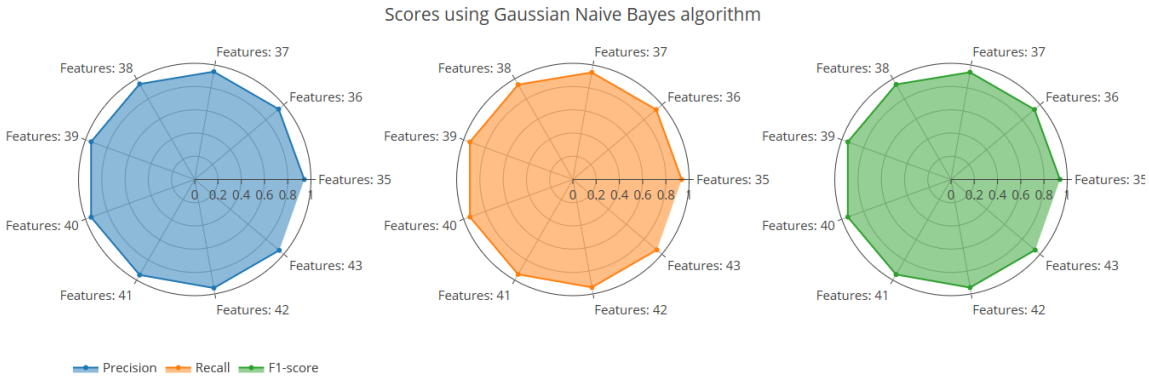

Figure D.11: Gaussian Naive scores

Gaussian Naive Bayes algorithm reaches its best performance with 38 features obtaining the next scores:

- Accuracy: 0.94 (+/- 0.09)
- Precision: 0.95 (+/- 0.06)
- Recall: 0.94 (+/- 0.09)
- f1-score: 0.94 (+/- 0.09)

D.3.4 Multilayer Perceptron

The Multilayer Perceptron obtains its best scores in the range of 47 to 55 variables with the Logistic activation function:

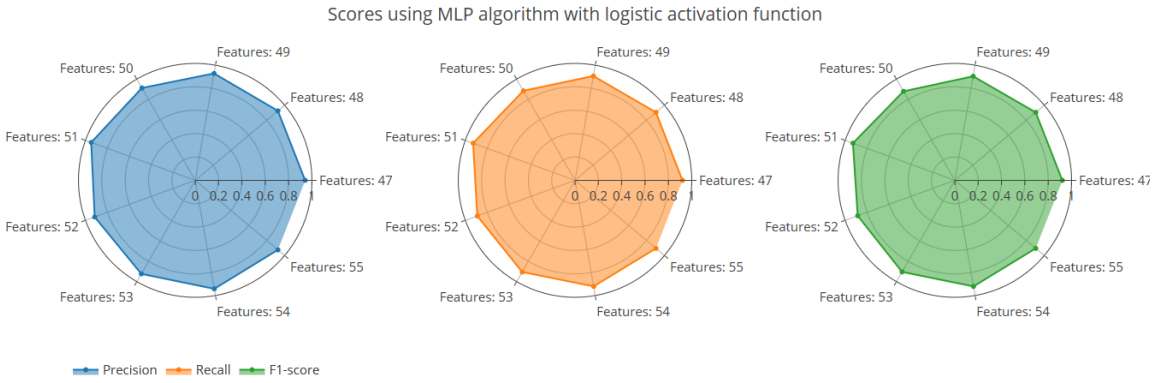

Figure D.12: Multilayer Perceptron scores

The best configuration for Multilayer Perceptron is using 51 variables with Logistic activation function. With this configuration we get the next scores:

- Accuracy: 0.92 (+/- 0.10)

- Precision: 0.94 (+/- 0.07)
- Recall: 0.92 (+/- 0.10)
- f1-score: 0.92 (+/- 0.10)

D.4 Botnet

D.4.1 k-Nearest Neighbors

The k-NN algorithm obtains its best scores in the range of 42 to 50 variables with 3 neighbours:

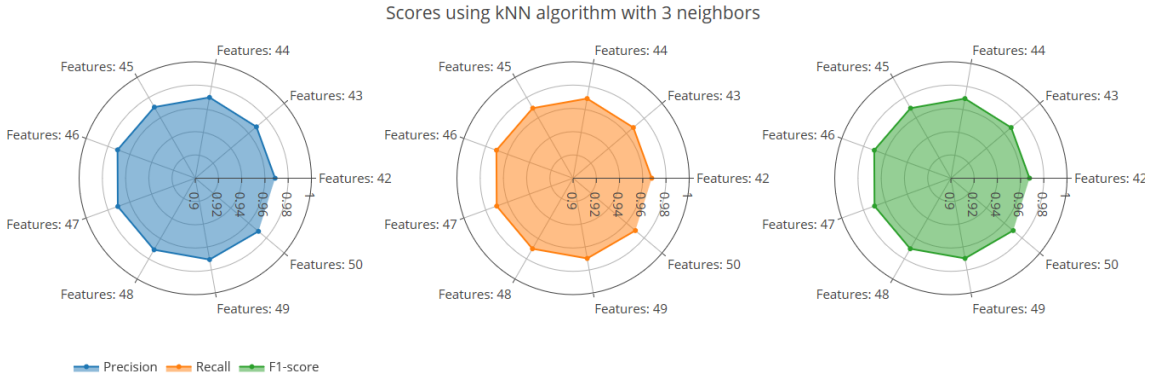

Figure D.13: k-NN scores

Notice that the radial axis is delimited between 0.9 and 1 in the previous graphic representation.

The k-Nearest Neighbors algorithm reaches its best performance with 3 neighbours and 46 features obtaining the next scores:

- Accuracy: 0.97 (+/- 0.02)
- Precision: 0.97 (+/- 0.02)
- Recall: 0.97 (+/- 0.02)
- f1-score: 0.97 (+/- 0.02)

D.4.2 Logistic Regression

Logistic Regression algorithm obtains its best scores in the range of 16 to 24 variables:

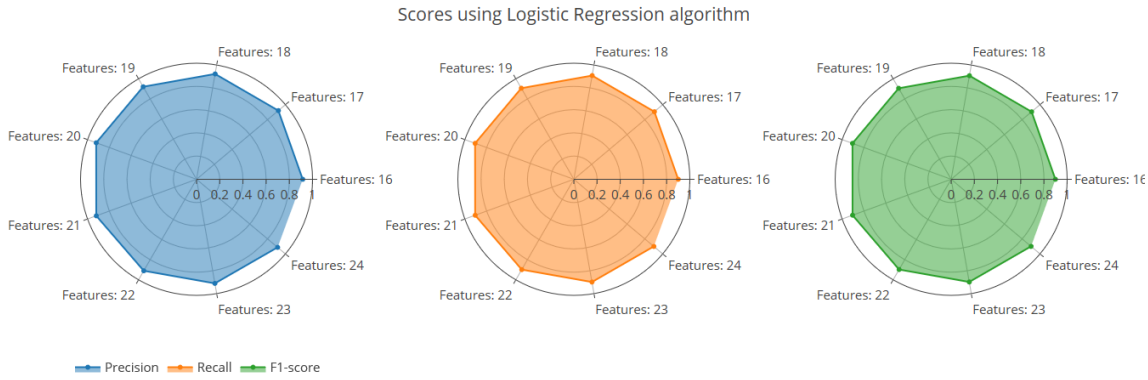

Figure D.14: Logistic Regression scores

The best configuration for Logistic Regression algorithm is using 18 variables. With this configuration, we get the next performance:

- Accuracy: 0.91 (+/- 0.03)
- Precision: 0.92 (+/- 0.02)
- Recall: 0.91 (+/- 0.03)
- f1-score: 0.91 (+/- 0.04)

D.4.3 Gaussian Naive Bayes

Logistic Regression algorithm obtains its best scores in the range of 6 to 14 variables:

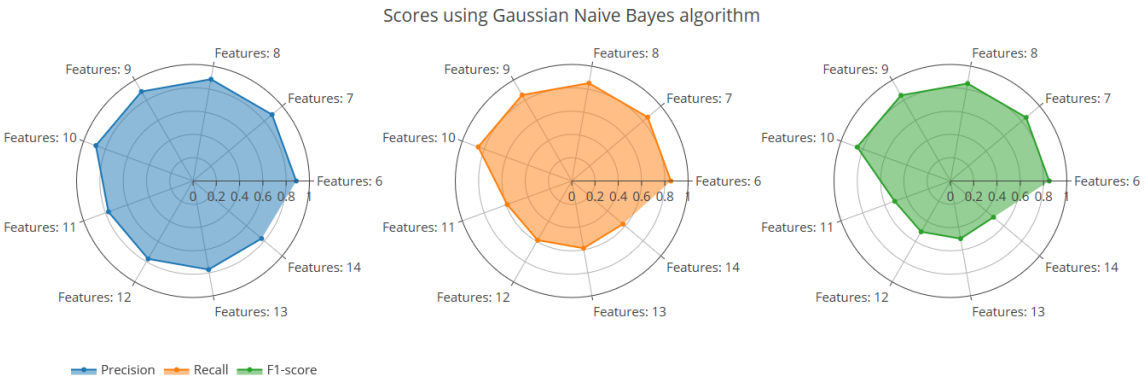

Figure D.15: Gaussian Naive scores

Gaussian Naive Bayes algorithm reaches its best performance with 10 features obtaining the next scores:

- Accuracy: 0.86 (+/- 0.04)

- Precision: 0.89 (+/- 0.02)
- Recall: 0.86 (+/- 0.04)
- f1-score: 0.85 (+/- 0.04)

D.4.4 Multilayer Perceptron

The Multilayer Perceptron obtains its best scores in the range of 59 to 67 variables with the Rectified Linear unit activation function:

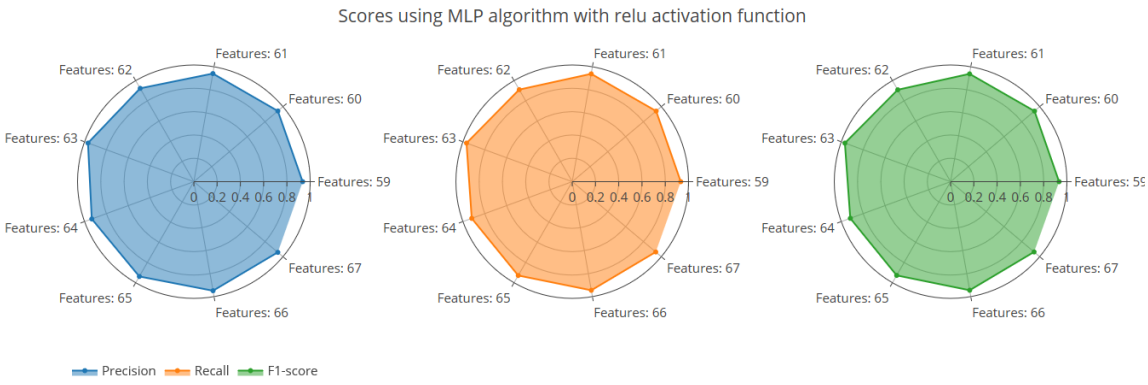

Figure D.16: Multilayer Perceptron scores

The best configuration for Multilayer Perceptron is using 63 variables with Rectified Linear unit activation function. With this configuration we get the next scores:

- Accuracy: 0.96 (+/- 0.02)
- Precision: 0.96 (+/- 0.02)
- Recall: 0.96 (+/- 0.02)
- f1-score: 0.96 (+/- 0.02)

D.5 Web Attacks

D.5.1 k-Nearest Neighbors

The k-NN algorithm obtains its best scores in the range of 58 to 66 variables with 3 neighbours:

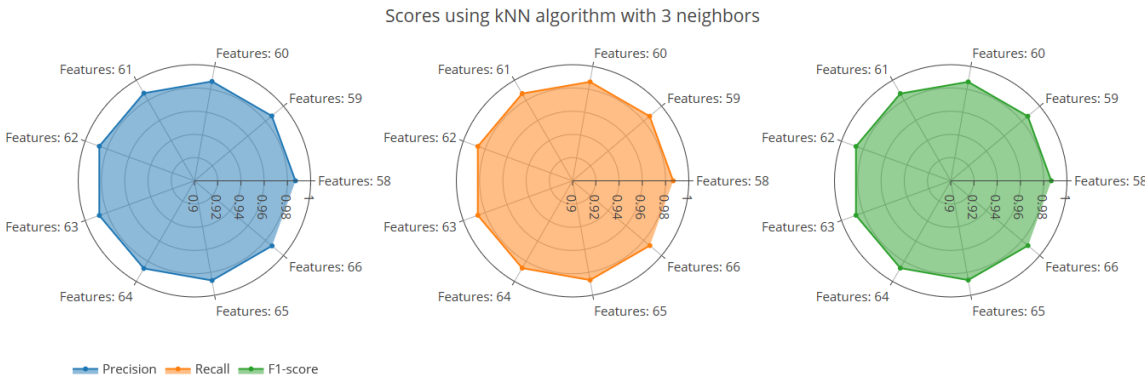

Figure D.17: k-NN scores

Notice that the radial axis is delimited between 0.9 and 1 in the previous graphic representation.

The k-Nearest Neighbors algorithm reaches its best performance with 3 neighbours and 61 features obtaining the next scores:

- Accuracy: 0.987 (+/- 0.023)
- Precision: 0.987 (+/- 0.022)
- Recall: 0.987 (+/- 0.023)
- f1-score: 0.987 (+/- 0.023)

D.5.2 Logistic Regression

Logistic Regression algorithm obtains its best scores in the range of 1 to 9 variables:

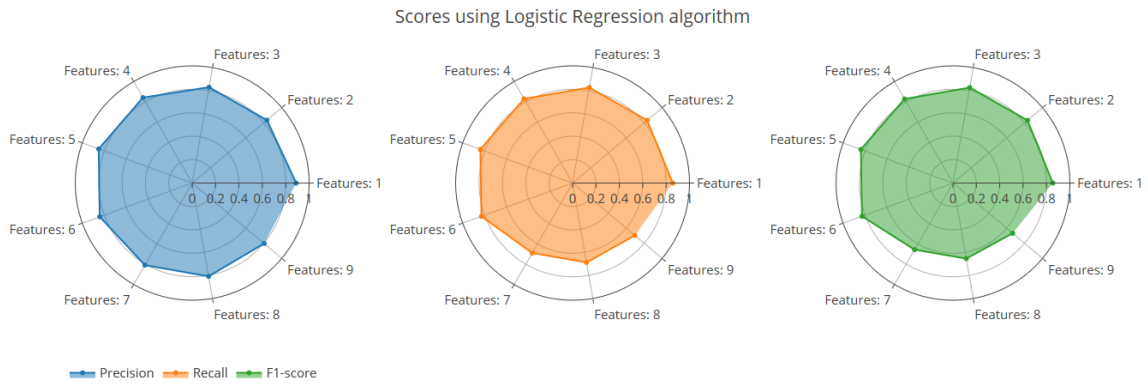

Figure D.18: Logistic Regression scores

The best configuration for Logistic Regression algorithm is using 1 variable. With this

configuration, we get the next performance:

- Accuracy: 0.86 (+/- 0.03)
- Precision: 0.89 (+/- 0.02)
- Recall: 0.86 (+/- 0.03)
- f1-score: 0.85 (+/- 0.03)

D.5.3 Gaussian Naive Bayes

Logistic Regression algorithm obtains its best scores in the range of 49 to 57 variables:

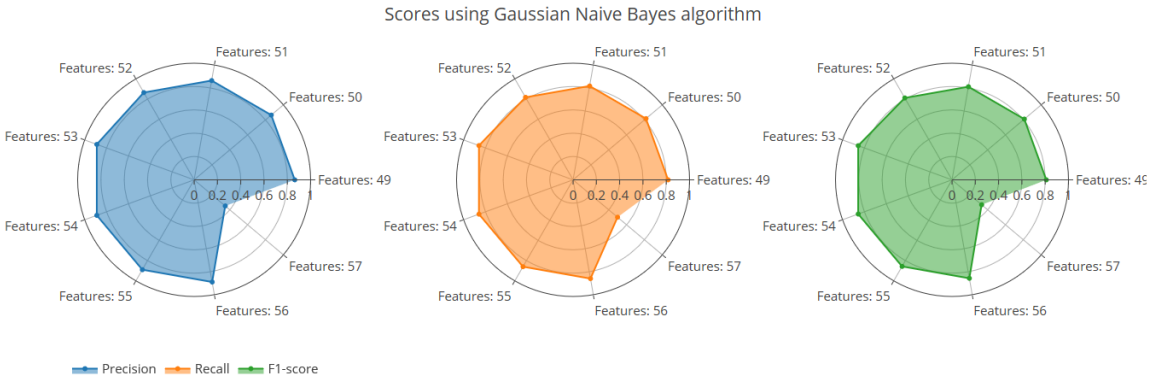

Figure D.19: Gaussian Naive scores

Gaussian Naive Bayes algorithm reaches its best performance with 53 features obtaining the next scores:

- Accuracy: 0.86 (+/- 0.04)
- Precision: 0.89 (+/- 0.03)
- Recall: 0.86 (+/- 0.04)
- f1-score: 0.86 (+/- 0.05)

D.5.4 Multilayer Perceptron

The Multilayer Perceptron obtains its best scores in the range of 7 to 15 variables with the Rectified Linear unit activation function:

xxx

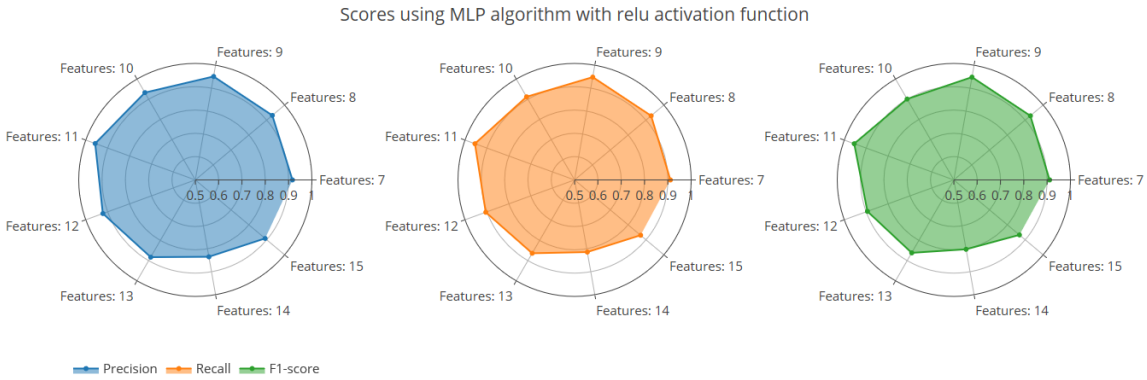

Figure D.20: Multilayer Perceptron scores

The best configuration for Multilayer Perceptron is using 11 variables with Rectified Linear unit activation function. With this configuration we get the next scores:

- Accuracy: 0.96 (+/- 0.03)
- Precision: 0.96 (+/- 0.03)
- Recall: 0.96 (+/- 0.03)
- f1-score: 0.96 (+/- 0.03)

D.6 Brute Force

D.6.1 k-Nearest Neighbors

The k-NN algorithm obtains its best scores in the range of 31 to 39 variables with 3 neighbours:

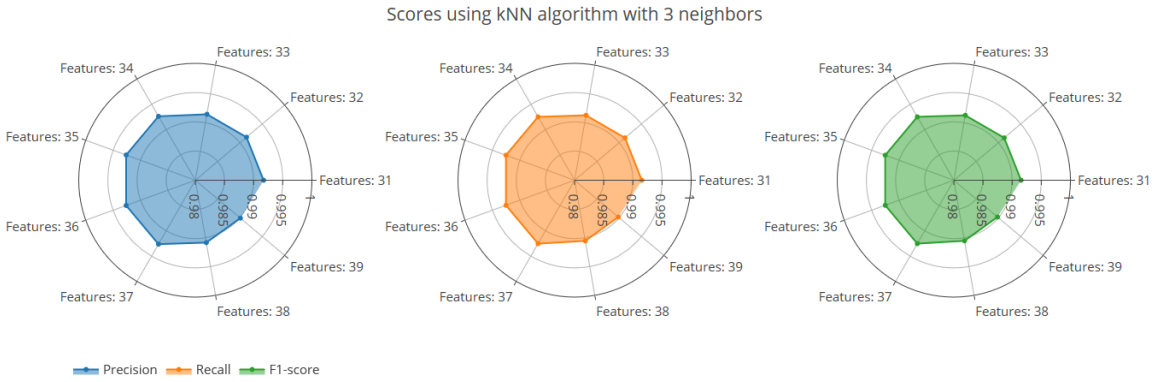

Figure D.21: k-NN scores

Notice that the radial axis is delimited between 0.98 and 1 in the previous graphic

representation.

The k-Nearest Neighbors algorithm reaches its best performance with 3 neighbours and 34 features obtaining the next scores:

- Accuracy: 0.993 (+/- 0.011)
- Precision: 0.993 (+/- 0.011)
- Recall: 0.993 (+/- 0.011)
- f1-score: 0.993 (+/- 0.011)

D.6.2 Logistic Regression

Logistic Regression algorithm obtains its best scores in the range of 36 to 44 variables:

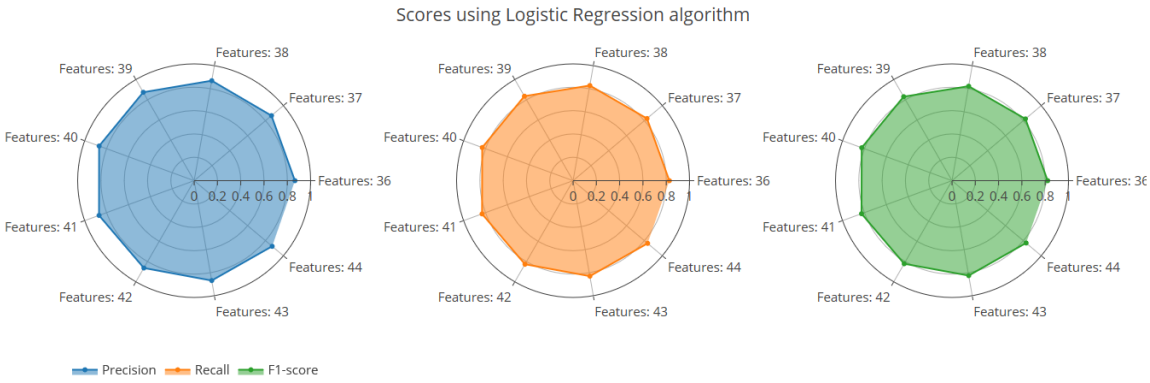

Figure D.22: Logistic Regression scores

The best configuration for Logistic Regression algorithm is using 39 variable. With this configuration, we get the next performance:

- Accuracy: 0.83 (+/- 0.09)
- Precision: 0.87 (+/- 0.06)
- Recall: 0.83 (+/- 0.09)
- f1-score: 0.83 (+/- 0.09)

D.6.3 Gaussian Naive Bayes

Logistic Regression algorithm obtains its best scores in the range of 31 to 39 variables:

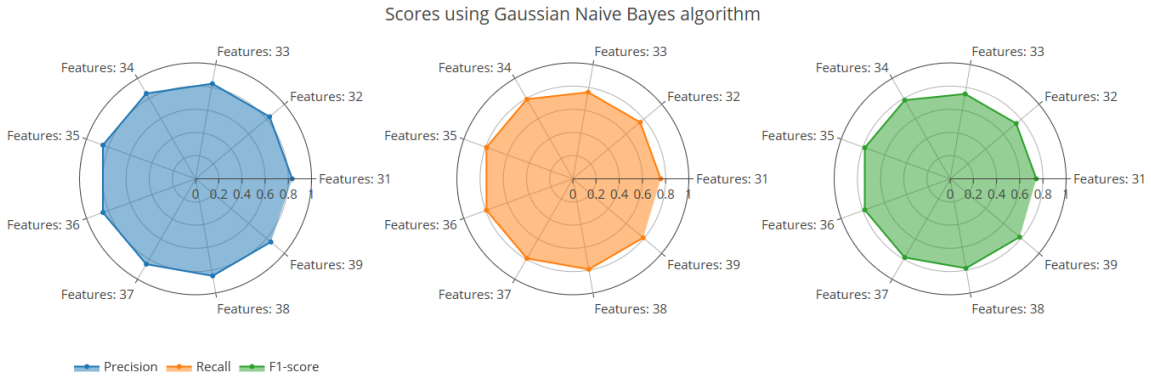

Figure D.23: Gaussian Naive scores

Gaussian Naive Bayes algorithm reaches its best performance with 34 features obtaining the next scores:

- Accuracy: 0.79 (+/- 0.04)
- Precision: 0.85 (+/- 0.02)
- Recall: 0.79 (+/- 0.04)
- f1-score: 0.78 (+/- 0.04)

D.6.4 Multilayer Perceptron

The Multilayer Perceptron obtains its best scores in the range of 11 to 19 variables with the Hyperbolic tangent activation function:

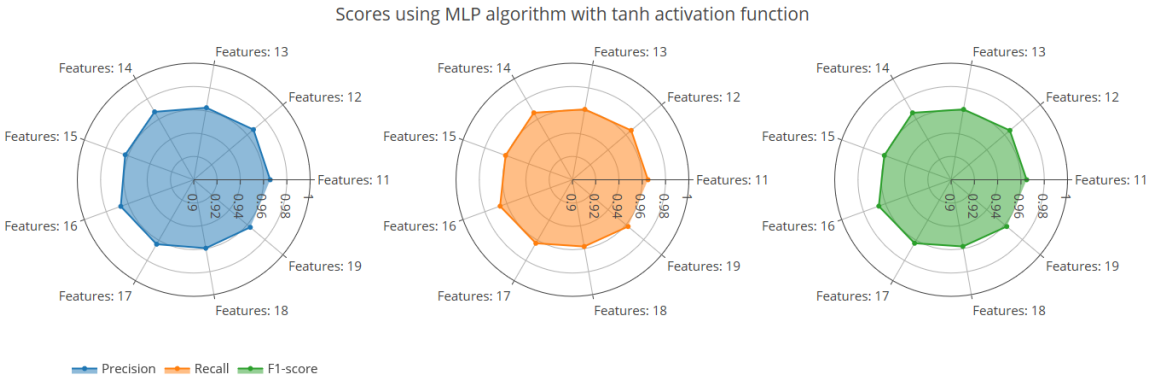

Figure D.24: Multilayer Perceptron scores

Notice that the radial axis is delimited between 0.9 and 1 in the previous graphic representation.

The best configuration for Multilayer Perceptron is using 14 variables with Hyperbolic

tangent activation function. With this configuration we get the next scores:

- Accuracy: 0.96 (+/- 0.05)
- Precision: 0.96 (+/- 0.05)
- Recall: 0.96 (+/- 0.05)
- f1-score: 0.96 (+/- 0.05)