UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA DE REDES Y SERVICIOS TELEMATICOS

TRABAJO FIN DE MÁSTER

APLICACIÓN DE TÉCNICAS DE APRENDIZAJE AUTOMÁTICO A FUENTES DE DATOS DE CIBERSEGURIDAD

ANDREA MEDINA PAREDES

2018

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

Máster Universitario en

Ingeniería de Redes y Servicios Telemáticos

TRABAJO FIN DE MÁSTER

APLICACIÓN DE TÉCNICAS DE APRENDIZAJE AUTOMÁTICO A FUENTES DE DATOS DE CIBERSEGURIDAD

Autor

Andrea Estefanía Medina Paredes

Director

Irene Romero Ibáñez

Departamento de Ingeniería de Sistemas Telemáticos

2018

Resumen

Resumen

La guerra contra las amenazas de intrusión se libra todos los días por usuarios y empresas que desean sentirse seguros frente a cualquier ataque cibernético que comprometa la integridad de los usuarios y equipos de la red. La expansión de soluciones tecnológicas también podría estar ligada a actividades perversas llevadas a cabo por atacantes que son consistentes en la creación de nuevos métodos para explotar vulnerabilidades de los dispositivos manejados por los usuarios. Es necesario prestar mucha atención al tráfico que circula en una red y analizar el impacto que puede llegar a tener un solo paquete de red comprometido en la misma.

Los sistemas de detección de intrusiones intentan analizar en tiempo real todo lo que sucede y pasa por la red, pero no siempre proporcionan una solución definitiva y precisa para este problema. El uso de la inteligencia supondría una gran ayuda para mejorar los sistemas de detección de intrusiones de ser aplicados de manera correcta a los datos generados por los dispositivos de red.

En un esfuerzo de proponer una opción prometedora a este problema, el propósito del presente trabajo fin de máster es comparar el potencial que puede tener el uso del aprendizaje automático en fuentes de datos en el campo de la ciberseguridad. El enfoque principal consistirá en una comparación del comportamiento de algoritmos de aprendizaje automático supervisado aplicado a las características extraídas de los flujos de red del conjunto de datos UGR’16, ya que contiene trazas de ataques recientes propiamente etiquetados.

Palabras clave

Inteligencia artificial, aprendizaje automático, bosque aleatorio, aumento del gradiente, redes neuronales artificiales, ciberseguridad, sistema de detección de intrusiones, UGR-16.

Abstract

The war against intrusion threats is fought every day by users and companies that want to feel safe against any cyber-attack that compromises the integrity of the users and computers in the network. The expansion of technological solutions may also be linked to perverse activities carried out by attackers that are consistent in the creation of new methods to exploit vulnerabilities of the devices managed by end users. It is necessary to pay close attention to the traffic passing through a network and weight the impact that a single compromised network package may have on it.

Intrusion detection systems try to analyze in real time any incidents that occur and the packets that go through the network, but these systems not always provide a definitive and accurate solution for this problem. The use of artificial intelligence would be of great help to enhance intrusion detection systems if applied correctly to the data generated by network devices.

To contribute with an option to this problem, the purpose of the present work is to compare the potential that machine learning algorithms may have in data sources related to the field of cybersecurity. The main approach will be a comparison of the performance of supervised machine learning algorithms applied to the features extracted from the network flows of the UGR'16 dataset, since it contains traces of properly labeled recent cyber-attacks.

Keywords

Artificial intelligence, machine learning, random forest, gradient boosting, artificial neural networks, cybersecurity, intrusion detection system, UGR-16.

Índice

Índice de figuras

Índice de tablas

Glosario

Adaline: ADAptative LINear Element (Elemento Lineal Adaptativo). Es de las primeras redes neuronales artificial de una sola capa desarrollada por el profesor Bernard Widrow y su alumno Ted Hoff en la Universidad de Stanford en 1960.

ART: Adaptive Resonance Theory (Teoría de Resonancia Adaptativa). Sección 2.4.

Backpropagation: La retropropagación es un método de cálculo del gradiente utilizado en algoritmos de aprendizaje supervisado utilizados para entrenar redes neuronales artificiales.

BAM: Bidirectional Associative Memory (Memoria Bidireccional Asociativa). Sección 2.4.

CERT: Computer Emergency Response Team (Equipo de respuesta de emergencia computacional). Sección 2.6.

Ciberespacio: entorno artificial que se desarrolla mediante herramientas informáticas

Ciberamenazas: actividades realizadas en el ciberespacio, que tienen como objeto la utilización de la información que circula por el mismo, para cometer distintos delitos.

Cibercriminales: Persona que comete crímenes por medio de Internet.

Cognitron: Una red neuronal multicapa auto-organizada. Sección 2.4.

DNN: Deep Neural Network (Red Neuronal Profunda). Sección 2.7.

DDoS: Distributed Denial of Service (Ataques de Denegación de Servicio Distribuidos). Sección 3.1.

DoS: Denial of Service (Ataques de Denegación de Servicio). Sección 3.1.

Feedforward: La prealimentación es una red neuronal artificial donde las conexiones entre las neuronas no forman un ciclo.

GB: Gradient Boosting (Aumento del Gradiente) Tabla 2.

Gini Impurity: es una medida de la frecuencia con la que un elemento elegido al azar del conjunto se etiquetaría incorrectamente si se etiquetara al azar de acuerdo con la distribución de las etiquetas en el subconjunto.

HIDS: Host-Based Intrusion Detection System (sistema de detección de intrusos basado en usuario). Sección 2.6.

HTTP: Hypertext Transfer Protocol (Protocolo de Transferencia de Hipertexto). Protocolo de comunicación que permite las transferencias de información en la *World Wide Web*.

HTTPS: Hypertext Transfer Protocol Secure (Protocolo de Transferencia de Hipertexto Seguro). Protocolo de comunicación que permite las transferencias de información en la *World Wide Web* de manera segura.

ICMP: Internet Control Message Protocol (Protocolo Mensajes de Control de Internet).

IDS: Intrusion Detection System (Sistema de Detección de Intrusiones).

IP: Internet Protocol (Protocolo de Internet).

ISACA: Information Systems Audit and Control Association (Asociación de Auditoría y Control de Sistemas de Información) Sección 2.5.

K-means: es un algoritmo de clasificación no supervisada que agrupa objetos en k grupos basándose en sus características. Tabla 2.

LQV: Learning Quantization Vector (Vector de cuantización de aprendizaje). Sección 2.4.

Madaline: Many ADAptative LINear Element (Multi Elemento Lineal Adaptativo). Es una red neuronal artificial de tres capas completamente conectada y de alimentación hacia delante.

MLP: Multi Layer Perceptron (Perceptrón Multicapa). Sección 2.7.

NIDS: Network-Based Intrusion Detection System (sistema de detección de intrusos basado en red). Sección 2.6.

Python: lenguaje de programación multiparadigma, interpretado, que hace uso de tipado dinámico y multiplataforma.

R2L: Remote to user attack (Ataque en Remoto). Sección 2.7.

RF: Random Forest (Bosque Aleatorio). Tabla 2.

SC: Spectral Clustering (Agrupamiento Espectral) Sección 2.7.

SPAM: los mensajes electrónicos no solicitados, habitualmente de tipo publicitario, enviados en forma masiva.

SSH: Secure Shell (Intérprete de Órdenes Seguro). Protocolo y programa que lo implementa que cuya aplicación más común es la ejecución remota de comandos.

SVM: Support Vector Machines (Máquinas de soporte vectorial). Tabla 2.

SYN: bit de control dentro del segmento **TCP**.

TCP: Transmission Control Protocol (Protocolo de Control de Transmisión).

U2R: User to Root attack (Ataque de usuario). Sección 2.7.

UDP: User Datagram Protocol (Protocolo de Datagramas de Usuario).

1. Introducción

1.1. Descripción general

El software malicioso, ataques cibernéticos como denegación de servicio y sistemas que evitan la intrusión de amenazas existen desde hace cierto tiempo y siguen mejorando, evolucionando y creciendo. Los sistemas de detección de intrusiones intentan hacer un seguimiento de estas nuevas tendencias emergentes, pero algunos no proporcionan una solución definitiva y precisa para este problema.

La ciberseguridad informática, es la práctica de defender los ordenadores, sistemas de redes y servidores de ataques maliciosos. Cada día que pasa la protección de los datos es más crítica dentro de las empresas, convirtiéndolo en uno de los retos más grandes para estas. Los métodos comunes que usan los ciber atacantes para controlar los ordenadores o redes incluyen virus, gusanos, spyware y troyanos. Los virus y los gusanos se pueden autorreplicar y dañar archivos, en tanto que el spyware y los troyanos a menudo se utilizan para la recopilación oculta de datos.

En este Trabajo Fin de Máster se planea trabajar con algoritmos de aprendizaje automático supervisado en un esfuerzo de proponer una solución prometedora a este problema. Ya que es difícil mantener las firmas de ataques cibernéticos actualizadas y mitigar estas amenazas antes de que sucedan, se puede intentar generalizar el comportamiento de estas amenazas con el uso del aprendizaje automático.

1.2. Justificación

Se plantea como propósito el estudio de los diferentes algoritmos de aprendizaje automático que existen y su viabilidad para detectar el comportamiento de amenazas sospechosas dentro de una red, valiéndose del uso de ejemplos que pueden ayudar a la detección de patrones dentro de los flujos de red, que luego serán empleados para generalizar este comportamiento.

Luego de analizar los diferentes algoritmos se procederá a su uso en el ámbito de la ciberseguridad. Se comparan los resultados obtenidos identificando los beneficios que este campo podría aportar y cuan prometedor es el uso del aprendizaje automático dentro de los sistemas de detección de intrusiones. Para realizar dicha solución se usarán datos etiquetados de una red monitorizada.

1.3. Objetivos y metodología

Los objetivos del trabajo son los siguientes:

- Estudiar las ventajas que proporciona el aprendizaje automático en la ciberseguridad y sus fuentes de datos.
- Selección y análisis de 3 algoritmos de aprendizaje automático para evaluar la viabilidad de su rendimiento en fuentes de datos de ciberseguridad.

Plan de investigación que se llevó a cabo:

- Análisis de las ventajas que se derivan del uso del aprendizaje automático y aprendizaje a profundidad.
- Análisis de la aplicación del aprendizaje automático en ciberseguridad.
- Análisis y selección de 3 algoritmos de aprendizaje automático para su evaluación como una solución tecnológica de ciberseguridad.
- Discusión de los resultados obtenidos y sus respectivas conclusiones.

1.4. Estructura del documento

El documento se estructura en 4 capítulos distribuidos de la siguiente forma:

- CAPÍTULO 1 – INTRODUCCIÓN: Incluye una descripción general del Trabajo Fin de Máster junto con la justificación del mismo. Indicando los objetivos y metodología empleada con un resumen de la estructura del documento.
- CAPÍTULO 2 – INTRODUCCIÓN TEÓRICA: Se establecen los fundamentos teóricos del aprendizaje automático, inteligencia artificial y la ciberseguridad. Además, se analizarán los principales algoritmos de aprendizaje automático utilizados, seleccionando 3 de ellos para su posterior aplicación en una solución tecnológica relacionada con la detección de intrusiones. Los algoritmos empleados se describirán a detalle. Por último, se hablará sobre el estado del arte de la inteligencia artificial en la ciberseguridad y sobre los posibles beneficios de emplearlo en el ámbito.
- CAPÍTULO 3 – DESARROLLO SOLUCIÓN: Incluye la implementación de los algoritmos seleccionados y su posible desempeño en un sistema de detección de intrusiones.
- CAPÍTULO 4 – CONCLUSIONES: Se discutirán los resultados del trabajo agregando posibles líneas de trabajo a futuro.

2. Introducción teórica

2.1. ¿Qué es la inteligencia artificial?

La inteligencia artificial es un amplio campo científico relativamente nuevo que aún cuenta con espacio para participar con nuevas ideas, uno de sus objetivos es intentar comprender la inteligencia de los seres humanos siendo capaz de construir entidades inteligentes.

Dentro de algunas de las definiciones que se esfuerzan en definir qué es la inteligencia artificial podemos encontrar 8 [1], descritas en la **Tabla 1**, que se dividen en dos dimensiones: Las definiciones superiores se preocupan por el *proceso de pensamiento* y *razonamiento* mientras que las definiciones inferiores se encargan del *comportamiento*. Las definiciones en la izquierda miden el éxito de los sistemas en función del rendimiento humano, mientras las definiciones de la derecha miden el éxito basándose en el rendimiento "ideal", es decir a qué medida toma la decisión correcta, que es llamado racionalidad.

Sistemas que piensan como humanos	**Sistemas que piensa racionalmente**
«El nuevo y excitante esfuerzo de hacer que los ordenadores piensen… *máquinas con mente*, en el más amplio sentido literal» (Haugeland, 1985) «[La automatización de] actividades que vinculamos con procesos de pensamiento humano, actividades como la toma de decisiones, resolución de problemas, aprendizaje…» (Bellman, 1978)	«El estudio de las facultades mentales mediante el uso de modelos computacionales» (Charniak y McDermott, 1985) «El estudio de los cálculos que permiten percibir, razonar y actuar (Winston, 1992)
Sistemas que actúan como humanos	**Sistemas que actúan racionalmente**
«El arte de desarrollar máquinas con capacidad para realizar funciones que cuando son realizadas por personas requieren de inteligencia» (Kurzweil, 1990) «El estudio de cómo lograr que los ordenadores realicen tareas que, por el momento, los humanos hacen mejor» (Rich y Knight, 1991)	«La Inteligencia Computacional es el estudio del diseño de agentes inteligentes» (Poole et al., 1998) «Inteligencia Artificial … está relacionada con conductas inteligentes en artefactos» (Nilsson, 1998)

Tabla 1. Algunas definiciones de inteligencia artificial organizadas por 4 categorías [1].

A lo largo de la historia, desde 1956 cuando se acuñó por primera vez el término de inteligencia artificial durante la conferencia Dartmouth organizada y propuesta por John McCarthy, Marvin L. Minsky, Nathaniel Rochester y Claude E. Shannon, los 4

enfoques descritos en la figura 1 se han seguido por diferentes investigadores que pueden pertenecer a cualquiera de las dos escuelas de pensamiento: Un enfoque centrado en los humanos, fundamentado por observaciones empíricas e hipótesis sobre el comportamiento humano, o un enfoque racional, que combina los campos de las matemáticas junto al de la ingeniería.

Para proporcionar una definición aceptable al enfoque que evalúa los sistemas por su capacidad de imitar el comportamiento humano, en 1950 Alan Turing propuso un examen, llamado Test de Turing. Este examen consiste en que una máquina tratara de exhibir un comportamiento inteligente similar al de un ser humano, respondiendo una serie de preguntas escritas, convenciéndolo de que no es una máquina con lo que está interactuando sino con otro ser humano.

En la actualidad si una máquina quiere emular el comportamiento humano sin ninguna interacción física, deberá poseer las siguientes capacidades [1]:

• **Procesamiento del lenguaje natural** para permitirle comunicarse con éxito en el idioma que se la programado;

• **Representación del conocimiento** para almacenar lo que aprende u oye;

• **Razonamiento automatizado** para usar la información almacenada para responder preguntas y formular nuevas conclusiones;

• **Aprendizaje automático** para adaptarse a las nuevas circunstancias detectando y extrapolando patrones.

Pero si lo que busca es una interacción completa del sistema inteligente con un ser humano, se necesita aplicar el Test Total de Turing que incluye dos ramas adicionales a la versión anterior:

•**Visión computacional** para percibir objetos, y

• **Robótica** para manipular objetos y moverse.

Estas 6 disciplinas comprenden en su mayoría el campo de inteligencia artificial, pero podemos dar crédito a aportes de otras ciencias en este tema como ser la psicología, ya que para crear máquinas que piensen y actúen como humanos debemos entender en su totalidad al mismo.

Los sistemas inteligentes no solo deben emular a los seres humano, también necesitan ser aptos tomando decisiones correctas, es un punto importante para tener en cuenta ya que es difícil elaborar la solución más racional si el entorno es muy complicado o si el sistema no cuenta con suficiente información. Un sistema racional actúa en función de un objetivo, el de conseguir el mejor resultado o si existe algo de incertidumbre, el mejor resultado esperado que sea aceptable.

2.2. Aprendizaje automático

El aprendizaje automático es una rama de la inteligencia artificial como se mencionó en el apartado anterior que permite a los ordenadores aprender, sin ser explícitamente programados para esto. El proceso de aprendizaje automático es similar al de la minería de datos, pero en lugar de extraer los datos para la comprensión humana –como es el caso de las aplicaciones de minería de datos– el aprendizaje automático utiliza los datos para inferir patrones en los datos y ajustar las futuras acciones del programa para conseguir el mejor rendimiento posible cuando es expuesto a nuevos datos.

El aprendizaje automático se adapta a muchos y diversos problemas, otorgándole un papel significativo por su versatilidad que proporciona apoyo en áreas como la extracción de información, recuperación de la información, interfaces de usuario adaptables, agentes inteligentes, robótica, etc.

La manera en que logra solucionar las tareas tiene está relacionado con la selección de las características apropiadas para construir los modelos. En esencia, las **características** definen un 'lenguaje' en el que describimos los objetos relevantes de nuestro dominio, es decir una representación adecuada de los objetos que serán estudiados. Una **tarea** es una representación abstracta de un problema que queremos resolver con respecto a esos objetos de dominio: la forma común de estos problemas suele necesitar ser clasificados en dos o más clases. Finalmente creamos un **modelo** como el resultado de salida de un algoritmo de aprendizaje automático que luego será aplicado a los datos de entrenamiento, este proceso está descrito en la **Figura 1**. [2]

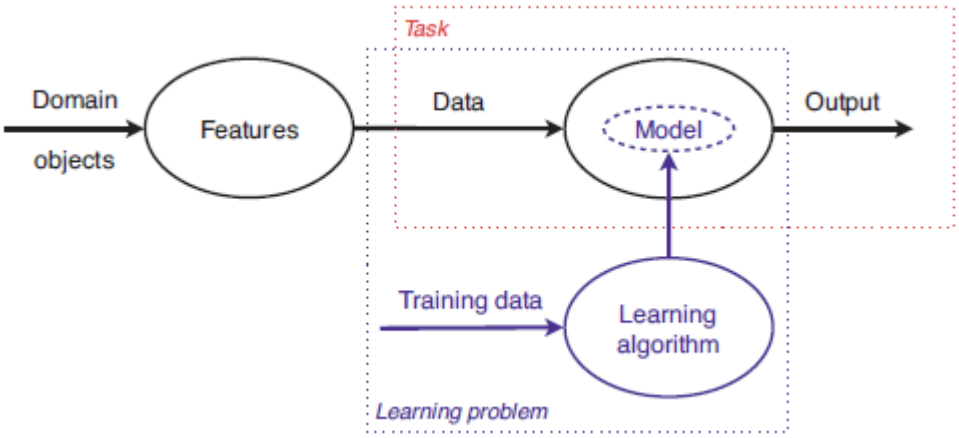

Figura 1. Visión general de cómo se utiliza el aprendizaje automático para abordar una tarea determinada. Una tarea (cuadro rojo) requiere un mapeo apropiado - un modelo - a partir de los datos descritos por las características a los resultados. La obtención de una asignación de este tipo a partir de los datos de capacitación es lo que constituye un problema de aprendizaje (recuadro azul) [2].

Como trataremos con una tarea de categorización, definiremos el tipo de aprendizaje en función del tipo de datos introducidos a los algoritmos durante la fase de entrenamiento. Podemos distinguir cuatro tipos de aprendizaje:

- **Aprendizaje supervisado:** En el aprendizaje supervisado, el algoritmo recibe una secuencia de entradas (que son los datos) y también se le da una secuencia de resultados deseados, el objetivo es aprender cómo producir una salida generalizada y correcta dada una nueva entrada.

- **Aprendizaje no supervisado:** En el aprendizaje no supervisado, el algoritmo simplemente recibe entradas, pero no obtiene las etiquetas de los resultados deseados, ni retroalimentación de su entorno. Sin embargo, es posible desarrollar un marco formal para el aprendizaje no supervisado basado en la noción de que el objetivo de la máquina es construir representaciones de los datos de entrada que puedan usarse para la toma de decisiones o predecir las entradas futuras. Básicamente el aprendizaje no supervisado trata de encontrar un significado oculto en los datos para ayudar a construir patrones a partir de lo que se puede considerar ruido puro no estructurado. Dos ejemplos clásicos muy simples de aprendizaje no supervisado son la agrupación y la reducción de la dimensionalidad.
- **Aprendizaje semi-supervisado:** En el aprendizaje semi-supervisado, también se trata de construir un modelo predictivo partir de un conjunto de datos. La particularidad de este conjunto de datos es que solo una pequeña parte tendrá un valor conocido de datos propiamente etiquetados, que son de mucha utilidad para crear un modelo inicial que luego puede afinarse utilizando el resto de los datos sin etiquetas, ayudando a mejorar la exactitud de los modelos basados en aprendizaje no supervisado.
- **Aprendizaje por Refuerzo:** En el aprendizaje por refuerzo el algoritmo trabaja con recompensas acumuladas al realizar bien una tarea dentro del entorno planteado siguiendo un conjunto de reglas previamente definidas. Es un área inspirada en la conducta y es utilizada en campos de teoría de juegos, teoría de control y algoritmos genéticos [1].

Existen muchos algoritmos de aprendizaje automático que pueden ser usados, algunos de los más relevantes están descritos en la **Tabla 2**.

Este trabajo fin de máster se valdrá de las técnicas de aprendizaje automático para analizar un conjunto de datos de ataques de ciberseguridad. Esta rama de la inteligencia artificial nos da libertad de crear modelos capaces de adaptarse a nuevas circunstancias, como lo son las amenazas en a seguridad informática. Los algoritmos de agregación y redes neuronales artificiales han sido los seleccionados para desarrollar los experimentos y por lo tanto serán comentados más en profundidad en la siguiente sección.

Algoritmos	Descripción	Tipo de Aprendizaje	Tipo de Modelo
Máquinas de soporte vectorial (**SVM** - *Support Vector Machines*)	Separa los datos en clases lo mediante el uso de hiperplanos en el dominio.	Aprendizaje Supervisado	Clasificación
Clasificador Naïve Bayes	Clasificador probabilístico fundamentado en el teorema de Bayes.	Aprendizaje Supervisado	clasificación
K-Vecinos más cercanos (*K-Nearest Neighbor*)	Divide en clases midiendo la distancia entre las muestras introducidas para asignar el valor más adecuado.	Aprendizaje Supervisado	Clasificación
Random Forest	Meta estimador que se ajusta a una serie de árboles de decisión en varias submuestras del conjunto de datos y utiliza un promedio para mejorar la precisión predictiva y controlar el sobre ajuste.	Aprendizaje Supervisado	Clasificación / Regresión
Árboles de decisión	El objetivo es crear un modelo que prediga el valor de una variable mediante el aprendizaje de reglas simples de decisión inferidas a partir de las características de los datos.	Aprendizaje Supervisado	Regresión
Regresión lineal	Aproxima los datos de entrada a una ecuación lineal con fines predictivos.	Aprendizaje Supervisado	Regresión
Redes Neuronales Artificiales	Conjunto de neuronas que funcionan conectadas entre sí imitando el comportamiento del cerebro para realizar una tarea.	Aprendizaje supervisado/no supervisado	Regresión /Agrupamiento
K-means	Particiona los datos en grupos de *k* centros agrupándolos por distancia entre las muestras.	Aprendizaje no Supervisado	Agrupamiento
Algoritmos de Ensamblaje	Métodos que combinan distintos algoritmos de aprendizaje para obtener un mejor resultado que utilizándolos por separado.	Aprendizaje Supervisado	Regresión
Algoritmos genéticos	Algoritmos de computación evolutiva que realizan procesos de búsqueda heurística que simulan la selección natural.	Aprendizaje por Refuerzo	Búsqueda

Tabla 2.Algunos algoritmos de aprendizaje automático.

2.3. Árboles de decisión

Los modelos de árboles de decisión se encuentran entre los modelos más populares en el aprendizaje automático. Los árboles son expresivos y fáciles de entender, y de particular atractivo para los investigadores por su naturaleza recursiva y estrategia de ‘divide y vencerás'. Como su nombre lo indica crea un modelo de decisiones con la forma de un árbol y el algoritmo puede cubrir tareas de clasificación y de regresión.

El algoritmo central se basa en ID3 propuesto por J. R. Quinlan [3] que emplea una búsqueda codiciosa de arriba hacia abajo a través del espacio de posibles ramas sin retroceso. ID3 usa las características de la ***entropía*** y ***ganancia de información*** para construir un árbol de decisión.

Para dar un ejemplo más claro consideremos el siguiente escenario para predecir el pronóstico del día. El modelo utilizara 3 atributos como ser soleado, nublado y lluvia.

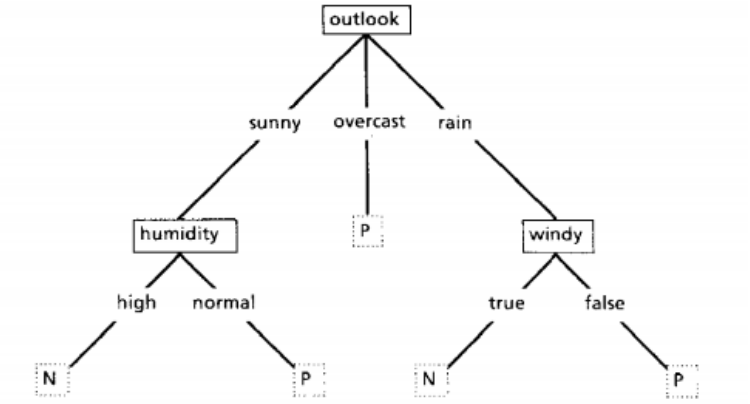

Figura 2. Ejemplo de un árbol de decisión simple [3].

Un árbol de decisión se construye de arriba hacia abajo desde un nodo raíz, en este caso “pronóstico” en la cabeza. El texto rodeado por un cuadrado la **Figura 2** representa una condición o nodo interno, que luego se bifurca en dos ramas o bordes. Al final de cada rama que no se bifurca más podemos encontrar la decisión u hoja que ha clasificado el algoritmo en este ejemplo la predicción sobre si el pronóstico del día es positivo o negativo para humedad, representado por las letras P y N respectivamente.

2.3.1 Entropía

Un árbol de decisión se construye de arriba hacia abajo desde un nodo raíz e implica dividir los datos en subconjuntos que contienen instancias con valores similares (homogéneos). El algoritmo ID3 usa entropía para calcular la homogeneidad de una muestra. Si la muestra es completamente homogénea, la entropía es cero y si la muestra está dividida por igual, tiene entropía de uno [4].

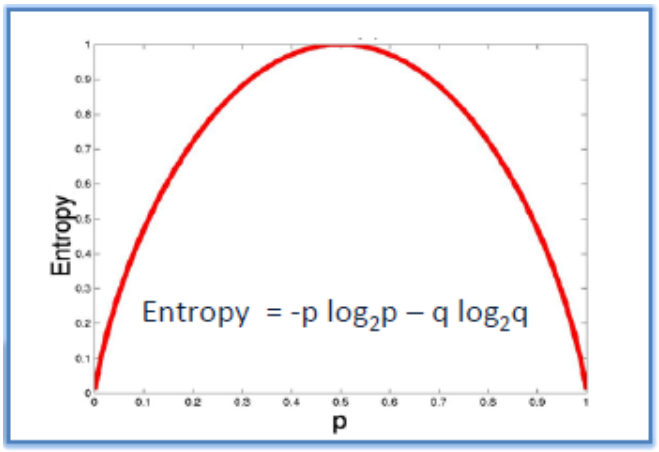

Figura 3. Entropía= -0.5 □□□□□.□ − □.□ □□□□□.□ = □ [4].

Para construir un árbol de decisión, necesitamos calcular dos tipos de entropía usando tablas de frecuencia de la siguiente manera:
a) Entropía usando la tabla de frecuencias de un atributo:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

Figura 4.Ejemplo de entropía usando la tabla de frecuencia de un atributo [4].

b) Entropía usando la tabla de frecuencias de dos atributos:

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)
= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971
= 0.693

Figura 5.Ejemplo de entropía usando la tabla de frecuencia de dos atributos [4].

2.3.2 Ganancia de información

La ganancia de información se basa en la disminución de la entropía después de que un conjunto de datos se divide en un atributo. La construcción de un árbol de decisión se trata de encontrar un atributo que devuelva la mayor ganancia de información (es decir, las ramas más homogéneas).

2.4 Bosque aleatorio (random forest)

El algoritmo de Bosque Aleatorio, al cual nos referiremos como **RF** de ahora en adelante por sus siglas en inglés, es un algoritmo de clasificación supervisado. Podemos inferir por su nombre, que crear un bosque de árboles de decisión de manera aleatoria, utilizando el método de "agregación". La idea general del método de agregación es que el uso de una combinación de modelos de aprendizaje automático aumenta la precisión del resultado general. Existe una relación directa entre la cantidad de árboles de decisiones en el bosque y los resultados que se pueden obtener: cuanto mayor sea la cantidad de árboles, más preciso será el resultado. Pero una cosa para tener en cuenta es que crear el bosque no es lo mismo que construir la decisión con un enfoque de ganancia de información o índice de ganancia [5].

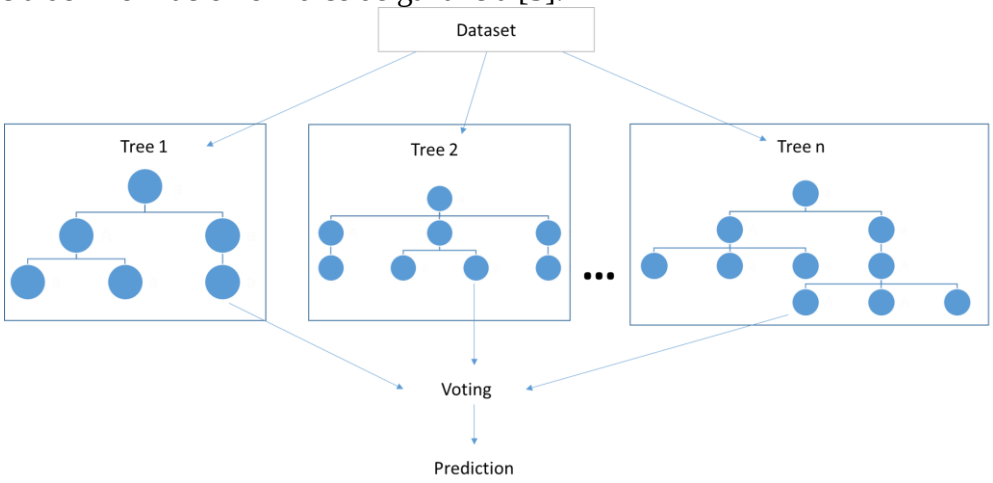

Figura 6. Estructura del algoritmo RF.

Con algunas excepciones, un clasificador de bosque aleatorio tiene todos los hiperparámetros de un árbol de decisión y también todos los hiperparámetros de un modelo de agregación, para controlar todos los árboles del método de agregación.

En lugar de buscar la mejor característica al dividir un nodo, busca la mejor característica entre un subconjunto aleatorio de características. Este proceso crea una gran diversidad, que generalmente resulta en un mejor modelo.

Por lo tanto, cuando está creciendo un árbol de decisión dentro de un bosque aleatorio, solo se considera un subconjunto aleatorio de las características para dividir un nodo. Incluso puede hacer árboles más aleatorios, mediante el uso de umbrales aleatorios en la parte superior, para cada característica en lugar de buscar los mejores umbrales posibles (como lo hace un árbol de decisión normal) de esta manera trata de evitar el sobre ajuste al conjunto de datos. Para dar el resultado final internamente realiza una votación de las predicciones de todos los árboles de decisión en el conjunto y entrega la salida de su proceso.

2.5 Aumento de gradiente (gradient boosting)

Aumento de gradiente, al cual nos referiremos como **GB** de ahora en adelante por sus siglas en inglés, es un algoritmo de aprendizaje que se puede utilizar en tareas de clasificación y regresión, el modelo creado es un conjunto de algoritmos con predicciones débiles, típicamente árboles de decisión. **GB** es un ejemplo de un algoritmo que se vale de la técnica de potenciación, esta técnica emplea una lógica en la que las predicciones posteriores aprenden de los errores de las predicciones previas, es decir corrige los errores de las predicciones de manera secuencial.

Por lo tanto, las instancias tienen una probabilidad desigual de aparecer en los modelos posteriores y las que tienen el error más alto aparecen más. Los algoritmos de predicción se pueden elegir de una gama de modelos como árboles de decisión, regresores, clasificadores, etc. Debido a que los nuevos predictores están aprendiendo de los errores cometidos por los predictores previos, se requiere menos tiempo o iteraciones para alcanzar las predicciones reales. Pero tenemos que elegir cuidadosamente los criterios de detención o podría llevar a un sobreajuste en los datos de entrenamiento.

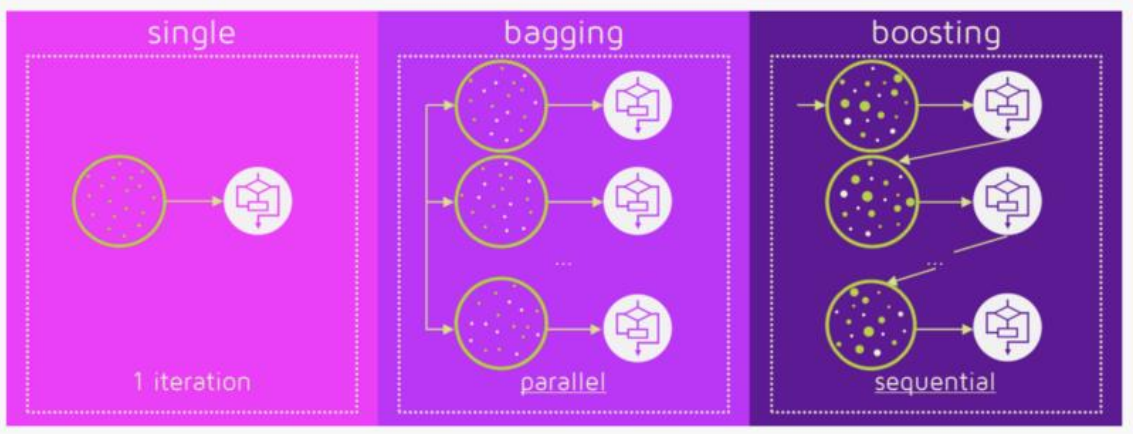

Figura 7.Método de un solo modelo con una iteración, método de agregación en paralelo con modelos independientes y método de potenciamiento con modelos secuenciales [6].

El algoritmo de GB involucra tres elementos [7]:

1. Una función de pérdida para ser optimizada.
2. Un algoritmo de predicción débil para crear las predicciones.
3. Un modelo aditivo para agregar los algoritmos débiles y minimizar la función de pérdida.

2.5.1. Función de pérdida

La función de pérdida utilizada depende del tipo de problema que se está resolviendo. Debe ser diferenciable, pero se admiten muchas funciones de pérdida estándar y tenemos la libertad de definir la nuestra. Por ejemplo, la regresión puede usar un error al cuadrado y la clasificación puede usar pérdida logarítmica.

Un beneficio de la estructura de GB es que no es necesario derivar un nuevo algoritmo de potenciación para cada función de pérdida que pueda desearse utilizar, sino que es

una estructura lo suficientemente genérica como para que se pueda utilizar cualquier función de pérdida diferenciable.

2.5.2. Algoritmo de predicción débil

Los árboles de decisión se utilizan como los algoritmos de predicción débiles por preferencia en GB. Específicamente, se utilizan árboles de regresión que generan valores reales para divisiones y cuyos resultados se pueden sumar, lo que permite agregar salidas de modelos posteriores y "corregir" los residuos en las predicciones.

Los árboles se construyen de manera codiciosa, eligiendo los mejores puntos de división basados en los puntajes de pureza (entropía) como **Gini** o para minimizar la pérdida.

Es común restringir a los algoritmos dentro del conjunto de maneras específicas, como un número máximo de capas, nodos, divisiones o nodos de hojas para asegurar que los algoritmos de aprendizaje sigan siendo débiles, pero aun así se pueda construir de manera codiciosa.

2.5.3. Modelo aditivo

Los árboles se agregan uno a la vez, y los árboles existentes en el modelo no se cambian. Un procedimiento de ***descenso de gradiente*** se utiliza para minimizar la pérdida al agregar árboles, definido como:

$$□(□) = □_{□□□□1}(□) + □_{□□□□2}(□)+…$$

Tradicionalmente, el descenso de gradiente se utiliza para minimizar un conjunto de parámetros, como los coeficientes en una ecuación de regresión o ponderaciones en una red neuronal. Después de calcular el error o la pérdida, los pesos se actualizan para minimizar ese error, que agregaría los resultados al siguiente algoritmo creado.

En lugar de parámetros, tenemos submodelos de aprendizaje débiles o, más específicamente, árboles de decisión. Después de calcular la pérdida, para realizar el procedimiento de descenso de gradiente, debemos agregar un árbol al modelo que reduce la pérdida (es decir, seguir el gradiente) definido como:

$$□(□) + □_{□□□□3}(□) = □(□).$$

Hacemos esto parametrizando el árbol, luego modificamos los parámetros del árbol y nos movemos en la dirección correcta (reduciendo la pérdida residual) definido como:

$$□(□) = □(□) - □(□).$$

La salida para el nuevo árbol se agrega luego a la salida de la secuencia existente de árboles en un esfuerzo por corregir o mejorar el resultado final del modelo.

Se agrega una cantidad fija de árboles o el entrenamiento se detiene una vez que la pérdida alcanza un nivel aceptable o ya no mejora en un conjunto de datos de validación externo.

2.4. Redes neuronales artificiales

El objetivo que da origen a las Redes Neuronales Artificiales, a las cuales nos referiremos como **RNA** de ahora en adelante, es construir un modelo que sea capaz de emular el método de aprendizaje del cerebro humano, estas redes fueron introducidas por McCulloch y Pitts en 1943[1]. Las células encargadas de este aprendizaje son las neuronas interconectadas entre sí a través de complejas redes.

Conociendo un poco de biología básica, sabemos que en el proceso de sinapsis se establece la posibilidad de "transmisión de información" entre unas neuronas y otras (desde las terminaciones en las que se ramifica el axón de una neurona hacia las dendritas de otra); cuando el estímulo llega a un terminal nervioso, hace que el nervio libere neurotransmisores representado en la **figura 8**. Dependiendo del tipo de neurotransmisor liberado, las neuronas receptoras pueden activarse si llegan a recibir el estímulo de las neuronas con las que está conectada o inhibirse si dicha información no llega a recibirse, generando una respuesta de uno u otro tipo en cada caso.

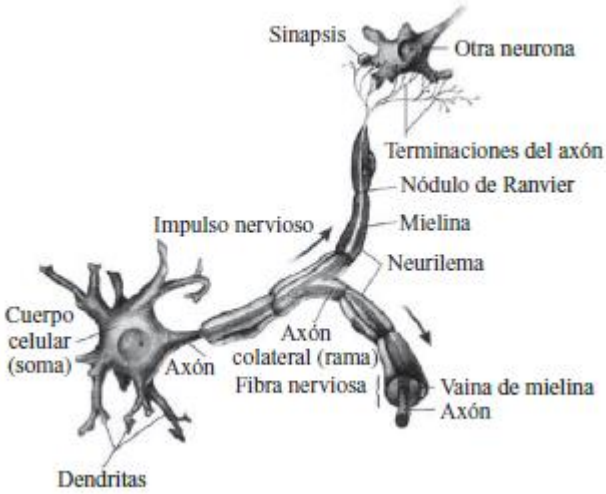

Figura 8. Elementos de una Neurona [9]

Así mismo, también son capaces de extraer patrones y detectar tramas que son muy difíciles de apreciar por otras técnicas computacionales, siendo la versatilidad al momento de implementación de estos métodos lo que ha facilitado su popularidad. Estas RNA se caracterizan por tres partes fundamentales: la topología de la red (propagación hacia adelante-***feedforward***- o recurrentes), la regla de aprendizaje (supervisado, no supervisado, reforzado, etc…) y el tipo de entrenamiento [9].

Las RNA, se componen de neuronas, que serían las unidades básicas del modelo. En la **Figura 9** se puede ver su modelo y los distintos elementos que la componen:

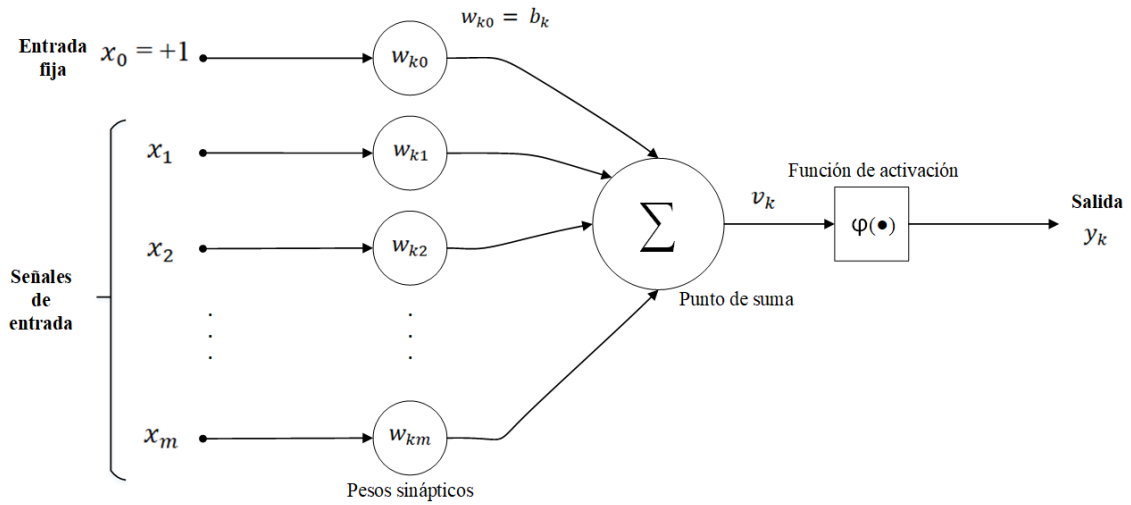

Figura 9. Modelo de una neurona [9]

1. Un conjunto de unidades de procesamiento o neuronas.

2. Un estado de activación para cada unidad, equivalente a la salida de la unidad.

3. Conexiones entre las unidades, generalmente definidas por un peso que determina el efecto de una señal de entrada en la unidad.

4. Una regla de propagación, que determina la entrada efectiva de una unidad a partir de las entradas externas.

5. Una función de activación que actualiza el nuevo nivel de activación basándose en la entrada efectiva y la activación anterior.

6. Una entrada externa que corresponde a un término determinado como sesgo para cada unidad.

7. Un método para reunir la información, correspondiente a la regla del aprendizaje

8. Un ambiente en el que el sistema va a operar, con señales de entrada e incluso señales de error.

Trataremos de explicar el proceso de manera “sencilla”: cada neurona recibe una serie de entradas, que llevarán un peso, emitiendo una salida. La salida es dada por tres funciones fundamentales:

- **La función de propagación:** suele ser la sumatoria de cada entrada multiplicada por el peso asignado.
- **La función de activación:** tiene como misión modificar a la de propagación. No siempre aparece, coincidiendo en estos casos con la dicha propagación. Las más habituales suelen ser la función tipo escalón (Heaviside) o funciones no lineales como la sigmoidea (parecida al escalón, pero suavizada), logística, tangente hiperbólica…
- **La función de transferencia:** se aplica al valor dado por la función de aplicación y se utiliza para acotar la salida de cada neurona según la interpretación que le queramos dar al resultado.

En muchas redes las unidades de proceso entregan su respuesta de siguiente la forma:

$$\square = \square\left(\sum_{\square} \quad \square_{\square}\square_{\square}\right)$$

Donde:

$\square_{\square}$: Señales de salida de otros nodos o entradas externas.
$\omega_{\square}$: Pesos de las ligas de conexión.
$\square$ (•): Función no lineal simple.

La función f puede ser sigmoidal, tangente hiperbólica, escalón, entre otras. Cada unidad de proceso tiene una tarea: recibir la entrada de otras unidades o de fuentes externas y procesar la información para obtener una salida que se propaga a otras unidades.

Una red puede tener una estructura arbitraria, pero las capas que contienen estas estructuras están definidas de acuerdo con su ubicación en la topología de la red neuronal. Las entradas externas son aplicadas en la primera capa, y las salidas se consideran la última capa. Las capas internas que no se observan como entradas o salidas se denominan ***capas ocultas***. Por convención, las entradas no se consideran como capa porque no realizan procesamiento.

La entrada total u de una unidad k es la suma de los pesos de las entradas conectadas, más un sesgo (**bias**) θ:

$$\square = \sum_{\square} \quad \square_{\square}\square_{\square} + \square.$$

Si el peso $\omega_{\square}$ es positivo se habla de una excitación y si el peso es negativo se considera una inhibición de la entrada. Si consideramos a las entradas como funciones del tiempo, la expresión anterior se convierte en:

$$\square(\square) = \sum_{\square} \quad \square_{\square}(\square)\square_{\square}(\square) + \square(\square).$$

2.4.1 Funciones de Activación

La regla que logra establecer el efecto de la entrada total u(t) en la activación de la unidad k se denomina función de activación ($\square_{\square}$).

$$\square(\square + 1) = \square_{\square}(\square(\square), \square(\square)).$$

En muchas ocasiones esta función es de la forma no decreciente respecto a la entrada total de la unidad:

$$\square(\square + 1) = \square\left(\sum_{\square} \quad \square_{\square}(\square)\square_{\square}(\square) + \square(\square)\right).$$

- **Función escalón**

La función de activación escalón se asocia a neuronas binarias en las cuales, cuando la suma de las entradas es mayor o igual que el umbral de la neurona, la activación es 1; si es menor, la activación es 0 (o 21) (véase la **figura 10**).

- **Función lineal y mixta (Relu)**

La función lineal o identidad responde a la expresión Fk(u) 5 u. En las neuronas con función mixta, si la suma de las señales de entrada es menor que un límite inferior, la función se define como 0 (o 21). Si dicha suma es mayor o igual que el límite superior, entonces la activación es 1. Si la suma de entrada está comprendida entre los dos límites, entonces la activación es 1. Si la suma de entrada está comprendida entre ambos límites, superior e inferior, entonces la activación se define como una función lineal de la suma de las señales de entrada (véase la **figura 10**).

- **Función tangente hiperbólica**

La función de activación tangente hiperbólica se emplea en los casos que presentan variaciones suaves de valores positivos y negativos de la señal a clasificar. Como se puede ver en su descripción en la figura 11, es una de las funciones más empleadas en entrenamientos supervisados, como en el caso del entrenamiento de retropropagación del error. Debe tenerse cuidado de emplear esta figura entre los umbrales positivos y negativos antes de la saturación, de otra forma la salida siempre generará valores saturados iguales a 1 y 21.

- **Función sigmoidal**

Con la función sigmoidal el valor dado por la función es cercano a uno de los valores asintóticos. Esto hace que, en la mayoría de los casos, el valor de salida esté comprendido en la zona alta o baja del sigmoide. De hecho, cuando la pendiente es elevada, esta función tiende a la función escalón. Sin embargo, la importancia de la función sigmoidal es que su derivada siempre es positiva y cercana a cero para los valores grandes positivos o negativos; además, toma su valor máximo cuando x 5 0. Esto hace que se puedan utilizar reglas de aprendizaje definidas para las funciones escalón, con la ventaja, respecto a esta función, de que la derivada está definida en todo el intervalo (véase la **figura 10**).

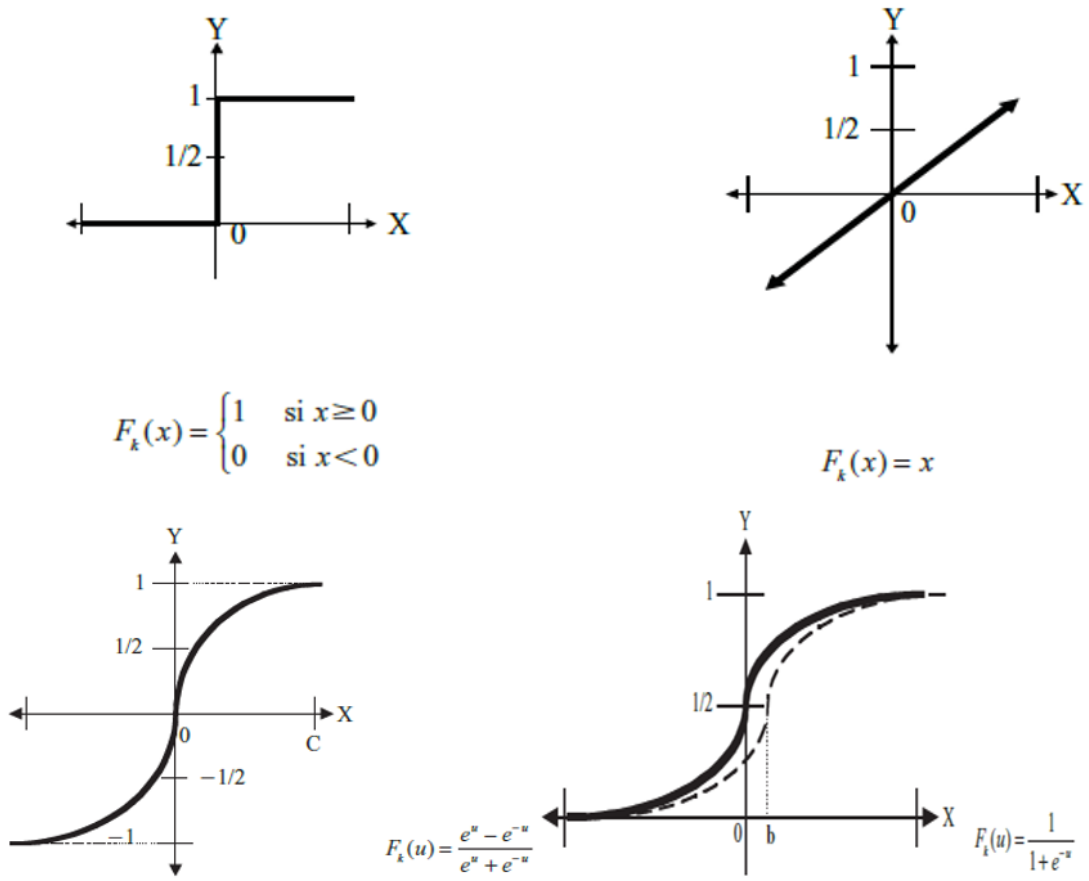

Figura 10. Funciones de activación para entrenar modelos de RNA (en orden de izquierda a derecha): Función escalón, función lineal, función tangente hiperbólica y función sigmoidal [9].

El aprendizaje de la red neuronal artificial se basa en la variación de sus parámetros, específicamente los pesos entre neuronas y el umbral de polarización. Esta variación de parámetros denominada aprendizaje se realizará por medio de los algoritmos de aprendizaje, se mencionan algunos a continuación:

- Corrección del error: minimiza una función de coste (criterio) basada en la señal de error, de tal modo que la respuesta de cada neurona de salida de la red se aproxime lo máximo posible a la respuesta deseada. La señal de error se define como la diferencia entre la respuesta deseada y la respuesta real.

- Regla de Hebb: se trata del método de aprendizaje más antiguo y es un mecanismo que varía en el tiempo, local y altamente interactivo que incrementa la eficiencia de la sinapsis como una función de la correlación entre las actividades pre y pos sinápticas.

- Aprendizaje competitivo: en este tipo de aprendizaje las neuronas de la capa de salida compiten entre sí para ser la única ganadora. Esta forma de aprendizaje es especialmente ventajosa para descubrir características importantes en el reconocimiento

de patrones ya que cada neurona se especializa en detectar una característica en particular.

El algoritmo de aprendizaje más popular en las redes neuronales es el de corrección del error, se llevará a cabo mediante diferentes métodos de optimización que se ejecutarán de forma iterativa. Algunos de estos métodos de optimización son:

- Descenso del gradiente: algoritmo donde la búsqueda de un mínimo está asociada a la resolución secuencial de una serie de problemas unidimensionales. En este método, la dirección de descenso viene indicada por el negativo del gradiente de la función objetivo.

- Adagrad: algoritmo que permite a la tasa de aprendizaje adaptarse basándose en los parámetros del modelo, realizando grandes ajustes para los parámetros poco frecuentes y ajustes pequeños para los parámetros más frecuentes. Por esta razón es ideal para manejar datos dispersos con la desventaja que la tasa de aprendizaje se adapta solamente decreciendo.

- AdaDelta: es una extensión del algoritmo Adagrad que tiende eliminar el problema de la tasa de aprendizaje desvaneciente, ya que calcula la tasa para cada parámetro limitando el número de gradientes pasados acumulados, se ha mostrado robusto frente a entornos ruidosos.

- Adam: se trata de un método de optimización estocástica eficiente que únicamente requiere gradientes de primer orden y tiene un requerimiento bajo de memoria. Es otro método que adapta las tasas de aprendizaje para cada parámetro. Además de almacenar un promedio exponencialmente en descomposición de gradientes cuadrados anteriores como AdaDelta, Adam también mantiene un promedio exponencialmente decreciente de gradientes pasados.

En cuanto a redes neuronales, se pueden diferenciar tres clases de arquitecturas: redes neuronales de una capa, redes neuronales artificiales multicapa y redes neuronales artificiales recurrentes. La arquitectura de una red neuronal definirá la manera en la que las neuronas están ordenadas.

En cuanto a redes neuronales, se pueden diferenciar tres clases de arquitecturas: redes neuronales de una capa, redes neuronales multicapa y redes neuronales recurrentes. La arquitectura de una red neuronal definirá la manera en la que las neuronas están ordenadas.

2.4.2 REDES NEURONALES DE UNA CAPA

El tipo de estructura más simple son las redes neuronales de una capa ya que solo cuenta con la capa de salida, ya que como se mencionó anteriormente la capa de entrada no se toma en cuenta. La **Figura 11** ilustra un ejemplo de esta arquitectura de red donde

las distintas entradas (situadas en la capa de entrada) se conectan con las neuronas de la única capa existente (capa de salida) pero no viceversa, por lo tanto, nos encontramos ante una red denominada *feedforward*- propagación hacia adelante-.

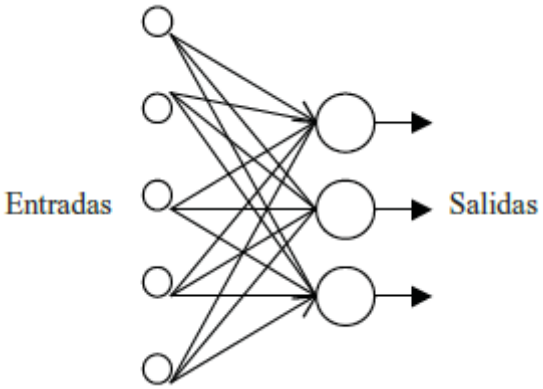

Figura 11. Red neuronal de una capa

2.4.3 REDES NEURONALES MULTICAPA

Este tipo de redes será similar a las redes neuronales de una capa, pero contarán con una o más capas entre la entrada y salida que se convierte en ***capas ocultas***. La existencia de capas ocultas permite a la red obtener estadísticas de orden mayor al de sus entradas, este hecho permite a la red adquirir una perspectiva global a pesar de su conectividad local. La **Figura 12** muestra un ejemplo de red neuronal de dos capas, completamente conectada.

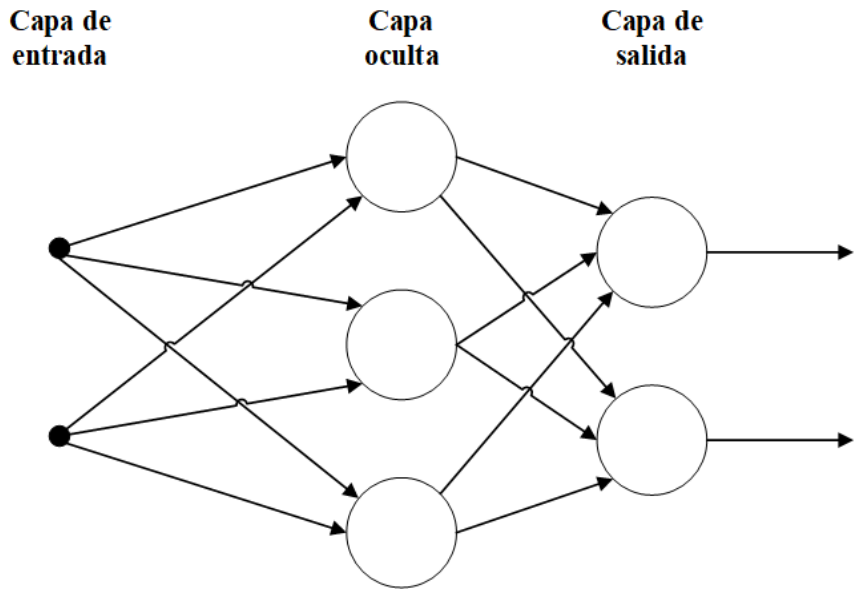

Figura 12.Red neuronal multicapa

- **Redes con conexiones hacia adelante**:

Este tipo de redes contienen solo conexiones entre capas hacia delante. Esto implica que una capa no puede tener conexiones a una que reciba la señal antes que ella en la dinámica de la computación. Ejemplos de estas redes son Perceptron, ***Adaline***, ***Madaline***, ***Backpropagation*** y los modelos LQV y TMP de Kohonen [10].

- **Redes con conexiones hacia atrás:**

Este tipo de redes se diferencian con las anteriores en que, si pueden existir conexiones de capas hacia atrás y por tanto la información puede regresar a capas anteriores en la dinámica de la red, este tipo de RNA también son conocidas como redes neuronales competitivas. Este Tipo de redes suelen ser bicapas algunos ejemplos de estas redes son las redes ***ART***, Bidirectional Associative Memory (BAM) y Cognitron [11].

2.5. Ciberseguridad informática

En la actualidad, un término ampliamente utilizado es "ciberseguridad", que puede asociarse con otras palabras como **ciberespacio**, **ciberamenazas**, **cibercriminales** u otros conceptos compuestos. Aunque se tiene una percepción general sobre lo que representa, en ocasiones puede utilizarse como sinónimo de seguridad de la información, seguridad informática o seguridad en cómputo -pero esta idea no es del todo correcta [12].

En la pasada edición de Secure Conference, profesionales de seguridad de **ISACA** (*Information Systems Audit and Control Association*) capítulo Monterrey, comenzaron su participación a partir de definir qué es la ciberseguridad. De acuerdo con la asociación, puede entenderse como:

"Protección de activos de información, a través del tratamiento de amenazas que ponen en riesgo la información que es procesada, almacenada y transportada por los sistemas de información que se encuentran interconectados" [12].

La norma ISO 27001 [13] define activo de información como los conocimientos o datos que tienen valor para una organización, mientras que los sistemas de información comprenden a las aplicaciones, servicios, activos de tecnologías de información u otros componentes que permiten el manejo de la misma.

Por lo tanto, la ciberseguridad tiene como foco la protección de la información digital que "vive" en los sistemas interconectados. En consecuencia, está comprendida dentro de la seguridad de la información.

2.5.1 SEGURIDAD DE LA INFORMACIÓN: DISTINTAS FORMAS Y ESTADOS DE LOS DATOS

Para conocer la diferencia principal con la seguridad de la información, revisemos otros conceptos interesantes que nos permitirán tener el contexto general. De acuerdo con la Real Academia Española (**RAE**), la seguridad se define como "libre o exento de todo peligro, daño o riesgo". Sin embargo, se trata de una condición ideal, ya que en la

realidad no es posible tener la certeza de que se pueden evitar todos los peligros. Con este concepto el propósito de la seguridad es reducir riesgos hasta un nivel aceptable para los interesados y mitigar amenazas latentes.

Según la IS0-27002 [13], “La seguridad de la información se puede caracterizar por la preservación de:

• **Confidencialidad**: asegura que el acceso a la información está adecuadamente autorizado.

• **Integridad**: salvaguarda la precisión y completitud de la información y sus métodos de proceso

• **Disponibilidad**: Asegura que los usuarios autorizados pueden acceder a la información cuando la necesitan”

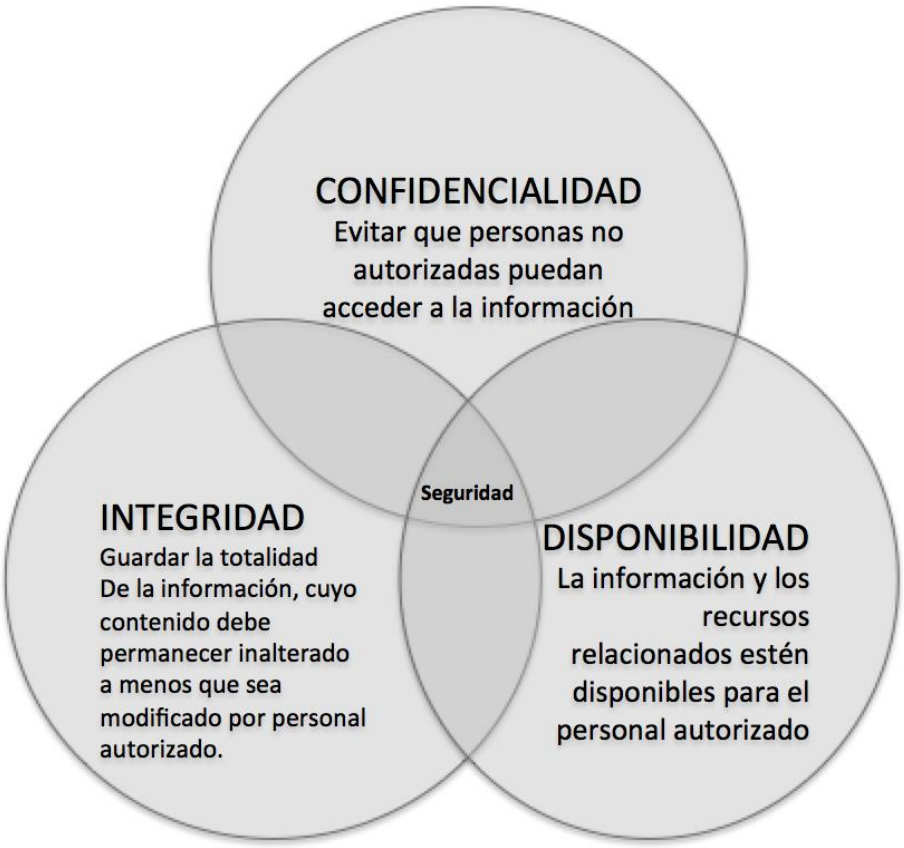

Figura 13. Objetivos de la seguridad informática [14]

Otra de las definiciones de lo seguridad informática dada por INFOSEC Glossary 2000 [15]: “Seguridad Informática son las medidas y controles que aseguran la confidencialidad, integridad y disponibilidad de los activos de los sistemas de información, incluyendo hardware, software, firmware y aquella información que

procesan, almacenan y comunican". De estas definiciones podemos deducir que los principales objetivos de la seguridad informática son [16]:

• **Confidencialidad**: consiste en la capacidad de garantizar que la información, almacenada en el sistema informático o transmitido por la red, solamente va a estar disponible para aquellas personas autorizadas a acceder a dicha información.

• **Disponibilidad**: la definiremos como la capacidad de garantizar que tanto el sistema como los datos van a estar disponibles al usuario en todo momento. Pensemos, por ejemplo, en la importancia que tiene este objetivo para una empresa encargada de impartir ciclos formativos a distancia. Constantemente está recibiendo consultas, descargas a su sitio web, etc., por lo que siempre deberá estar disponible para sus usuarios.

• **Integridad**: diremos que es la capacidad de garantizar que los datos no han sido modificados desde su creación sin autorización. La información que disponemos es válida y consistente. Se deberá garantizar que ningún intruso pueda capturar y modificar los datos en tránsito.

• **No repudio:** este objetivo garantiza la participación de las partes en una comunicación. En toda comunicación, existe un emisor y un receptor, por lo que podemos distinguir dos tipos de no repudio:

a) No repudio en origen: garantiza que la persona que envía el mensaje no puede negar que es el emisor de este, ya que el receptor tendrá pruebas del envío.

b) No repudio en destino: El receptor no puede negar que recibió el mensaje, porque el emisor tiene pruebas de la recepción de este.

•**Autenticación:** La autenticación consiste en la confirmación de la identidad de un usuario; es decir, la garantía para cada una de las partes de que su actor es realmente quien dice ser.

2.6. Los sistemas de detección de intrusos (IDS)

En este Trabajo Fin de Máster se ha evaluado una comparación que tiene como objetivo proteger los activos de la información de ataques mediante el uso de algoritmos de aprendizaje dentro de los sistemas de detección de intrusos.

Un Sistema de Detección de Intrusos o IDS (Intrusion Detection System) es una herramienta de seguridad encargada de monitorizar los eventos que ocurren en un sistema informático en busca de intentos de intrusión.
La detección de intrusiones permite a las organizaciones proteger sus sistemas de las amenazas que aparecen al incrementar la conectividad en red y la dependencia que tenemos hacia los sistemas de información. Los IDSs han ganado aceptación como una pieza fundamental en la infraestructura de seguridad de la organización.

Existen varias propuestas sobre la arquitectura de los IDSs pero la arquitectura de un **IDS** debe contar con una fuente de datos donde se recogerán los eventos del sistema

monitorizado (por ejemplo paquetes de red o eventos de los sistemas operativos), un motor que analizará los datos en busca de evidencia, una base de datos, patrones de uso y tipos de ataques y por último un módulo de respuesta. Dicha arquitectura se muestra en la **Figura 14**.

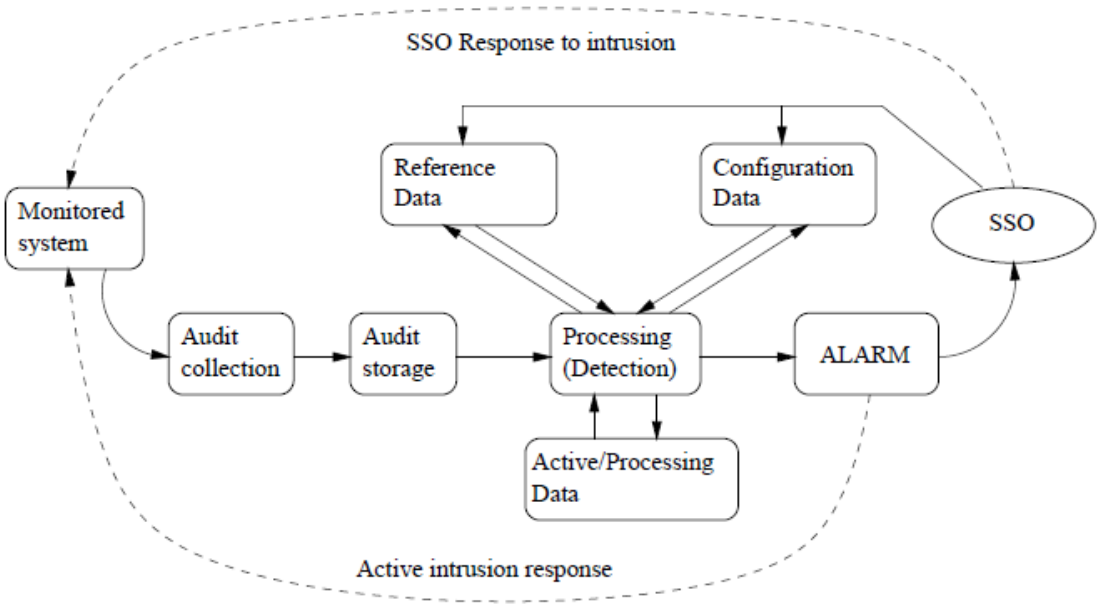

Figura 14.Arquitectura General de un IDS [17].

Los sistemas de IDS están disponibles en diferentes tipos; los dos tipos principales son el sistema de intrusión basado en el host (HIDS) y el sistema de intrusión basado en la red (NIDS). Además, hay IDS que también detectan movimientos buscando firmas particulares de amenazas bien conocidas.

Un IDS complementa, o es parte de, un sistema de seguridad más grande que también contiene firewalls, software antivirus, etc. Un NIDS intenta detectar actividad maliciosa como ataques de denegación de servicio, escaneos de puertos y ataques al monitorear el tráfico de la red.

Un sistema de detección de intrusiones basado en la red (NIDS) se utiliza para monitorear y analizar el tráfico de red para proteger un sistema de las amenazas basadas en la red.

Un NIDS lee todos los paquetes entrantes y busca cualquier patrón sospechoso. Cuando se descubren amenazas, en función de su gravedad, el sistema puede tomar medidas, como notificar a los administradores, o impedir que la dirección IP de origen acceda a la red [18].

2.6.1 Diferencia entre un HIDS y un NIDS

Las soluciones HIDS están instaladas en cada computadora de la red para analizar y monitorear el tráfico que entra y sale del nodo en cuestión. Un HIDS también rastrea y monitorea los cambios de archivos locales y posibles alteraciones debido a un acceso no autorizado y / o compromiso.

En contraste, un NIDS está estratégicamente posicionado en varios puntos de la red para monitorear el tráfico que entra y sale de los dispositivos de red. Las soluciones NIDS ofrecen sofisticadas capacidades de detección de intrusiones en tiempo real que a menudo consisten en un conjunto de piezas interoperables: un dispositivo autónomo, sensores de hardware y componentes de software son componentes típicos que conforman un NIDS. Estas piezas que trabajan en conjunto permiten una gama más amplia de capacidades de detección de intrusos de red que las soluciones HIDS.

Un modelo integral de seguridad de tecnologías de la información debe emplear tanto un HIDS como un NIDS, ya que cada uno tiene sus ventajas / desventajas. Por ejemplo, dado que una solución HIDS está instalada en el host y tiene acceso a detalles como la configuración del registro, los registros y otra información del sistema, son posibles capacidades forenses extendidas. Un inconveniente de esto, sin embargo, es que los recursos se extraen del host para alimentar el HIDS. Además, las soluciones HIDS son de naturaleza reactiva y solo pueden responder después de que ha ocurrido un ataque.

Por el contrario, un NIDS suele ser un dispositivo de hardware instalado en la red, por lo que no necesita acceder a ningún sistema host subyacente para obtener recursos. La instalación de estas soluciones también tiende a ser trivial; en la mayoría de los casos, simplemente se insertan en la red para comenzar a monitorear pasivamente el tráfico sospechoso. Aunque las soluciones NIDS suelen ser costosas y están dirigidas a la empresa, hay una selección decente de soluciones gratuitas de código abierto disponibles en el mercado. Estas soluciones utilizan software de código abierto y hardware básico, y ofrecen niveles comparables de seguridad y protección como ofertas comerciales de NIDS. Algunos los **IDS** que se comercializan actualmente y que más se utilizan son: Snort (NIDS) [19], Suricata (NIDS) [20], Bro(NIDS) [21], OSSEC (HIDS) [22] y Samhain (HIDS) [23].

2.6.2 NIDS basado en detección de firmas

Un NIDS puede incorporar uno de dos (o ambos) tipos de detección de intrusos en sus soluciones: basado en firmas o basado en anomalías. Un NIDS basado en firmas supervisa el tráfico de la red en busca de patrones sospechosos en paquetes de datos, "firmas" de patrones de intrusión de red conocidos, para detectar y remediar ataques y compromisos. Al utilizar una base de datos de tipos de intrusión bien conocidos y sus patrones de datos, un NIDS basado en firmas puede identificar rápidamente las intrusiones e iniciar un curso de acción apropiado.

Ventajas: Los detectores de firmas son muy efectivos en la detección de ataques sin que generen un número elevado de falsas alarmas. Pueden rápidamente y de forma precisa diagnosticar el uso de una herramienta o técnica de ataque específico. Esto puede ayudar a los encargados de la seguridad a priorizar medidas correctivas. Pueden permitir a los administradores de seguridad, sin importar su nivel o su experiencia en este campo, el seguir la pista de los problemas de seguridad de sus sistemas.

Desventajas: Solo detectan aquellos ataques que conocen, por lo que deben ser constantemente actualizados con firmas de nuevos ataques. Muchos detectores de abusos son diseñados para usar firmas muy ajustadas que les privan de detectar variantes de ataques comunes.

2.6.3 NIDS basado en detección de anomalías

Un NIDS basado en anomalías utiliza una línea base del sistema en un estado operacional normal para rastrear si hay actividad inusual o sospechosa. Aunque este método lleva tiempo establecerlo, ya que la línea base requiere que la solución "aprenda" sobre los patrones de uso del sistema, su enfoque heurístico orgánico para la detección de intrusos puede ser más flexible y poderoso que un enfoque basado en firmas, que requiere el patrón del tipo de intrusión preexistente está archivado para coincidir con el patrón. Las medidas y técnicas usadas en la detección de anomalías incluyen:

- *Detección de un umbral sobre ciertos atributos del comportamiento del usuario.* Tales atributos de comportamiento pueden incluir el número de ficheros accedidos por un usuario en un periodo de tiempo dado, el número de intentos fallidos para entrar en el sistema, la cantidad de CPU utilizada por un proceso, etc. Este nivel puede ser estático o heurístico.
- *Medidas estadísticas*, que pueden ser paramétricas, donde la distribución de los atributos perfilados se asume que encaja con un determinado patrón, o no paramétricas, donde la distribución de los atributos perfilados es aprendida de un conjunto de valores históricos, observados a lo largo del tiempo.
- Otras técnicas incluyen *redes neuronales, algoritmos genéticos y aprendizaje automático.*

Ventajas: Los IDSs basados en detección de anomalías detectan comportamientos inusuales. De esta forma tienen la capacidad de detectar ataques para los cuales no tienen un conocimiento específico. Los detectores de anomalías pueden producir información que puede ser utilizada para definir firmas en la detección de abusos.

Desventajas: La detección de anomalías produce un gran número de falsas alarmas debido a los comportamientos no predecibles de usuarios y redes. Requieren conjuntos de entrenamiento muy grandes para caracterizar los patrones de comportamiento normal.

Una vez se ha producido un análisis de los eventos y hemos detectado un ataque, el IDS reacciona. Las respuestas las podemos agrupar en dos tipos: **pasivas y activas.**

Las pasivas envían informes a personas, que se encargaran de tomar acciones al respecto, si procede. Las activas lanzan automáticamente respuestas a dichos ataques.

Respuestas pasivas: En este tipo de respuestas se notifica al responsable de seguridad de la organización, al usuario del sistema atacado o a algún CERT de lo sucedido.

También es posible avisar al administrador del sitio desde el cual se produjo el ataque avisándole de lo ocurrido, pero es posible que el atacante monitorice el correo electrónico de esa organización o que haya usado una IP falsa para su ataque.

Respuestas activas: Las respuestas activas son acciones automáticas que se toman cuando ciertos tipos de intrusiones son detectados. Podemos establecer dos categorías distintas:

- Recogida de información adicional: consiste en incrementar el nivel de sensibilidad de los sensores para obtener más pistas del posible ataque (por ejemplo, capturando todos los paquetes que vienen de la fuente que originó el ataque durante un cierto tiempo o para un máximo número de paquetes).
- Cambio del entorno: otra respuesta activa puede ser la de parar el ataque; por ejemplo, en el caso de una conexión TCP se puede cerrar la sesión establecida inyectando segmentos TCP al atacante y a la víctima o filtrar en el enrutador de acceso o en el firewall la dirección IP del intruso o el puerto atacado para evitar futuros ataques.

2.7. Estado del arte de la inteligencia artificial en la ciberseguridad

Como se ha comentado anteriormente este Trabajo Fin de Máster combina la ciberseguridad y se pretende analizar la viabilidad del uso de algoritmos de aprendizaje automático sobre las fuentes de datos.

Asegurar la información del ordenador y la red es importante para organizaciones e individuos porque la información comprometida puede causar considerable daño por esta razón fundamental los sistemas de detección de intrusos son importantes dentro de la seguridad informática. En la actualidad, se han propuesto diferentes enfoques de aprendizaje automático para mejorar el rendimiento de la detección de intrusiones por los sistemas. Este trabajo se centrará en la tarea de clasificación entre las conexiones de red normales y conexiones de ataques. Mencionaremos algunos trabajos que se basan en los algoritmos de aprendizaje automático y una breve descripción de los resultados que han obtenido, como por ejemplo pruebas utilizando algoritmos de aprendizaje supervisado junto al conjunto de datos NSL-KDD en los trabajos de [27] [28].

El método de Wang et al. [27] que tiene una tasa de exactitud del 99.92% utilizando el algoritmo de SVM, como antes se mencionó en la tabla 1, este supone un excelente rendimiento por parte del algoritmo pero también está fuertemente ligado a la calidad del conjunto de datos que se utilizarán para evaluar los modelos, en un intento para mejorar la calidad de los datos utilizaron el aumento de características usando densidad marginal logarítmica (LMDRT por sus siglas en inglés) para transformar los datos; Luego una comparación del rendimiento entre los algoritmos de aprendizaje SVM, RF y ELM por Ahmad et al.[28] dado que diferentes tipos de datos pueden llegar a mostrar mejor resultados si se ajustan mejor al método de aprendizaje empleado, en nuestro caso se trata de manejar grandes volúmenes de información, flujos de red, en cuestión de segundos, la carga que podría suponer el constante análisis en busca de amenazas puede ser abrumadora para el sistema, pero si se encuentra el método adecuado podría ser una herramienta de gran poder dentro de los sistemas informáticos. Su estudio de los tres algoritmos los llevó a apreciar el rendimiento superior de ELM con una exactitud del 99.5% usando el conjunto de datos completo, lo cual sería idóneo para manejar grandes cantidades de datos, pero SVM dio mejores resultados con una exactitud de 98.6% con sólo ¼ del tamaño del conjunto de datos, lo cual resultaría beneficioso si lo que se busca es analizar con la mayor exactitud y menor tiempo posible. Siguiendo con el uso de RF y el conjunto de datos NSL-KDD Farnaaz y Jabbar [29] propusieron un método para detectar cuatro tipos de ataques DOS, probe, U2R y R2L, con los porcentajes de precisión del 99.67%, 99.73%, 99% y 99% respectivamente cargando el conjunto de datos pre-procesado.

Algunos trabajos utilizando el conjunto de datos KDD-99 incluyen [30] [31] [32]. Con un método que combina RF y k-means ponderado, Aung y Min [30] proponen un modelo de detección de intrusos que demostró resultados con 99.8% de precisión. La

principal limitación del algoritmo de RF es que muchos árboles pueden hacer que el algoritmo sea lento para la predicción en tiempo real, pero es un algoritmo prometedor con altos porcentajes de precisión.

A. Aburomman con M. B. Ibne Reaz [31] utilizan el método de agregación utilizando la combinación de SVM-KNN adicionalmente optimizando los pesos que los algoritmos de clasificación utilizan con optimización por enjambre de partículas -**PSO** por sus siglas en Inglés, para analizar las trazas maliciosas en el conjunto de datos KDD-99. Los métodos de agregación/ensamblado han comenzado a ganar impulso y su uso en aplicaciones de sistemas para intrusiones también, entrenando 12 modelos en paralelo para luego combinar sus resultados por medio del enfoque de PSO obtuvieron una exactitud media del 92%.

Syarif y col. [33] aplicó en conjunto las tres técnicas de agregación embolsado, potenciamiento y apilamiento al problema de detección de intrusión para mejorar la precisión y reducir sus tasas de falsos positivos. Como clasificadores de base para estos métodos de conjunto se emplearon naïve Bayes, J48 (árbol de decisión), JRip (inducción de reglas) e iBK (vecino más cercano). Como resultado su enfoque logra una precisión de más del 99% en la detección de intrusiones conocidas, pero solo puede detectar nuevas intrusiones con tasas de precisión de alrededor del 60%. El uso de embolsado, potenciamiento y apilamiento no mostró una ganancia significativa en la precisión. Si bien el apilamiento fue el único método que condujo a una reducción significativa de las tasas de falsos positivos del 46,84%, también tiene el tiempo de ejecución más largo y, por lo tanto, es demasiado ineficiente para ser práctico para el problema de detección de intrusos. Consideremos otros esfuerzos que involucren el uso de PSO con el conjunto de datos NSL-KDD, motivados por el éxito del índice de Gini y el algoritmo de gradiente aumentando con árboles de decisión- GBDT en otros campos, crearon un método híbrido y novedoso para un sistema de intrusiones. El prototipo primero extrae un subconjunto óptimo de atributos del conjunto de datos entero por medio de índice Gini, luego el algoritmo GBDT está propiamente adoptado para detectar anomalías en las conexiones, utilizando PSO nos permite optimizar los parámetros del algoritmo de clasificación para potenciar su rendimiento total detectando amenazas generalizadas y aun mejor categorizar efectivamente cada tipo de ataque.

Como la mayoría de las intrusiones se pueden detectar examinando patrones de la conducta de las actividades que realizan los usuarios, muchos IDSs se han construido alrededor de este concepto utilizando el reconocimiento de los ataques por los patrones de uso o actividades sospechosas dentro de la red. Las RNA son una Buena opción en búsqueda de este tipo de patrones, un sistema de detección de anomalías basado en un perceptrón multicapa (**MLP**) usando la retropropagación para identificar el perfil normal de los usuarios fue propuesto por Ryan et al. [34]. Su modelo de MLP evalúa los comandos de los usuarios para identificar posibles intrusiones al final de cada sesión de registro, en su trabajo, cada vector de atributos describe parámetros de la conexión de un usuario determinado durante toda la sesión. Los 100 comandos frecuentemente

utilizados por los usuarios a lo largo de la sesión se utilizaron para determinar su comportamiento. Se creo un modelo MLP de 3 capas con dos capas ocultas y encontró que su modelo MLP fue capaz de identificar correctamente 22 casos de 24.

Subba el al [35] también utilizó una RNA de 3 capas en su modelo con atributos del conjunto de datos NSL-KDD, el cual alcanza una exactitud similar a la del algoritmo SVM de 95% con el mismo porcentaje de detección acertada, pero en menos tiempo de ejecución y con menos uso de los recursos computacionales por contar con una sola red oculta, estas ventajas lo colocan en las primeras opciones para analizar tráfico en tiempo real.

Dentro de la rama del aprendizaje profundo "Deep Learning" también se puede proponer modelos híbridos que combinen varios algoritmos de inteligencia artificial para potenciar su rendimiento y reducir la cuenta de falsos positivos. Tao Ma et al [36] haciendo uso del conjunto de datos KDD-99 aplicando su novedoso método que abreviado como SCDNN, ya que combina los algoritmos de agrupamiento espectral (SC) y redes neuronales profundas (DNN). El trabajo se divide en dos pasos, en el primero se crean sub-agrupaciones *k* con los atributos de las conexiones de red, estas se categorizan en los subconjuntos con características similares para ser alimentados a la red DNN en el segundo paso alcanzando una exactitud de hasta un 95 % durante las pruebas.

Con este trasfondo de resultados prometedores, en el uso de la inteligencia artificial para los IDS en este trabajo fin de máster se decidió por apostarle a las bondades de estos modelos. Se llevará a cabo una comparación entre tres algoritmos de inteligencia artificial, RF y GB que se encuentran dentro de la clasificación de agrupación/ensamblaje, y una red neuronal artificial multicapa.

3. Desarrollo de la solución

En este Trabajo Fin de Máster se ha relacionado el mundo de la ciberseguridad y el de aprendizaje automático. Se decidió usar algoritmos de aprendizaje automático y de inteligencia artificial en la detección de intrusiones en una red.

Intentamos demostrar el desempeño que pueden alcanzar los algoritmos seleccionados en la detección de intrusiones de una red. Por lo que contamos con los datos de conexiones de una red que se alimentarán a los algoritmos. Otra característica de este programa es la capacidad de modificación de los hiperparámetros de los algoritmos, ya que para encontrar la configuración idónea a este problema con los datos disponibles ha sido necesario realizar varias pruebas para poder justificar la elección de los distintos parámetros. Las diferentes pruebas realizadas también han servido para comparar el desempeño de la red neuronal contra los algoritmos de ensamblaje bajo distintas configuraciones.

En las secciones de este capítulo están descritos de forma detallada el conjunto de datos elegido para entrenar los algoritmos de aprendizaje automático, los tipos de algoritmos utilizados, la tecnología y software utilizado, así como las pruebas y resultados obtenidos.

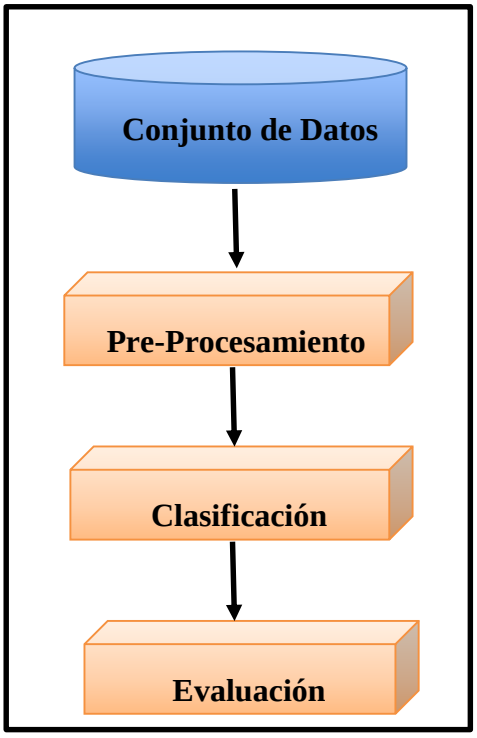

Figura 15.Secuencia de pasos para el método propuesto.

3.1. Elección del conjunto de datos

Para lograr los mejores resultados con algoritmos de aprendizaje supervisado es necesario darle la importancia al conjunto de datos que serán utilizados como la entrada de los algoritmos de aprendizaje. Estos suponen una parte clave para mejorar la correcta clasificación de nuevos flujos de red por lo que un conjunto de datos propiamente diseñado es imperativo para lleva a cabo esta tarea. Es importante contar con un conjunto de datos sobre distintas trazas de paquetes de red catalogados como ataque con su tipo de ataque y con trazas de tráfico normal con sus respectivas etiquetas. Ya que se contó con un conjunto de datos bastante extenso y propiamente etiquetado se hará uso del aprendizaje supervisado y por lo tanto necesitará la etiqueta que clasifique el tipo de dato, como se vio en la sección **2.1.1** de este documento.

A continuación, haremos mención de algunos conjuntos de datos antes utilizados por trabajos similares:

- **UNB ISCX 2012**: [37] conjunto de datos creados en el 2010 por la Universidad de Nuevo Brunswick (UNB – *University of New Brunswick*). La contribución relevante de este conjunto de datos es el uso de perfiles para el tráfico de red generado, el perfil se refiere al tráfico de los ataques y el perfil B al tráfico normal. El tráfico recolectado tiene una duración de 7 días utilizando una red de 17 ordenadores Windows XP y un ordenador Windows 7. Dentro de este conjunto de datos aparecen distintos ataques de **DoS**, **DDoS** y ataques de fuerza bruta **SSH**. Pero el uso de este representa desventajas por la corta duración de sus datos y el uso de sistemas operativos viejos (39).
- **CTU-13:** [38] base de datos realizada en el 2013 por la Universidad Técnica Checa de Praga (CTU – *Czech Technical University*). El tráfico de la red se divide en 13 capturas de diferentes botnets que contienen malware (llamados escenarios). Este tráfico puede dividirse en tres tipos: malicioso, normal y de fondo. El tráfico de fondo tiene esta etiqueta porque sus creados no pueden garantizar si es tráfico malicioso o no. Las principales desventajas de este conjunto de datos es la corta duración de las trazas que no permiten construir modelos que tomen en cuenta la evolución ciclo estacionaria de una red, es decir el comportamiento del tráfico durante el día/noche y durante días de semana/fines de semana, además de no detallar información sobre la estructura de la red donde se recolectaron estas trazas solamente se comparte que la información proviene de un enrutador situado dentro de la red de la universidad (40).
- **KDD-99:** [39] este conjunto de datos se obtuvo de una red real para una competencia, deriva del conjunto de datos DARPA-98 y consiste en 41 características y dos etiquetas, ataque o normal, dentro de los ataques se incluyen **DoS, U2R, R2L** y ataque de sondeo. El principal problema de esta base de datos reside en el hecho de que fue desarrollada en 1998 y, por lo

tanto, los datos son antiguos y no se ajustan con las características actuales de las redes.

- **UGR-16:** [40] De entre todas estas bases de datos, se ha seleccionado la base de datos UGR-16 para alimentar al algoritmo de aprendizaje automático que clasificará los flujos de red. Este conjunto de datos consta de 13 características con sus etiquetas debidamente separadas, incluye ataques de botnets, DoS, SSH y anomalías como spam. La elección de esta base de datos está motivada principalmente por su reciente creación y no está tan explotado como las bases de datos anteriores. La captura de paquetes está divida en archivos netflow y en archivos CSV, lo cual con el último formato se facilita la alimentación de las variables a los algoritmos de aprendizaje y supone una integración relativamente fácil con otras fuentes de datos dentro de **IDS** comerciales como Snort o Suricata.

3.2. Conjunto de datos UGR-16

La base de datos UGR-16 contiene aproximadamente 16,900M conexiones etiquetadas como fondo (los autores no están completamente seguros de que no contenga trazas de ataques) o ataque, que se generan intencionalmente para el experimento. Además de esta etiqueta, los archivos contienen 13 características por cada conexión. Es posible descargar los archivos completos o parcialmente de la página web oficial de la competición (https://nesg.ugr.es/nesg-ugr16/), los datos se dividen en archivos para la calibración y archivos de prueba. Este conjunto de datos ha sido recolectado de un ISP de nivel 3 situado en España por lo que está elaborado con tráfico de una red real y contiene ataques actualizados. El conjunto de datos de calibración tiene una duración de 4 meses desde Marzo hasta Junio del 2016 dentro de los archivos encontramos datos en formato «nfcapd» y «csv». Los archivos de formato «nfcap» contienen todas las características que un sistema IDS necesitará considerar mientras que el formato «csv» se utilizan las características comúnmente utilizadas por otros conjuntos de datos para su debido estudio. El conjunto de datos de prueba tiene una duración de 2 meses desde Julio hasta Agosto 2016 que contiene datos de ataques generados de manera sintética que corresponden con ataques actuales reconocidos.

En la figura notemos la topología de la red de la cual provienen las capturas, el ISP es un proveedor de servicios en la nube, muchos de los cuales son virtualizados. Por lo que su uso diario por parte de los usuarios nos garantiza que el tráfico que atraviesa la red es heterogéneo, dando como ventaja la representación de varios perfiles de usuarios que navegan diariamente en internet.

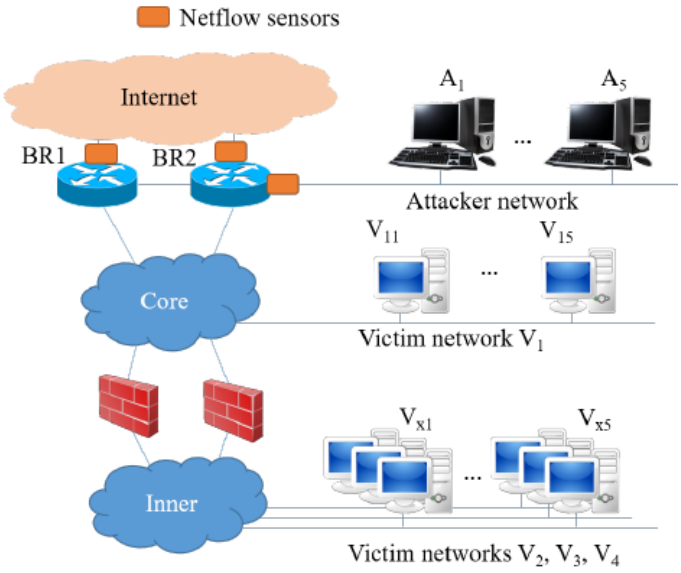

Figura 16.Topología de la red del ISP y maquinas víctimas colocadas [40].

La red está compuesta por 2 enrutadores de borde redundantes (BR1 y BR2) que proporcionan el acceso a Internet. En cada una de las interfaces de salida a internet, se han colocado sensores de flujo de red para la captura del tráfico de todas las conexiones entrantes y salientes. En esta porción superior de la red se implementa una "red atacante" con 5 máquinas que se los conoce como A1-A5. El ISP se divide en dos subredes diferentes:

1. Red Central: donde se encuentran los servicios que no están protegidos por firewall. Dentro de esta red se configuró 5 máquinas de víctimas que se usan solo para la colección de conjuntos de datos. Son colocados con otros clientes reales en una red existente que llamamos red de víctimas V1. Estas máquinas son nombradas como V11-V15.

2. Red Interna: donde se proporcionan servicios de firewall a los clientes. Dentro de esta red se colocan un total de 15 máquinas de víctimas adicionales donde se distribuyen en tres redes diferentes, previamente existentes, cada una consta de 5 máquinas. Se asignó la siguiente nomenclatura para las redes: red de víctimas V2 (máquinas V21-V25), red de víctimas V3 (máquinas V31-V35) y red de víctimas V4 (máquinas V41-V45).

En este trabajo fin de máster se utilizó la información contenida en los archivos de formato «csv» de los distintos meses durante los cuales se recolectaron ataques, una información más detallada de este uso se puede encontrar en la sección **3.6. Pruebas y resultados**. Dentro de este apartado se explicarán los diferentes ataques que hay en el conjunto de datos UGR-16 y las características sobre las conexiones que contiene.

3.3. Ataques analizados

Debido a que solamente se recoge tráfico de netflow, y, por tanto, no se considera el **payload** de la información en la traza, no se incluyen tipos de ataque susceptibles de ser detectados mediante análisis del payload. Solamente se consideran ataques relacionados con la red. Los ataques que se pueden encontrar en este conjunto de datos pueden ser divididos en dos categorías (que han sido explicadas en la sección **2.6** de este documento):

- **Basados en firmas**
  - **DoS**
  - **Botnets**
  - **Blacklist**
  - **Escaneo de puertos**
- **Basados en anomalías**
  - **Ataque de escaneo UDP**
  - **Ataque de escaneo SSH**
  - **Ataque de spam**

Describamos con más detalle cómo se componen los ataques del conjunto de datos, comenzaremos por los ataques DoS, para efectos de simplicidad y porque los ataques son muy similares entre sí, se les ha otorgado la etiqueta de *DOS* a todos estos:

DoS de baja tasa: Se envían paquetes TCP SYN a las víctimas utilizando la herramienta ***hping3***. El puerto destino es el 80, por lo que el tráfico se mezcla con el tráfico web de fondo real. Como puede comprobarse, la tasa de ataque es baja, por lo que no se afecta la operación normal de la red. Se consideran tres escenarios distintos de ataque:

- **DoS11:** Ataque DoS uno-a-uno, donde el atacante A1 ataca a la víctima V21. La duración total de DoS11 es de 3 minutos.
- **DoS53s:** Los cinco atacantes A1-A5 atacan a tres de las víctimas, cada una en una red diferente, durante 3 minutos. En particular, estos ataques siguen esta estructura: (A1, A2) ➏ V21, (A3, A4) ➏ V31 and A5 ➏ V41. La letra ‘s’ al final del nombre del ataque representa ‘síncrono’, lo que significa que los ataques se inician por todos los atacantes al mismo tiempo. Debido a esta sincronización, la duración de DoS53s es de 3 minutos también.
- **DoS53a:** Los ataques se ejecutan como en DoS53s, pero ahora cada víctima es seleccionada secuencialmente, siendo atacada durante 3 minutos con un periodo de inactividad de 30 segundos entre los tres ataques. De esta forma, la duración total de DoS53a es de 10 minutos. En este caso, la letra ‘a’ al final del nombre del ataque representa ‘asíncrono’.

Escaneo de Puertos: Se ejecuta un escaneo SYN continuo a los puertos comunes de las víctimas durante 3 minutos, utilizando la herramienta ***nmap*** [41]**.** Los ataques de escaneo de puertos que se encuentran en esta base de datos son:

- **Scan44:** Ataque de escaneo cuatro-a-cuatro, donde los atacantes A1, A2, A3 y A4 inician un escaneo al mismo tiempo a las víctimas V21, V11, V31 y V41, respectivamente.
- **Scan11:** Ataque de escaneo uno-a-uno, donde el atacante A1 escanea a la víctima V41.

Tráfico de una Botnet: Se simula tráfico de botnet mediante la exfiltración de datos desde algunas máquinas infectadas al puerto 80 de un ***botmaster***, el encargado de manejar las máquinas dentro de la red botnet, localizado en el ordenador A1. Se consideran veinte bots, correspondientes a todas las máquinas víctima. Cada uno de los bots lleva a cabo estas variantes de exfiltración:

- **Exf1KB:** Se envía un fragmento de información de 1KB al botmaster.
- **Exf1MB:** Se envía un total de 1MB de información al botmaster en una única conexión.
- **Exf1MBp:** El fragmento de información de 1MB a enviar al botmaster se divide en trozos de 1KB cada uno, y se envía al botmaster en conexiones distintas.

De nuevo, la transmisión desde cada bot puede ser síncrona (sufijo ‘s’), lo que significa que todos ellos inician la transmisión de información al mismo tiempo, o asíncrona (sufijo ‘a’), donde cada uno de ellos selecciona un instante aleatorio en una ventana de 3 minutos para empezar el correspondiente procedimiento de exfiltración.

En el **Anexo 1: Ataques UGR-16** se encuentra una tabla con los nombres de los ataques que aparecen en los archivos de esta base de datos junto con la categoría a la que pertenecen y el número de veces que aparece en el conjunto completo de datos.

El número de conexiones de cada categoría que aparece en cada archivo de la base de datos es el que se puede encontrar en la **Tabla 3** a continuación mostrada.

Característica	Calibración	Test
Comienzo de captura	10:47h 03/18/2016	13:38h 07/27/2016
Final de captura	18:27h 06/26/2016	09:27h 08/29/2016
Comienzo de ataques	N/A	00:00h 07/28/2016
Final de ataques	N/A	12:00h 08/09/2016
Número de Archivos	17	6
Tamaño (comprimido)	181GB	55GB
Número de Conexiones	≈ 13,000M	≈ 3,900M

Tabla 3.Características de los conjuntos de datos de UGR-16.

Dada la gran cantidad de conexiones en el conjunto de datos UGR-16 (236 GB), así como la diversidad de los ataques, en este Trabajo Fin de Máster se ha optado por centrar los esfuerzos del algoritmo en la detección de una sola categoría de ataque

versus la categoría de fondo. Si logramos mitigar un ataque de ciberseguridad con el uso de herramientas de inteligencia artificial, los beneficios serán evidentes dentro de una red en funcionamiento continuo si logramos la detección temprana de ciberataques.

La **figura 17** muestra la distribución de las conexiones en función del puerto asociados a los servicios más comúnmente utilizados por los usuarios dentro de una red, relacionado al conjunto de datos de calibración. Los servicios que generan mayor volumen tráfico son DNS, HTTP y HTTPS con los cuales dentro del conjunto de datos se notaran el uso de los protocolos TCP y UDP.

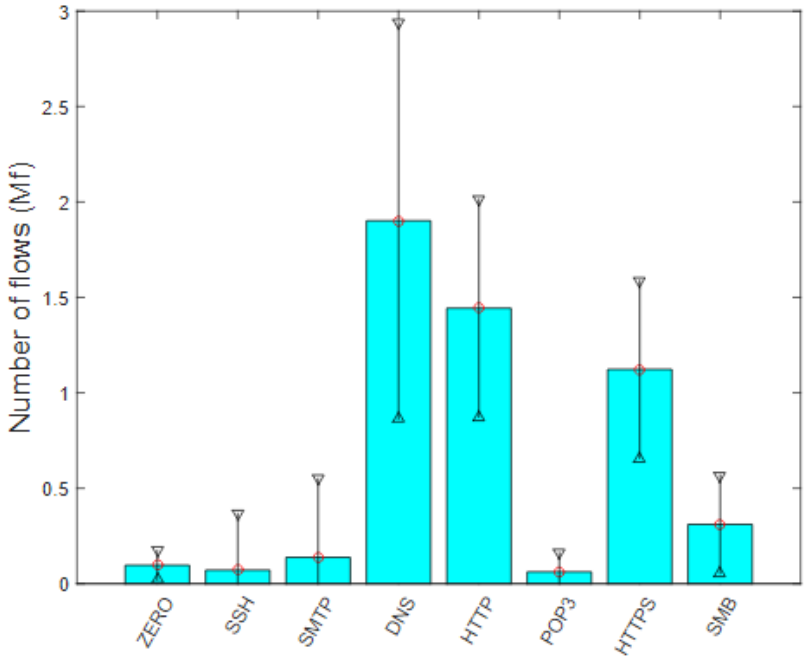

Figura 17.Distribución de las Conexiones en Función del Puerto.

3.4. Características de las conexiones

Como se ha comentado anteriormente en la **Tabla 3** el número de flujos total en el conjunto de datos es superior a 16,900 millones y cada conexión está compuesta de 13 características que corresponden a las características comúnmente utilizadas para el estudio de intrusiones y anomalías dentro de una red. En este conjunto de datos una conexión se considera como una secuencia de paquetes que comienzan y terminan en tiempos cuidadosamente documentados, además los datos fluyen entre una dirección **IP** de origen y una dirección **IP** destino bajo un protocolo de transporte, las direcciones IP han sido anonimizadas usando CryptoPan [42] anonimizarían conservando el prefijo.

Los 13 parámetros que definen una conexión están enumerados y descritos en la **Tabla 4**.

Nombre	Descripción	Tipo de variable
TimeStamp(te)	Estampa de tiempo de la finalización de la conexión	discreta
DurationFlow(td)	Número de segundos que dura la conexión	continua
SourceIP(sa)	Dirección IP de origen de la conexión	discreta
DestinationIP(da)	Dirección IP de destino de la conexión	discreta
SourcePort(sp)	Número de puerto donde se originó la conexión	continua
DestinationPort(dp)	Número de puerto destino de la conexión	continua
Protocol(pr)	Tipo de protocolo usado en la conexión	discreta
Flags(flg)	Banderas TCP usadas en la conexión	discreta
ForwardingStatus(fwd)	Estado del reenvió de paquetes en la conexión	continua
TypeService(stos)	El estado del tipo de servicio de la conexión	continua
PacketsExchangeFlow(pkt)	Número de paquetes intercambiados en la conexión	continua
Bytes(byt)	Conteo de bytes en el registro de la conexión	continua
Label	Clasificación de la conexión	discreta

Tabla 4. Características de la base de datos UGR-16 [40].

En el trabajo realizado las características fueron separadas según la división que se ha comentado. Se actuó de esta manera para poder realizar diferentes pruebas y comprobar que variables influyen más en el resultado final del algoritmo. Este hecho se explica de forma más detallada en la sección **3.12. Pruebas y resultados** de este documento.

3.5. Modelado de datos

Para utilizar la base de datos ***UGR-16*** en los algoritmos seleccionados (los cuales son explicados en el apartado **3.9**) es necesario un previo estudio y manipulación de los datos descargados. El objetivo es acomodar la información al algoritmo para asegurar una completa compatibilidad entre ambos y mejorar la capacidad de categorizar los ataques. Este proceso realizado se puede dividir en 3 fases: comprensión, preparación y modificación. En los próximos apartados se explicará el trabajo realizado en cada fase.

3.5.1 Fase 1: Comprensión

En esta primera fase se ha pretendido conocer mejor los datos de los cuales nos valdremos para tratar de solucionar el problema. Se averigua información acerca del número de registros, formato de los datos y estructura de estos, combinado con la información previa obtenida del conjunto de datos (apartado **3.2** de este documento) y los requerimientos de los algoritmos (apartado **3.9** de este documento), marcarán los pasos que se realizan en las siguientes fases.

Para explorar los datos hemos utilizando las librerías de NumPy y Pandas que nos permite manejarlos y visualizar en histogramas la distribución de las variables. La información obtenida en esta fase se ha recaudado de varios archivos del conjunto que están distribuidos en diferentes meses, los meses y sus correspondientes semanas que se descargaron están resumidos en la **Tabla 5** y **Tabla 6** («*Conjunto de Calibración*» *y* «*Conjunto de prueba*»).

Mes Descargado	Semana Descargada	Tipo de Tráfico
Marzo	Semana 3	1. Spam 2. SSH Scan 3. UDP Scan 4. Blacklist
Abril	Semana 2	1. Background 2. Spam 3. SSH Scan 4. UDP Scan 5. Blacklist
Mayo	Semana 1	1. Spam 2. SSH Scan 3. UDP Scan 4. Blacklist
Junio	Semana 4	1. Spam 2. SSH Scan 3. UDP Scan 4. Blacklist

Tabla 5. «Conjunto de Calibración» Número de registros de cada archivo de datos.

Mes Descargado	Semana Descargada	Tipo de Tráfico
Julio	Semana 5	1. Spam 2. SSH Scan 3. UDP Scan 4. Blacklist 5. Botnet 6. Scan 11 7. Scan 44 8. DOS
Agosto	Semana 1	1. Background 2. Spam 3. SSH Scan 4. UDP Scan 5. Blacklist 6. Botnet 7. Scan 11 8. Scan 44 9. DOS

Tabla 6. «Conjunto de prueba» Número de registros de cada archivo de datos.

Algunos ataques de reconocimiento que aparecen en el archivo «*Conjunto de prueba*» no aparecen en el resto de los archivos disponibles del «*Conjunto de Calibración»*, pero se han combinado para su evaluación.

3.5.2 Fase 2: Preparación

Los registros originales han sido reducidos ya que el conjunto en su totalidad es pesado, así que se tomarán instancias de los archivos para obtener las conexiones etiquetadas como fondo -al cual nos referiremos como *background* de ahora en adelante- y como ataques.

Se preparan varios ficheros de datos en formato «.csv» de manera manual utilizando Microsoft Excel, la separación de los datos en varios grupos se basa en la división del conjunto de datos en tres tipos de archivos, uno que contenga solamente los ataques de firmas, uno que contenga solamente ataques de anomalías y el último contendrá todos los ataques combinados para medir el desempeño de los algoritmos en cada conjunto de datos. Los ficheros obtenidos son los siguientes:

- **TrainDatasetSig.csv:** registros que provienen del archivo «*Conjunto de prueba*» y «*Conjunto de calibración*», con todas las características.
- **TestDatasetSig.csv:** registros que provienen del archivo «*Conjunto de prueba*», con todas las características.

- **TrainDatasetAno.csv:** registros que provienen del archivo «*Conjunto de calibración*», con todas las características (excepto las características descartadas).
- **TestDatasetAno.csv:** registros que provienen del archivo «*Conjunto de prueba*», con todas las características.
- **TrainDatasetAll.csv:** registros que provienen del archivo «*Conjunto de calibración*» y «*Conjunto de prueba*», con todas las características.
- **TestDatasetAll.csv:** registros que provienen del archivo «*Conjunto de prueba*», con todas las características.

3.5.3 Fase 3: Manipulación

Por último, se han modificado los datos obtenidos en la fase anterior para adaptarlos a los algoritmos de aprendizaje automático ya que solamente entienden datos en formato numérico y no en categorías como algunos de los atributos están compuestos, es necesario una transformación de los mismos.

Dicha transformación se llevará a cabo mediante la codificación de los distintos valores que puede tomar la variable ya que solamente codificar la variable con un número entero podría introducir un sesgo en la predicción del algoritmo de aprendizaje que beneficiaría a las variables cuyo número tras la codificación es mayor (por ejemplo, si el valor de la variable codificada con un número «2» tendría mayor peso comparado con el valor codificado con número «1»). Por lo tanto, se utilizará la librería de Pandas que incluye una función que transforma los valores con una codificación llamada *one-hot encoding*,

En la codificación *one-hot* representaremos los valores categóricos en vectores binarios, primero los mapeamos a valores enteros que posteriormente se transforma en un vector binario compuesto por todos 0 a excepción del valor entero del índice que es marcado con un número «1».

En nuestro caso, se creará una nueva característica por cada número de estados de la variable ha codificar, las características de conexión que necesitan ser codificadas son: *protocol type*, *flag* y la etiqueta que identifica la conexión como fondo o ataque. A modo de ejemplo, en la **Tabla 9** se muestra la codificación *one-hot* de la etiqueta identificativa. El vector que aparece sustituye la característica original, cada elemento del vector se corresponderá con una característica nueva.

Ataques de la etiqueta identificativa= Attack	**Codificación *one-hot***
Dos	[1,0]
Scan11	[1,0]
Scan44	[1,0]

Nerisbotnet	[1,0]
Blacklist	[1,0]
Anomaly-udpscan	[1,0]
Anomaly-sshscan	[1,0]
Anomaly-spam	[1,0]
Background	[0,1]

Tabla 7. Codificación one-hot de la etiqueta identificativa de la intrusión.

El resto de las codificaciones se pueden encontrar en el **Anexo 2: Codificaciones**.

Cabe mencionar que la modificación final, no contiene los 13 atributos iniciales, no son tomados en cuenta los siguientes atributos:

- *SourceIP(sa)* y *DestinationIP(da):* ya que este par de direcciones de origen/destino cambian dependiendo de la configuración de la red.
- *Flags(flg):* porque luego de las pruebas con los algoritmos se notó que no afecta mucho al desempeño de los algoritmos y estas no son fijas porque dependen de la conexión.
- *TimeStamp(te):* este atributo también es cambiante depende de cómo se genera el tráfico en la red.

3.6 Selección del algoritmo

Los algoritmos seleccionados en este Trabajo Fin de Máster para comprar el desempeño de los mismos al momento de clasificar datos de conexiones han sido el de RF, GB y RNA Multicapa. Estos algoritmos de aprendizaje automático se entrenarán con los datos obtenidos de la base de datos UGR-16, tras haberlos modelado de manera adecuada como se explica en la sección **3.5. Modelado de datos**. Tras el aprendizaje, para probar el desempeño de este algoritmo en la detección de intrusiones se hará uso de la misma base de datos, con los datos seleccionados de manera aleatoria para conformar el conjunto de datos de prueba.

Existen varios trabajos que comparan la exactitud de RF, GB y RNA cuando son presentados con un conjunto de características que describen los flujos de conexiones dentro de una red, pero hasta el momento no hay muchos trabajos que utilicen el conjunto de datos UGR'16 por su novedad y por contar con el beneficio de utilizar firmas de ataques y anomalías actuales.

El modelo creado con GB desarrollado en la modalidad de clasificador crea un modelo secuencial y aditivo de árboles de decisión que permite la optimización de la función de pérdida. Por defecto el modelo usa la función de desviación para la clasificación, una tasa de aprendizaje del 0.1 con 100 estados de optimización.

El modelo creado con el RF en la modalidad de clasificador toma el tamaño de los subconjuntos del tamaño de la entrada original y el número de árboles de decisión creados dentro del bosque por defecto es 10, la medida de pureza es Gini y la entropía es seleccionada para la ganancia de información. La profundidad del bosque dependerá

en qué medida se han expandido los árboles de decisión hasta llegar a hojas finales con alto número de pureza. Ayudamos a la predicción eligiendo balancear los pesos que son asignados a los resultados de cada clasificador individual.

La red multicapa desarrollada consta de tres capas (dos capas ocultas y una capa de salida) y se tratará de una red neuronal completamente conectada. Dicha red contará con un número de neuronas en cada capa en función de las características de entrada. La variación de neuronas en función de las características de entrada es debido a los diferentes experimentos que se han realizado con la red neuronal y que se explican en el apartado **3.9. Pruebas y resultados**. En cada neurona se multiplicará las entradas por los pesos correspondientes, posteriormente se sumará el sesgo. La salida de la red que se obtiene será la etiqueta que identifica la conexión y estará codificada mediante *one-hot*, por lo tanto, se obtendrá un vector unidimensional de 1 componente con la salida de la predicción binaria. En la **Figura 18** se muestra una representación del esquema de la red neuronal multicapa desarrollada y en el **Anexo 3. Descripción del modelo RNA** el resumen junto a la gráfica real del modelo.

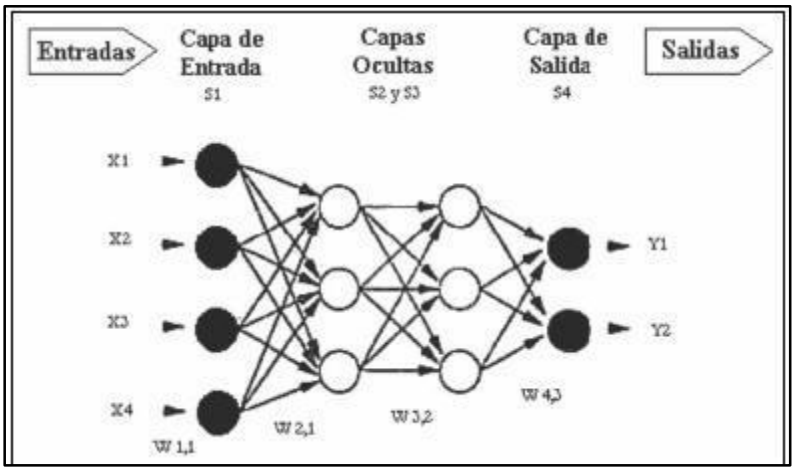

Figura 18. Red Neuronal Multicapa de 3 capas [43].

3.7 Tecnologías empleadas

Para realizar los modelos de inteligencia artificial comentados en el apartado anterior, se ha hecho uso del lenguaje de programación **Python**. Concretamente se ha utilizado la distribución de licencia libre de **Python** y **R** llamada Anaconda, creada por la empresa Continuum Analytics. Esta distribución es gratuita, instala varias versiones de **Python** a las que podemos acceder mediante el cambio del entorno de trabajo, ya que en cada ambiente puede ser personalizado al gusto del usuario. La facilidad de manejo y variedad en cuanto a los paquetes porque incluye hace que la distribución Anaconda sea la idónea para el desarrollo de este Trabajo Fin de Máster.

Existen otros lenguajes de programación que han sido usados para el desarrollo de redes neuronales artificiales en otros proyectos como Matlab, C++ y Java. Sin embargo,

dada la gran cantidad de documentación y librerías de acceso libre (como Scikit-learn y Keras, que se comentará posteriormente) desarrolladas en Python, se considera que **Python** es la mejor opción para el desarrollo del prototipo además de facilitar el desarrollo de los modelos al tratarse de un lenguaje sencillo a la hora programar.

La distribución de **Anaconda** cuenta **Jupyter**, una interface web interactiva similar a un "cuaderno" de trabajo. Las librerías que se utilizan en este trabajo son:

- **NumPy y Pandas:** Para el preprocesamiento y visualización de los datos respectivamente. Ambos son librerías de código abierto que brinda alto rendimiento en estructuras fáciles de usar.
- **Scikit-learn:** Para el desarrollo de los modelos de RF y GB. Es un paquete que proporciona versiones eficientes de una gran cantidad de algoritmos comunes. Scikit-Learn se caracteriza por una API limpia, uniforme y optimizada, así como por una documentación en línea muy útil y completa, que la convierte en la opción predilecta para el desarrollo de este trabajo fin de máster.
- **Keras:** Para el desarrollo de la red neuronal multicapa. La biblioteca Keras utiliza en su motor de procesamiento TensorFlow que es una biblioteca de código abierto creada por Google Brain Team, proyecto de investigación de la compañía Google. Dicha biblioteca ofrece todas las herramientas necesarias para construir, entrenar y comprobar la eficacia de redes neuronales artificiales. Cabe destacar que TensorFlow incorpora un conjunto de herramientas de visualización llamada TensorBoard muy útil, desafortunadamente algunas opciones de visualización no se encuentran disponibles aun por medio de la API de Keras, como por ejemplo los histogramas.

Existen un gran número de alternativas en cuanto a librerías **Python** que se pueden usar para realizar este proyecto, sin embargo, se ha usado la librería **Keras** por las múltiples ventajas que ofrece. Keras es una librería de muy alto nivel que funciona encima de Theano o Tensorflow, además enfatiza el minimalismo, puedes hacer una red neuronal con muy poquitas líneas de código. Estas características la convierten en una excelente opción para comenzar a crear modelos con conceptos de **Deep Learning**, en este trabajo fin de máster se utilizará Tensorflow como el motor principal de la librería de Keras.

Algunas de las librerías de **Python** alternativas a Keras que pueden ser usadas para el desarrollo de redes neuronales artificiales son:

- **Theano:** librería de bajo nivel especializada en optimización de computación numérica. Permite el cálculo de gradientes de funciones de forma automática, lo que junto con su interfaz en Python y su integración con Numpy, convirtió esta librería en sus inicios en una de las más usadas para Deep Learning de propósito general.
- **Lasagne:** librería ligera para poder construir y entrenar redes neuronales que trabaja encima de Theano. Trata de abstraer un poco los complejos cálculos

que hay por debajo de los algoritmos de Deep Learning y proporcionar una interfaz más amigable.

- **TensorFlow:** Cuando una red neuronal se vuelve más compleja, llena de capas y parámetros, el proceso se vuelve mucho más complejo. Es aquí donde entra TensorFlow, una librería de computación numérica que computa gradientes automáticamente, esto quiere decir que está a un nivel más bajo y profundo que Keras o Pytorch.
- **Pytorch:** es una estructura de Python que permite el crecimiento rápido del Deep Learning, con una fuerte aceleración de la GPU. La característica principal de Pytorch es que utiliza grafos computacionales dinámicos.

Toda la programación de los algoritmos de inteligencia artificial es desarrollada dentro de un entorno de trabajo en Anaconda versión 5.1.0, cuaderno Jupyter versión 5.4.0 junto con la versión **3.6.4** de **Python**. En dicho ambiente se ha instalado TensorFlow 1.8.0 (versión publicada en Abril del 2018). El código utilizado para el desarrollo de la red neuronal en Keras se basa en el código compartido por el usuario *Buomsoo-Kim* en la plataforma de desarrollo colaborativo GitHub. Dicho código ha sido modificado y adaptado a las necesidades de este proyecto ya que el código original estaba destinado al desarrollo de redes neuronales que clasifican dígitos y que hacen uso de una base de datos de dígitos manuscritos llamada MNIST.

3.8 Evaluación

Dado que las predicciones de los algoritmos son binarias por ser entregadas en dos valores background «0» y attack «1», introduciremos las medidas básicas de rendimiento derivadas de la matriz de confusión. La matriz de confusión es una tabla de dos por dos que contiene cuatro resultados producidos por un clasificador binario, esto nos ayuda a medir cómo se comporta nuestro modelo. Varias medidas, como la exactitud, la precisión, la sensibilidad y la medida-F, se derivan de la matriz de confusión [44].

Para la clase dada, la cantidad de instancias correctamente clasificadas como maliciosas se conoce como verdaderos positivos -True Positive- (TP). El número de instancias clasificadas como maliciosas, pero que deben ser normales y, por lo tanto, rechazadas, se conoce como falso positivo -False Positive- (FP). La cantidad de instancias clasificadas como normales, pero que en realidad son maliciosas se conoce como falso negativo -False Negative-(FN). El número de instancias normales de una clase rechazada correctamente se conoce como verdadero negativo -True Negative- (TN).

		Predicción	
		Positivos	Negativos
Observación	Positivos	Verdadero Positivo (TP)	Falso Negativo (FN)
	Negativos	Falso Positivo (FP)	Verdadero Negativo (TN)

Tabla 8. Características de los conjuntos de datos de UGR-16.

La exactitud de los algoritmos de inteligencia artificial se mide por la exactitud general, indicada como:

$$Exactitud = \frac{TP + TN}{(TP + TN + FP + FN)}$$

- ***Precisión***: La fracción de una categoría que consiste en objetos de una clase especificada. La precisión de la predicción de la clase *i* con respecto a la clase *j* es la precisión *(i, j)* = *Pij*. Está dado por:

$$Precisión(i,j) = \frac{TP}{(TP + FP)}$$

- ***Sensibilidad*** (o tasa de TP): El grado en que una categoría contiene todos los objetos de una clase especificada. El retiro de la clase *i* con respecto a la categoría *j* es, *Sensibilidad (i, j)* = m_{ij}/m_j, donde m_j es el número de objetos en la categoría *j*. Está dado por:

$$Sensibilidad(i,j) = \frac{TP}{(TP + FN)}$$

- ***Medida-F***: Una combinación de ambos precisión y sensibilidad que mide el grado en que una categoría contiene solamente instancias de una clase en particular y todas las instancias de esa clase, el balance entre la precisión y la sensibilidad al ser menor a 1 la medida se inclina a darle importancia a la precisión, un valor mayor a 1 supone una importancia por la sensibilidad, mientras un valor de 1 significaría un balance perfecto entre ambas. La medida-F de la clase *i* con respecto a la categoría *j* es:

$$F(i,j) = 2 * \frac{(Precisión(i,j) * Sensibilidad(i,j))}{(Precisión(i,j) + Sensibilidad(i,j))}$$

Con todos estos parámetros, estamos listos para comparar la validez de los resultados y los criterios de los tres algoritmos diferentes para clasificar el mismo conjunto de datos según sus propios métodos.

3.9 Pruebas y resultados

Los datos que alimentan nuestros tres algoritmos de inteligencia artificial son los archivos «.csv» explicados en el apartado **3.5.2. Fase 2: obtención**. Los experimentos se dividirán en 3 grupos de datos en función del conjunto de ataques contenido en cada conjunto de datos (conjuntos descritos en el apartado **3.4. Características de las conexiones)**; Para tener un conjunto de datos representativo del tráfico real de Internet, la proporción de flujos benignos se mantuvo más alta que la cantidad de flujos malignos. Utilizamos los algoritmos de agrupamiento aumento de gradiente, bosque aleatorio y red neuronal multicapa, que se comparan más adelante en términos de rendimiento y precisión; por cada uno de los conjuntos de datos se realizarán 5 pruebas en las cuales se modifican diversos hiperparámetros de los algoritmos de aprendizaje que clasificaron los datos (dichos datos están en los archivos «.csv» cuyos nombres siguen la nomenclatura test/traindataset--.csv). Al finalizar con todas pruebas de variación de parámetros de cada conjunto de datos se elegirá la configuración de los algoritmos que ha dado el mejor resultado en las pruebas para representar las gráficas y métricas de evaluación que luego se usarán para comparar el rendimiento final de los tres algoritmos de aprendizaje automático.

Todas las pruebas realizadas se han sido desarrollado con el ordenador cuyo procesador es de la marca Intel Core, el modelo i5-2430MU. Dicho procesador cuenta con 2 núcleos que trabajan a una frecuencia de 2,40 GHz y tiene la capacidad para contar con 4 subprocesos. Además, el ordenador cuenta con 4 GB de memoria **RAM** de la cual 3,88 GB es utilizable.

3.9.1 Experimento 1: conjunto de datos de firmas

Como se mencionó anteriormente los subconjuntos de datos armados contienen más porcentaje de datos benignos que malignos, en nuestro caso los que comprenden la etiqueta de «attack». A pesar de que este desbalance de clases convierte en un reto para el algoritmo la tarea de clasificar las instancias, esta proporción es representativa de un escenario real donde la mayoría del tráfico que cursa una red es benigno. El conjunto de datos firmas «TrainDatasetSig.csv» contiene solamente los ataques que caen dentro de la categoría de firmas (blacklist, Dos, Scan, botnets). Un total de 12,370 instancias son utilizadas para el conjunto de entrenamiento, donde el 74% conforma las instancias de fondo «background» y el 26% las instancias de ataque «attack», para el conjunto de prueba el número total de instancias es de 8,014, con 65% de instancias de fondo y el 35% instancias de ataque. El detalle de las instancias agregadas a este conjunto de datos por categoría está representado en la **Tabla 9** y la **Tabla 10** que se encuentran a continuación:

Conjunto de Datos Firmas de Entrenamiento		
Categoría	**Cantidad**	
Background	9,160	
Blacklist	738	
Scan44	700	
Dos	700	
Scan11	700	
Nerisbotnet	372	
Total de Instancias	12,370	
Etiqueta	**Etiqueta Binaria**	**Cantidad**
Background	0	9,160
Attack	1	3,210

Tabla 9. Características del conjunto de datos de firmas para entrenamiento.

Conjunto de Datos Firmas de Prueba		
Categoría	**Cantidad**	
Background	5,182	
Blacklist	610	
Scan44	600	
Dos	600	
Scan11	600	
Nerisbotnet	422	
Total de Instancias	8,014	
Etiqueta	**Etiqueta Binaria**	**Cantidad**
Background	0	5,182
Attack	1	2,832

Tabla 10. Características del conjunto de datos de firmas para prueba.

3.9.1.1 Aumento de gradiente (gradient boosting)

El modelo creado con **GB** probado en la modalidad de clasificador, como se ha descrito en capítulos anteriores, crea un modelo secuencial y aditivo de árboles de decisión algunos de los hiperparámetros serán conservados con los valores por defecto, el resto de las modificaciones por corrida del algoritmo se detallan en la **Tabla 11**, con sus respectivos resultados. Los parámetros utilizados para estos experimentos se detallan a continuación:

1. **«*learning_rate*»:** tasa de aprendizaje del clasificador seleccionado que reduce la contribución de cada árbol según la tasa de aprendizaje. Dicha tasa cambia durante los experimentos.
2. **«*n_estimators*»:** El número de etapas de potenciamiento a realizar, GB es bastante robusto contra el sobre ajuste del modelo, por lo que un número grande de estimadores generalmente da como resultado un mejor rendimiento. Dicho valor cambia durante los experimentos.
3. **«*max_features*»:** La cantidad de características a considerar cuando se busca la mejor división de los datos, usaremos el valor en decimal que nos permite indicarle al algoritmo que deseamos usar un porcentaje de los datos. Este parámetro se fija en el valor "0.8".
4. **«*min_samples_split*»:** Mínimo de muestras requeridas para dividir un nodo interno. Se fija en el valor "4".
5. **«*min_samples_leaf*»:** Mínimo de muestras requeridas por cada nodo hoja. Se fija en el valor "2".
6. **«*loss*»:** La función de pérdida a optimizar por el algoritmo. Se fija en el valor por defecto "deviance" = clasificación para regresión logística.

No. de Prueba	learning_rate	n_estimators	max_features	min_samples_split	Exactitud Datos Entrenamiento	Exactitud Datos Prueba
1	0.01	200	0.8	4	90.1%	87.1%
2	0.1	200	0.8	4	94%	92.7%
3	0.2	300	0.8	4	96.8%	94.4%
4	0.3	400	0.8	4	97.3%	94.4%
5	0.2	500	0.8	4	96.9%	94.1%

Tabla 11. Resultados de GB para el experimento 1.

Como medida de evaluación utilizaremos el reporte de clasificación de la librería «metrics» de sklearn, que nos provee las principales métricas de comparación del desempeño de los algoritmos de aprendizaje automático. El mejor resultado con el conjunto de datos de firmas de entrenamiento se ha producido con los parámetros de la **Prueba 4,** las métricas de evaluación para el conjunto de datos de entrenamiento se muestran en la **Figura 19**. Tenemos una precisión del 97% durante el entrenamiento con una sensibilidad del 0.97 recuperando el 97% de las trazas malignas del conjunto completo de datos conformado por los ataques de firmas. El balance entre la precisión y sensibilidad nos muestra un valor de 0.97 con una leve preferencia por la precisión.

	precision	recall	f1-score	support
0	0.96	0.99	0.98	1778
1	0.98	0.90	0.94	696
avg / total	0.97	0.97	0.97	2474

Figura 19. Reporte de Clasificación de GB con el mejor resultado de exactitud para datos de entrenamiento en el experimento 1.

Dado que las predicciones de los algoritmos son binarias por ser entregadas en dos valores background «0» y attack «1», la matriz de confusión de la **Figura 20** muestra la cantidad de 629 instancias correctamente clasificadas como malignas (TP), al igual que un bajo número de instancias mal ubicadas, como malignas cuando deberían ser de fondo (FP) y clasificadas como de fondo cuando deberían ser malignas (FN). Finalmente, GB ha clasificado 1767 instancias de manera correcta con la etiqueta de fondo (TN).

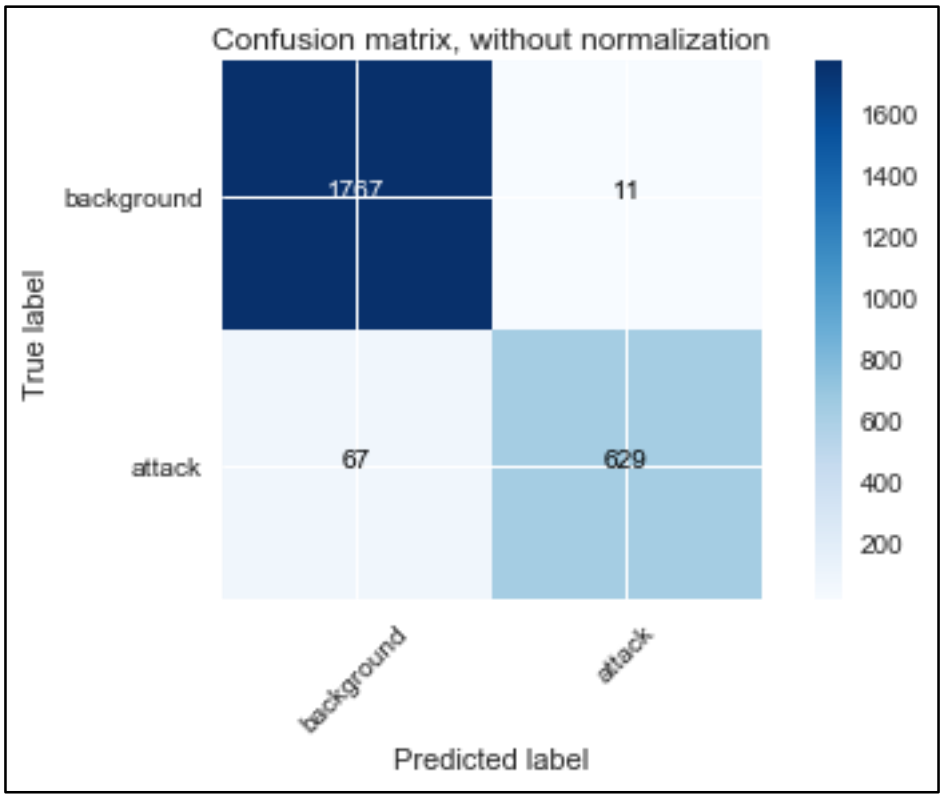

Figura 20. Matriz de Confusión de GB con el mejor resultado de exactitud en el conjunto de datos de entrenamiento para el experimento 1.

Las métricas de evaluación con el mejor resultado son producto de la **Prueba 4** para el conjunto de datos de prueba se muestran en la **Figura 21**, se logró una precisión del 93% al momento de predecir correctamente las instancias que caen bajo la etiqueta de ataque, la composición de las instancias clasificadas como un ataque se aprecian mejor más adelante en la matriz de confusión. De esas instancias clasificadas como ataque se

logró una sensibilidad del 0.94 recuperando el 94% de instancias malignas del conjunto total de ataques.

	precision	recall	f1-score	support
0	0.93	0.98	0.96	5182
1	0.97	0.87	0.92	2832
avg / total	0.95	0.94	0.94	8014

Figura 21. Reporte de Clasificación de GB con el mejor resultado de exactitud para datos de prueba en el experimento 1.

La matriz de confusión de la **Figura 22** está relacionada a las medidas de evaluación involucradas con el conjunto de datos de prueba, la cantidad de instancias correctamente clasificadas como malignas (TP) es de 2465 con un número de instancias mal ubicadas como de fondo cuando deberían ser malignas (FN) más alto que la cantidad clasificada como malignas cuando deberían ser de fondo (FP). Finalmente, GB ha clasificado 5102 instancias de manera correcta con la etiqueta de fondo (TN).

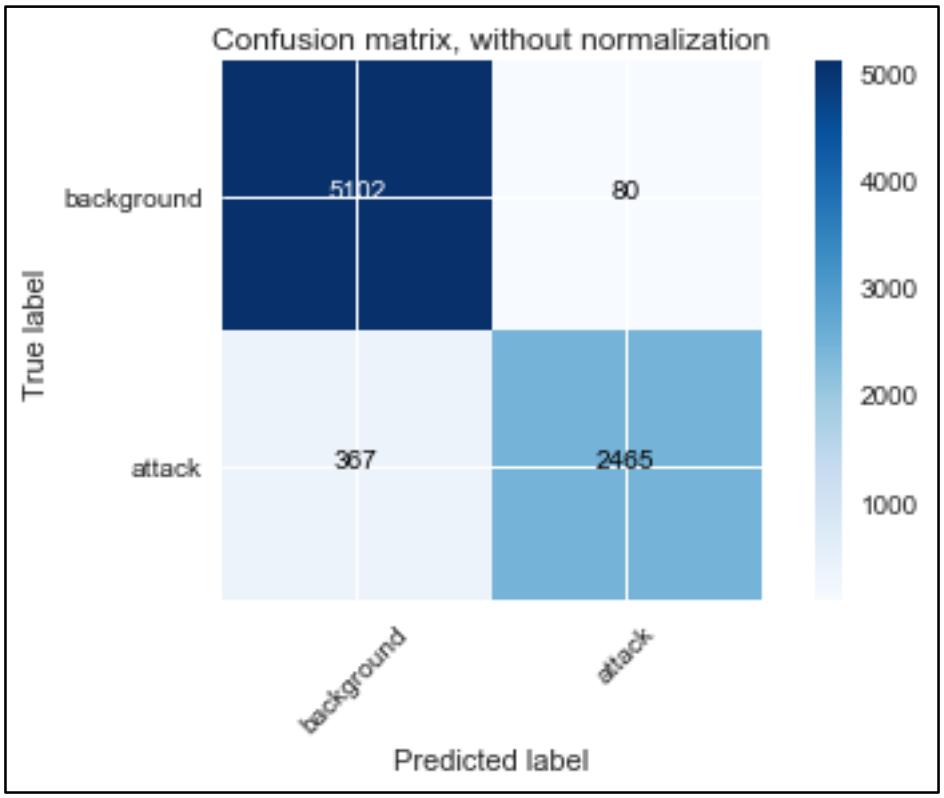

Figura 22. Matriz de Confusión de GB con el mejor resultado de exactitud en el conjunto de datos de prueba para el experimento 1.

3.9.1.2 Bosque aleatorio (random forest)

El modelo del clasificador RF crea varios árboles de decisión agrupados dentro de un bosque, de manera individual la predicción de cada árbol es tomada en cuenta para mejorar la precisión de la predicción general. En la **Tabla 12** se muestran los hiperparámetros utilizados para correr el algoritmo junto a sus resultados. Los parámetros utilizados para estos experimentos se detallan a continuación:

1. **«*n_estimators*»:** El número de árboles de decisión dentro del bosque. Dicho valor será modificado durante los experimentos.
2. **«*criterion*»:** La función para medir la calidad de una división. Los criterios soportados son "gini" para la impureza de Gini y "entropía" para la ganancia de información. Se fija en el valor por defecto "gini" para dividir los nodos basándose en la pureza de las clases.
3. **«*max_features*»:** La cantidad de características a considerar cuando se busca la mejor división de los datos. Se fija en el valor por defecto "auto" que calcula la cantidad como raíz cuadrada (n_atributos).
4. **«*min_samples_split*»:** Mínimo de muestras requeridas para dividir un nodo interno. Se fija en el valor "4".
5. **«*min_samples_leaf*»:** Mínimo de muestras requeridas por cada nodo hoja. Se fija en el valor "2".
6. **«*max_depth*»:** La máxima profundidad del árbol. Si es utilizado con el valor por defecto, los nodos se expanden hasta que todas las hojas son puras o hasta que todas las hojas contengan menos que el valor "min_samples_split". Se fija en el valor por defecto "None".

No. de Prueba	n_estimators	min_samples_split	Min_samples_leaf	Exactitud Datos Entrenamiento	Exactitud Datos Prueba
1	200	4	2	97.2%	94.6%
2	100	4	2	97.3%	92.3%
3	300	4	2	97.1%	94.3%
4	400	4	2	97.2%	91.6%
5	500	4	2	97.3%	94.4%

Tabla 12. Resultados de RF para el experimento 1

La variación en los resultados para los **Prueba 1, 3** y **5** con el conjunto de datos de anomalías de entrenamiento es mínima cuando comparamos la salida de las corridas del algoritmo. Notamos en la **Figura 23** que estos experimentos han producido el mismo valor de 0.97 para la precisión, la sensibilidad y medida-F durante el entrenamiento del algoritmo.

	precision	recall	f1-score	support
0	0.97	1.00	0.98	1778
1	0.99	0.91	0.95	696
avg / total	0.97	0.97	0.97	2474

Figura 23. Reporte de Clasificación de RF con el mejor resultado de exactitud para datos de entrenamiento en el experimento 1.

La matriz de confusión de la **Figura 24** muestra la cantidad de 634 instancias correctamente clasificadas como malignas (TP) junto a un número de instancias mal ubicadas notablemente bajo. Con este conjunto de datos de entrenamiento el algoritmo RF ha clasificado 1771 instancias de manera correcta con la etiqueta de fondo «0» (TN).

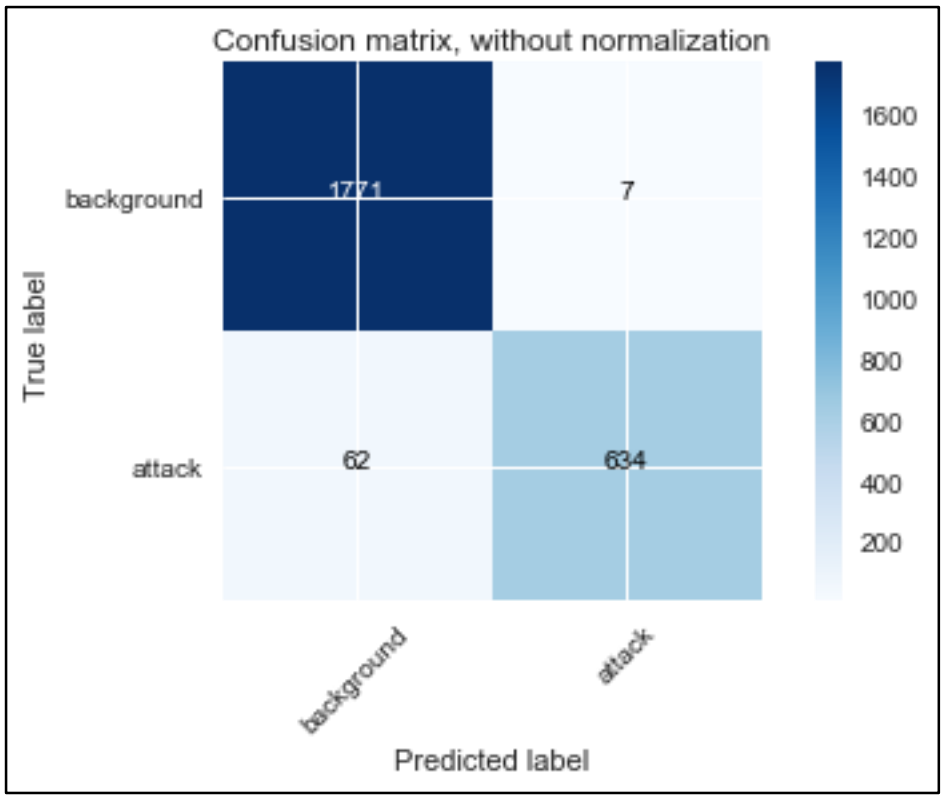

Figura 24. Matriz de Confusión de RF con el mejor resultado de exactitud en el conjunto de datos de entrenamiento para el experimento 1.

En la **Figura 25** se puede evaluar el desempeño de RF con las instancias que aún no ha analizado con el conjunto de datos de prueba, logró una precisión y sensibilidad del 95% al momento de predecir correctamente las instancias que caen bajo la etiqueta de ataque.

	precision	recall	f1-score	support
0	0.99	0.93	0.96	5502
1	0.87	0.98	0.92	2512
avg / total	0.95	0.95	0.95	8014

Figura 25. Reporte de Clasificación de RF con el mejor resultado de exactitud para datos de prueba en el experimento 1.

La matriz de confusión de la **Figura 26** representa al modelo creado con el conjunto de datos de prueba, la cantidad de instancias correctamente clasificadas como malignas (TP) es de 2456 con un número de instancias mal ubicadas como de fondo cuando deberían ser malignas (FN) más bajo que la cantidad clasificada como malignas cuando deberían ser de fondo (FP).

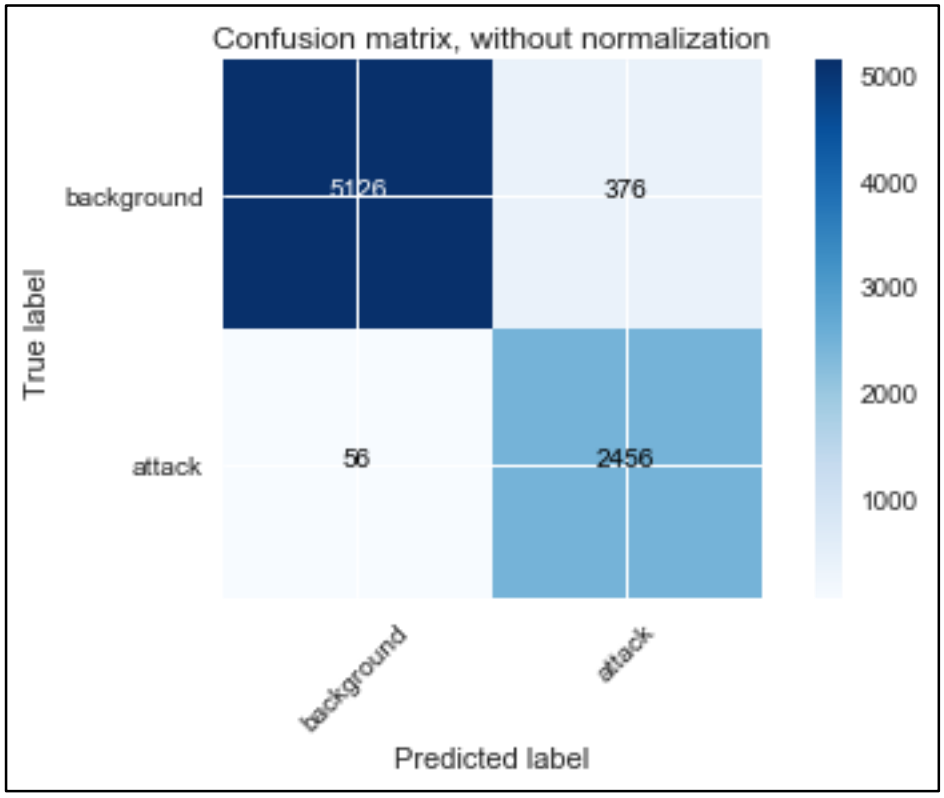

Figura 26. Matriz de Confusión de RF con el mejor resultado de exactitud en el conjunto de datos de prueba para el experimento 1.

3.9.1.3 Red neuronal artificial multicapa

Es un algoritmo supervisado que aprende una función $f(.): R^m \rightarrow R^o$ entrenando con el conjunto de datos alimentado, donde m es el número de dimensiones de entrada para la red neuronal y O es el número de dimensiones de salida como resultado. Dado un conjunto de características X = $\{x_1, x_2, \ldots, x_m\}$ y una etiqueta como objetivo, puede aprender un aproximador de función no lineal para la clasificación. Las diferentes funciones de activación y optimizadores usados en estos experimentos se encuentran descritos de forma detallada en el apartado **2.4 Redes neuronales artificiales** de este documento. Los resultados se resumen en la **Tabla 13**.

El modelo de red neuronal artificial multicapa es un modelo de 3 capas (2 ocultas y una de salida) que se encuentra completamente conectadas. Los parámetros del algoritmo que son modificados durante las pruebas son los siguientes:

- **«*learning_rate*»:** tasa de aprendizaje del optimizador seleccionado. Dicha tasa se fija en 0.001.
- **«*epochs*»:** número de veces que el modelo es expuesto al conjunto de datos de entrenamiento. El número de épocas se fija en 200.
- **«*batch_size*»:** número de ejemplos que contendrá cada grupo de datos en los cuales se dividirán los datos de entrada para ser mostrados al modelo antes de ajustar los pesos; con los grupos de datos obtenidos se operará en cada época de entrenamiento. El «*batch_size*» se fija en 10.
- **«*activation*»:** Función de activación para las capas de la red neuronal. Dicho valor se modificará durante las pruebas.
- **«*optimizer*»:** El optimizador es la técnica de búsqueda utilizada para actualizar los pesos de las capas. Dicho valor se modificará durante las pruebas.
- **«*loss*»:**la evaluación del modelo utilizado por el optimizador para navegar por el espacio de peso. Dicho valor se fija en "Binary_crossentropy" para pérdida logarítmica binaria por tener dos valores binarios a categorizar «1» y «0».

No. de Prueba	Función de activación capas ocultas	Función de activación capa de salida	Optimizador	Loss	Datos Entrenamiento	Datos Prueba
1	Relu	Sigmoid	SGD	Binary_crossentropy	87%	83%
2	Tanh	Relu	Adam	Binary_crossentropy	72%	64%
3	Relu	Tanh	SGD	Binary_crossentropy	28%	35%
4	Sigmoid	Sigmoid	SGD	Binary_crossentropy	72%	64%
5	Relu	Sigmoid	Adam	Binary_crossentropy	94%	74%

Tabla 13. Resultados de RNA para el experimento 1.

Durante el entrenamiento, la primera cantidad que es útil observar de cerca al crear los modelos RNA es la pérdida, ya que evalúa en cada época los grupos de datos individuales. El objetivo principal del modelo es minimizar la pérdida con respecto a los parámetros ajustando los pesos por medio del algoritmo de optimización. El valor de pérdida implica qué tan bien o mal se comporta el modelo después de cada iteración de entrenamiento. La **figura 27** muestra cómo la función de pérdida comienza a reducir su valor significativamente luego de la época 70 y continúa bajando hasta alcanzar un valor de 0.28.

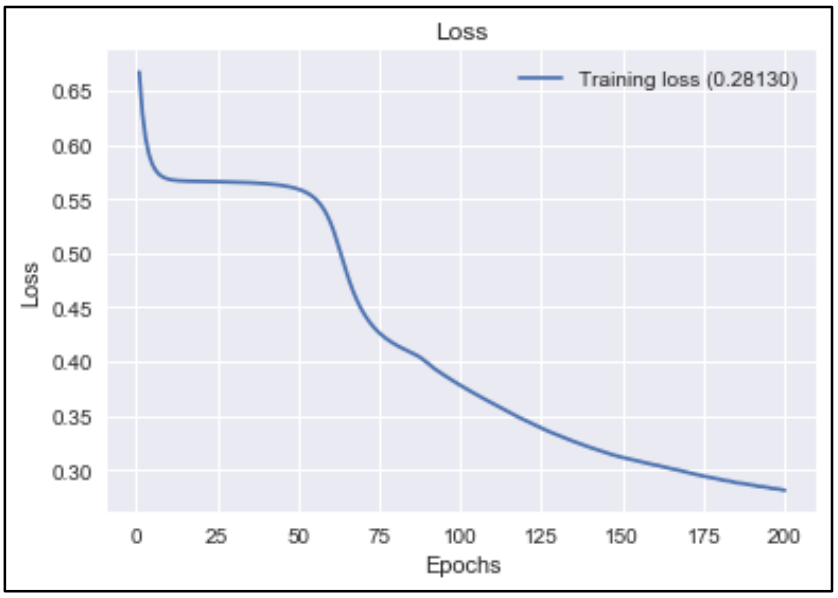

Figura 27. Valores de la función de pérdida durante épocas del modelo RNA para experimento 1.

La **figura 28** muestra como varía la exactitud del modelo RNA después de cada iteración, a partir de la época numero 98 notamos como la red neuronal empieza a aprender con el conjunto de datos de ataques de firmas. Su aprendizaje fue algo tardío, pero logró alcanzar una exactitud del 88% para el conjunto de entrenamiento.

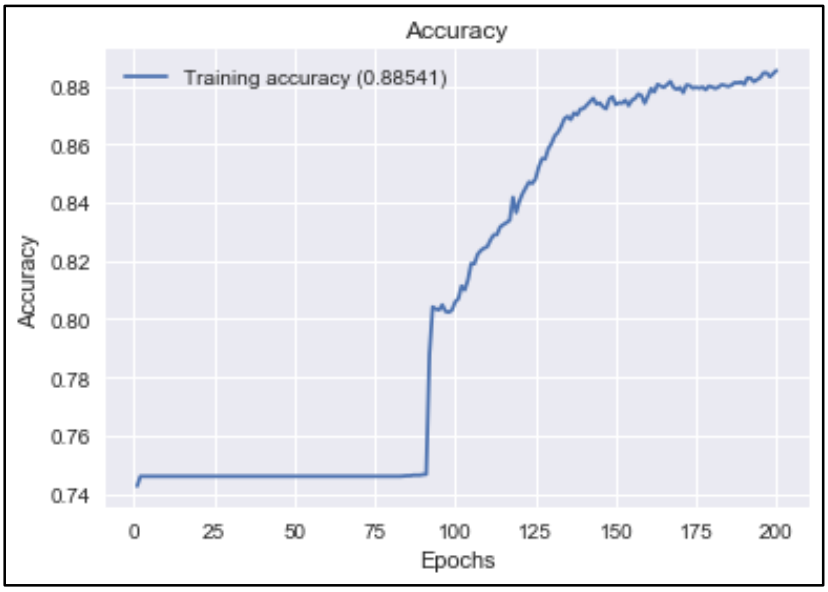

Figura 28. Valores de la exactitud durante épocas del modelo RNA para experimento 1.

Los resultados para las pruebas de la RNA son bastante variados, la exactitud del modelo está fuertemente ligada a la selección de los parámetros que mejor se ajusten al tipo de datos. La **Prueba 1** brinda los mejores resultados en cuanto a precisión, a pesar de no alcanzar el porcentaje de entrenamiento más alto del conjunto de entrenamiento, ha logrado un 87%, si logra un mejor rendimiento al ser presentado con los nuevos datos de prueba con un 83% de exactitud. Notamos en la **Figura 29** que utilizando el optimizador "SGD" y la función "sigmoid" para la capa de salida, esta corrida ha producido el valor de 0.95 para la sensibilidad del modelo al momento de recuperar las instancias de fondo, pero no ha logrado actuar tan bien recuperando las instancias de ataque, logrando una sensibilidad del 0.69.

	precision	recall	f1-score	support
0	0.89	0.95	0.92	1778
1	0.83	0.69	0.75	696
avg / total	0.87	0.87	0.87	2474

Figura 29. Reporte de Clasificación de RNA con el mejor resultado de exactitud para datos de entrenamiento en el experimento 1.

La matriz de confusión de la **Figura 30** muestra la cantidad de 479 instancias correctamente clasificadas como malignas (TP) junto a un número de 217 instancias de ataque mal ubicadas (FN). Con este conjunto de datos de entrenamiento el algoritmo RNA ha clasificado 1683 instancias de manera correcta con la etiqueta de fondo «0» (TN).

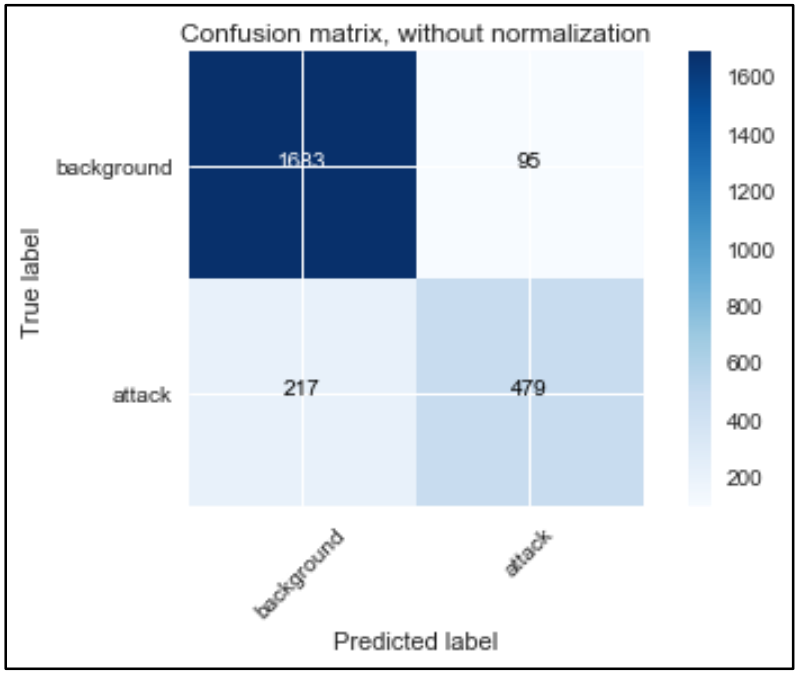

Figura 30. Matriz de Confusión de RNA con el mejor resultado de exactitud en el conjunto de datos de entrenamiento para el experimento 1.

En la **Figura 31** se puede evaluar el desempeño de la RNA con el conjunto de prueba, al igual que con el conjunto de entrenamiento, notemos que el modelo realiza mejor la tarea de recuperar las instancias de fondo que el de recuperar las instancias de ataque con los valores de sensibilidad del 0.97 y 0.57, respectivamente.

	precision	recall	f1-score	support
0	0.81	0.97	0.88	5182
1	0.92	0.57	0.70	2832
avg / total	0.84	0.83	0.82	8014

Figura 31. Reporte de Clasificación de RNA con el mejor resultado de exactitud para datos de prueba en el experimento 1.

La cantidad de instancias correctamente clasificadas como malignas (TP) es de 1619 con un número de 1213 instancias mal ubicadas como trazas de fondo cuando deberían ser malignas (FN) mucho más alto que la cantidad de 149 instancias clasificadas como malignas cuando deberían ser de fondo (FP). Estos detalles están representados en la F**igura 32**.

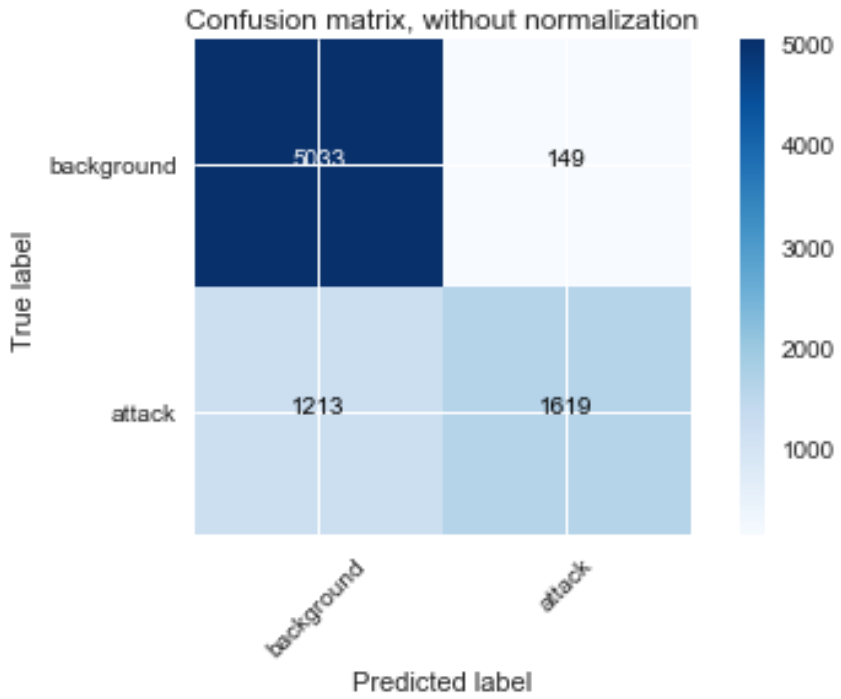

Figura 32. Matriz de Confusión de RNA con el mejor resultado de exactitud para datos de prueba en el experimento 1.

El resumen de los resultados para **Experimento 1** con el conjunto de datos para entrenamiento se representa en la **Tabla 14** y para el conjunto de datos de prueba en la **Tabla 15**. Tengamos en cuenta que estamos evaluando la precisión de cuán bueno son los algoritmos recuperando y clasificando correctamente las trazas de ataque del tráfico normal.

El algoritmo con el menor número de instancias incorrectamente clasificadas es **RF**, ligeramente menor que su contraparte **GB**. A pesar de que el recuento de FP para los tres algoritmos es relativamente bajo, fallan al separar correctamente el tráfico de fondo de los flujos maliciosos y agrupan un alto número de FN con los aciertos TN. Si recordamos que dentro de las instancias etiquetadas como *background* existe la posibilidad de encontrar trazas de ataques, podría explicar este alto número de instancias mal ubicadas ya que esto podría confundir a los modelos de aprendizaje automático.

Algoritmo	Mal Ubicados	Porcentaje Exactitud	TP	FP	FN	TN	Precisión	Sensibilidad	Medida-F1
GB	78	97.3%	629	11	67	1767	0.97	0.97	0.97
RF	69	97.2%	634	7	62	1771	0.97	0.97	0.97
RNA	312	87%	479	95	217	1683	0.87	0.87	0.87

Tabla 14. Resumen de pruebas realizadas con el conjunto de datos de entrenamiento para el Experimento 1.

Algoritmo	Mal Ubicados	Porcentaje Exactitud	TP	FP	FN	TN	Precisión	Sensibilidad	Medida-F1
GB	447	94.4%	2465	80	367	5102	0.95	0.94	0.94
RF	432	94.6%	2456	56	376	5126	0.95	0.95	0.95
RNA	1362	83%	1619	149	1213	5033	0.84	0.83	0.82

Tabla 15. Resumen de pruebas realizadas con el conjunto de datos de prueba para el Experimento 1.

3.9.2 Experimento 2: conjunto de datos de anomalías

El conjunto de datos anomalías «TrainDatasetAno.csv» contiene solamente los ataques que caen dentro de la categoría de anomalías (udpscan, ssh, spam). Un total de 20,744 instancias son utilizadas para el conjunto de entrenamiento, donde el 67% conforma las instancias de fondo «background» y el 33% las instancias de ataque «attack», para el conjunto de prueba el número total de instancias es de 9,161 con 70% de instancias de fondo y el 30% instancias de ataque. El detalle de las instancias agregadas a este conjunto de datos por categoría está representado en la **Tabla 16** y la **Tabla 17** que se encuentran a continuación:

Conjunto de Datos anomalías de Entrenamiento		
Categoría	**Cantidad**	
Background	13,981	
anomaly-udpscan	2,267	
anomaly-spam	2,250	
anomaly-ssh	2,246	
Total de Instancias	20,744	
Etiqueta	**Etiqueta Binaria**	**Cantidad**
Background	0	13,981
Attack	1	6,763

Tabla 16. Características del conjunto de datos de anomalías para entrenamiento.

Conjunto de Datos anomalías de Entrenamiento		
Categoría	**Cantidad**	
Background	6,463	
anomaly-udpscan	901	
anomaly-spam	900	
anomaly-ssh	897	
Total de Instancias	9,161	
Etiqueta	**Etiqueta Binaria**	**Cantidad**
Background	0	6,463
Attack	1	2,698

Tabla 17. Características del conjunto de datos de anomalías para prueba.

3.9.2.1 Aumento de gradiente (gradient boosting)

Las modificaciones y resultados por corrida del modelo creado con **GB** en la modalidad de clasificador se detallan en la **Tabla 18**, con el uso de los conjuntos de entrenamiento y prueba que contienen solamente las trazas de red con los ataques de tipo anomalías.

No. de Prueba	learning_rate	n_estimators	max_features	min_samples_split	Exactitud Datos Entrenamiento	Exactitud Datos Prueba
1	0.01	100	0.8	4	99.2%	89.7%
2	0.1	200	0.8	4	99.8%	92.5%
3	0.03	300	0.8	4	99.6%	92.3%
4	0.03	400	0.8	4	99.7%	92.4%
5	0.1	500	0.8	4	99.9%	92.6%

Tabla 18. Resultados de GB para el experimento 2.

El mejor resultado con el conjunto de datos de anomalías de entrenamiento se ha producido con los parámetros de la **Prueba 5,** el uso de 500 árboles de decisión aumenta ligeramente la exactitud del modelo con cada corrida. Las métricas de evaluación para el conjunto de datos de entrenamiento se muestran en la **Figura 33**. Tenemos una precisión de 1 durante el entrenamiento con una sensibilidad de 1 recuperando el 100% de las trazas malignas con un balance perfecto entre la precisión y sensibilidad.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2784
1	1.00	1.00	1.00	1365
avg / total	1.00	1.00	1.00	4149

Figura 33. Reporte de Clasificación de GB con el mejor resultado de exactitud para datos de entrenamiento en el experimento 2.

La matriz de confusión de la **Figura 34** muestra la cantidad de 1365 instancias correctamente clasificadas como malignas (TP) con un muy bajo número de 4 instancias mal ubicadas, durante el entrenamiento ha conseguido recuperar completamente las trazas de ataque al igual que de manera casi perfecta las 2780 trazas de fondo (TN).

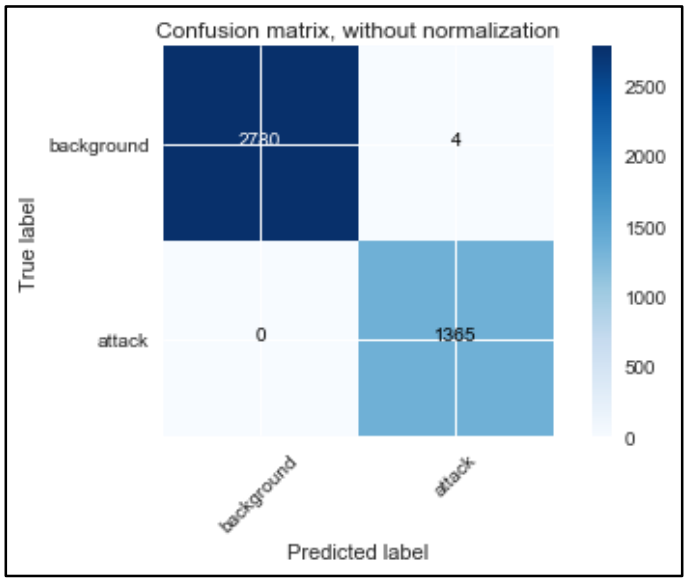

Figura 34. Matriz de Confusión de GB con el mejor resultado de exactitud en el conjunto de datos de entrenamiento para el experimento 2.

En la **Figura 35**, se logró una precisión del 0.93 al momento de predecir correctamente las instancias que caen bajo la etiqueta de ataque, de las cuales se alcanzó una sensibilidad del 0.93 recuperando el 93% de instancias malignas del conjunto total de ataques de prueba.

	precision	recall	f1-score	support
0	0.91	1.00	0.95	6463
1	0.99	0.76	0.86	2698
avg / total	0.93	0.93	0.92	9161

Figura 35. Reporte de Clasificación de GB con el mejor resultado de exactitud para datos de prueba en el experimento 2.

La matriz de confusión de la **Figura 36** muestra la cantidad de 2043 instancias correctamente clasificadas como malignas (TP) con un bajo número de 24 instancias mal ubicadas para el tráfico de fondo (FP), durante el entrenamiento ha conseguido recuperar la mayoría de las trazas de fondo (TN) pero desafortunadamente no ha conseguido la misma precisión con las 655 trazas de ataque clasificadas como fondo (FN).

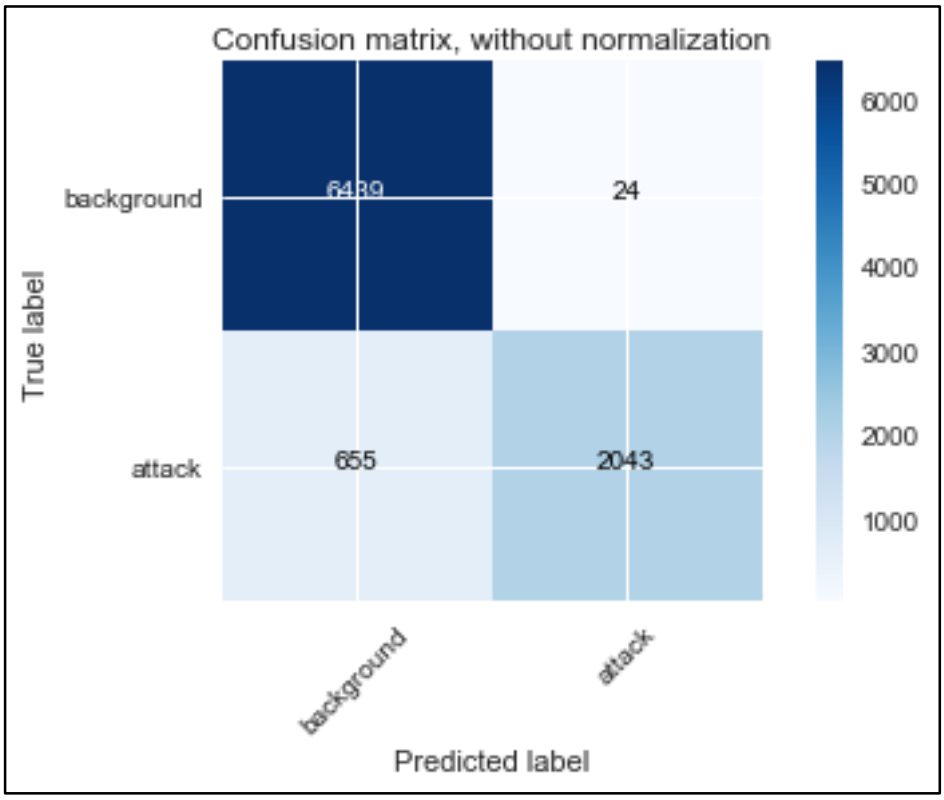

Figura 36. Matriz de Confusión de GB con el mejor resultado de exactitud en el conjunto de datos de prueba para el experimento 2.

3.9.2.2 Bosque aleatorio (random forest)

En la **Tabla 19** se muestran los hiperparámetros utilizados para correr el algoritmo **RF** junto a los resultados por corrida del modelo, con el uso de los conjuntos de entrenamiento y prueba que contienen los ataques de tipo anomalías.

No. de Prueba	n_estimators	min_samples_split	Min_samples_leaf	Exactitud Datos Entrenamiento	Exactitud Datos Prueba
1	100	6	4	99.7%	92.4%
2	100	4	2	99.8%	92.4%
3	300	4	2	99.8%	92.5%
4	400	4	2	99.8%	92.4%
5	500	4	2	99.8%	92.5%

Tabla 19. Resultados de RF para el experimento 2.

Los resultados para el modelo de entrenamiento han sido en general muy constantes con el porcentaje de exactitud rondando el 99% para los algoritmos. En este caso notamos como el número de estimadores no afecta la salida de las predicciones en gran medida para ambos conjuntos de datos. Las métricas de evaluación para el conjunto de datos de entrenamiento se muestran en la **Figura 37**. Tenemos una precisión de 1 durante el entrenamiento con una sensibilidad de 1 recuperando el 100% de las trazas malignas con un balance perfecto entre la precisión y sensibilidad.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2784
1	0.99	1.00	1.00	1365
avg / total	1.00	1.00	1.00	4149

Figura 37. Reporte de Clasificación de RF con el mejor resultado de exactitud para datos de entrenamiento en el experimento 2.

La matriz de confusión de la **Figura 38** muestra la cantidad de 1363 instancias correctamente clasificadas como malignas (TP) con un muy bajo número de 2 instancias mal ubicadas de ataque (FN), durante el entrenamiento ha conseguido recuperar casi por completo las trazas de ataque al igual que las 2777 instancias de fondo (TN).

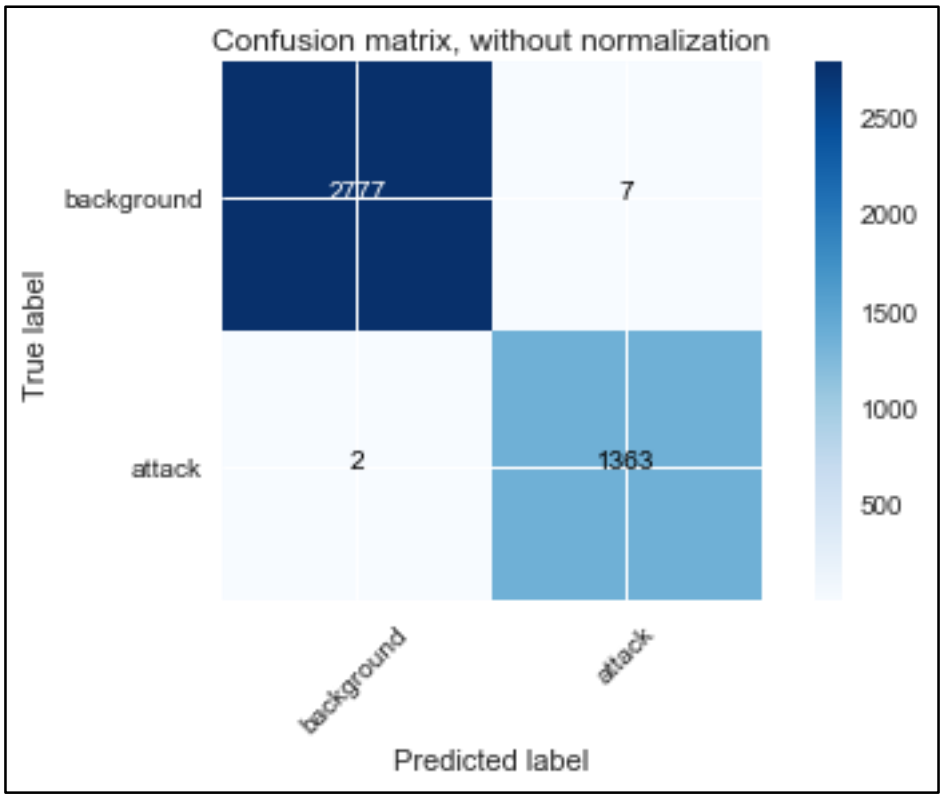

Figura 38. Matriz de Confusión de RF con el mejor resultado de exactitud en el conjunto de datos de entrenamiento para el experimento 2.

En la **Figura 39**, se logró una precisión del 0.94 al momento de predecir correctamente las instancias que caen bajo la etiqueta de ataque, de las cuales se alcanzó una sensibilidad del 0.92 recuperando el 92% de instancias malignas del conjunto total de ataques de prueba. Notamos que tiene cierta preferencia por la precisión con una medida F de 0.93.

	precision	recall	f1-score	support
0	1.00	0.91	0.95	7102
1	0.75	0.99	0.85	2059
avg / total	0.94	0.92	0.93	9161

Figura 39. Reporte de Clasificación de RF con el mejor resultado de exactitud para datos de prueba en el experimento 2.

La matriz de confusión de la **Figura 40** muestra la cantidad de 2033 instancias correctamente clasificadas como malignas (TP) con un bajo número de 26 instancias mal ubicadas para el tráfico de fondo (FN), durante el entrenamiento ha conseguido recuperar la mayoría de las trazas de fondo (TN) pero desafortunadamente no ha conseguido la misma precisión con las 665 trazas de fondo clasificadas como ataque (FP).

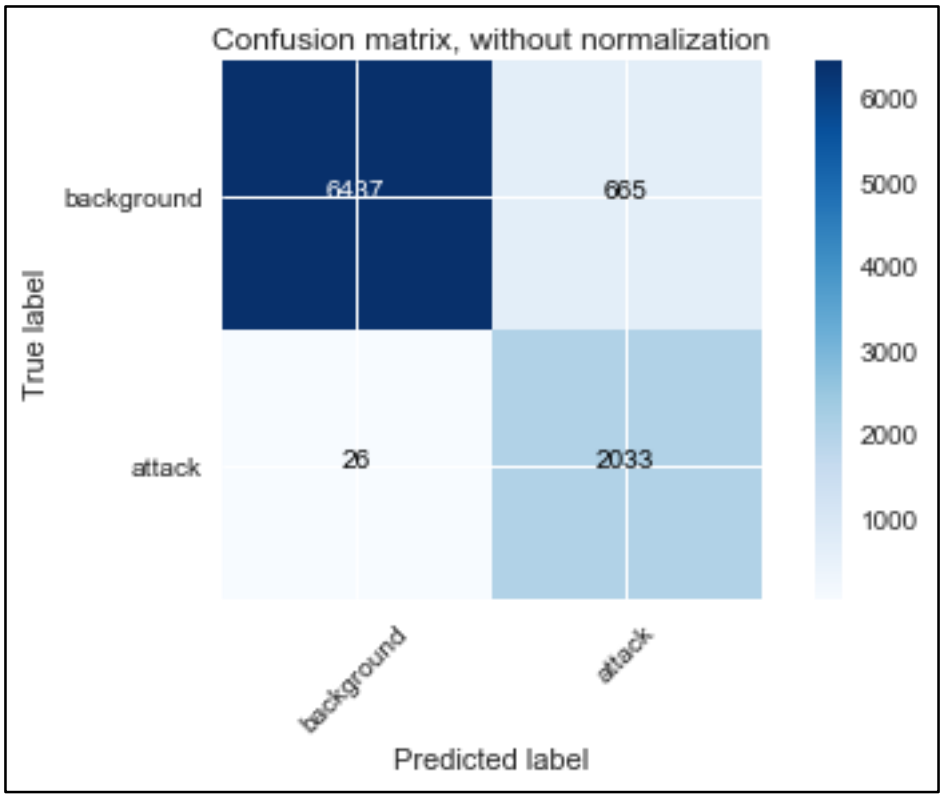

Figura 40. Matriz de Confusión de RF con el mejor resultado de exactitud en el conjunto de datos de prueba para el experimento 2.

3.9.2.3 Red neuronal artificial multicapa

Las diferentes funciones de activación y optimizadores usados en estas pruebas se resumen en la **Tabla 20**. El modelo de red neuronal artificial de 3 capas con el conjunto de datos de anomalías ha demostrado por centésimas de porcentaje un mejor rendimiento que con el conjunto de firmas. Cabe destacar que los valores de exactitud son la evaluación media de los valores de pérdida y exactitud de las 200 iteraciones realizadas por la RNA.

No. de Prueba	Función de activación capas ocultas	Función de activación capa de salida	Optimizador	Loss	Exactitud Datos Entrenamiento	Exactitud Datos Prueba
1	Relu	Sigmoid	SGD	Binary_crossentropy	97%	73%
2	Tanh	Relu	Adam	Binary_crossentropy	95%	82.5%
3	Relu	Tanh	SGD	Binary_crossentropy	33%	29.5%
4	Sigmoid	Sigmoid	SGD	Binary_crossentropy	67%	70.5%
5	Relu	Sigmoid	Adam	Binary_crossentropy	96%	81.3%

Tabla 20. Resultados de RNA para el experimento 2.

El mejor resultado con el conjunto de datos de anomalías de entrenamiento se ha producido con los parámetros de la **Prueba 2,** el uso de función de activación "tanh" para las capas ocultas del modelo y la función de activación "relu" para la capa de salida junto al optimizador "Adam" generó un modelo con exactitud media del 95% durante el entrenamiento y 82.5 % al evaluar el conjunto de datos de prueba a partir de la iteración número 30, favor referirse a la **Figura 42**. La función de pérdida comienza a reducir su valor significativamente luego de la época 30 y continúa bajando hasta alcanzar un valor de 0.105.

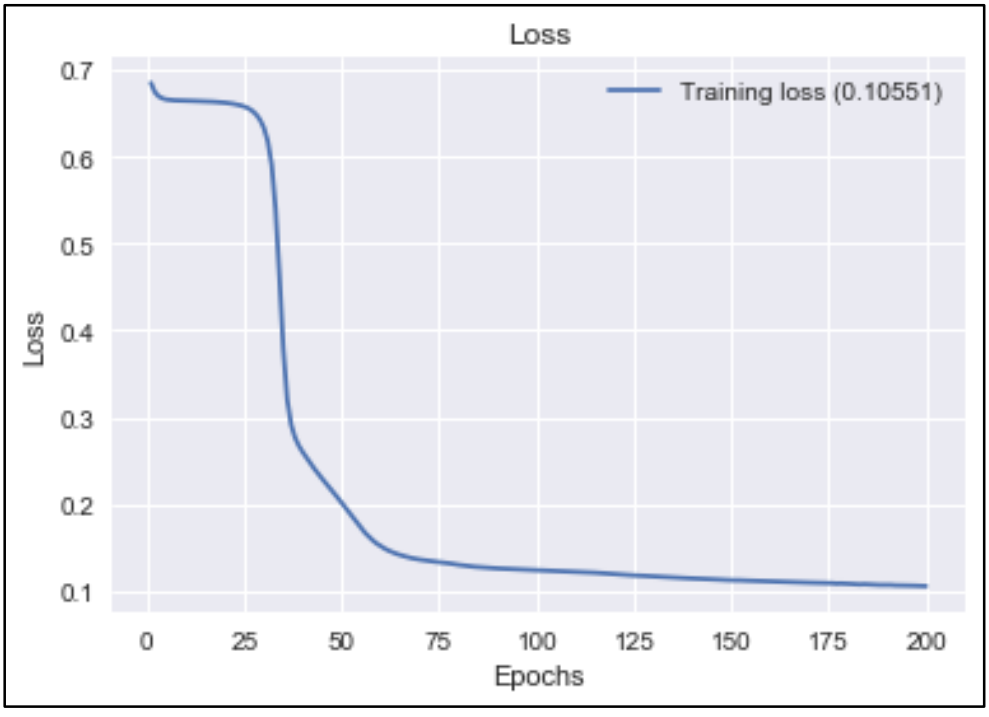

Figura 41. Valores de la función de pérdida durante épocas del modelo RNA para experimento 2.

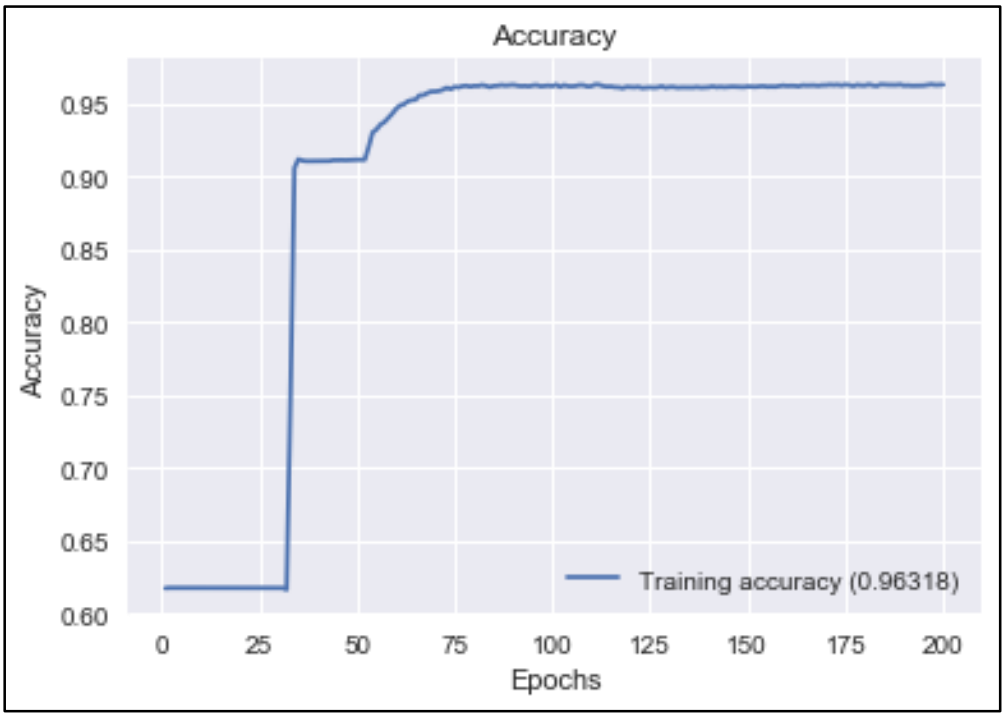

Figura 42. Valores de la exactitud durante épocas del modelo RNA para experimento 2.

La **Prueba 2** brinda los mejores resultados en cuanto a precisión con un 0.95, a pesar de no alcanzar el porcentaje de entrenamiento más alto del conjunto de pruebas, ha logrado un mejor rendimiento al ser presentado con los nuevos datos de prueba. Notamos en la **Figura 43** que durante esta corrida ha producido el valor de 0.96 para la sensibilidad del modelo al momento de recuperar las instancias de fondo, al igual que ha logrado actuar bien recuperando las instancias de ataque con una sensibilidad de 0.94 para un promedio del 95% de recuperación de las trazas.

	precision	recall	f1-score	support
0	0.97	0.96	0.96	2784
1	0.91	0.94	0.93	1365
avg / total	0.95	0.95	0.95	4149

Figura 43. Reporte de Clasificación de RNA con el mejor resultado de exactitud para datos de entrenamiento en el experimento 2.

La matriz de confusión de la **Figura 44** muestra la cantidad de 1284 instancias correctamente clasificadas como malignas (TP) junto a un número de 124 instancias de fondo mal ubicadas (FP). Con este conjunto de datos de entrenamiento el algoritmo RNA ha clasificado 2660 instancias de manera correcta con la etiqueta de fondo «0» (TN) y 81 instancias malignas como benignas (FN).

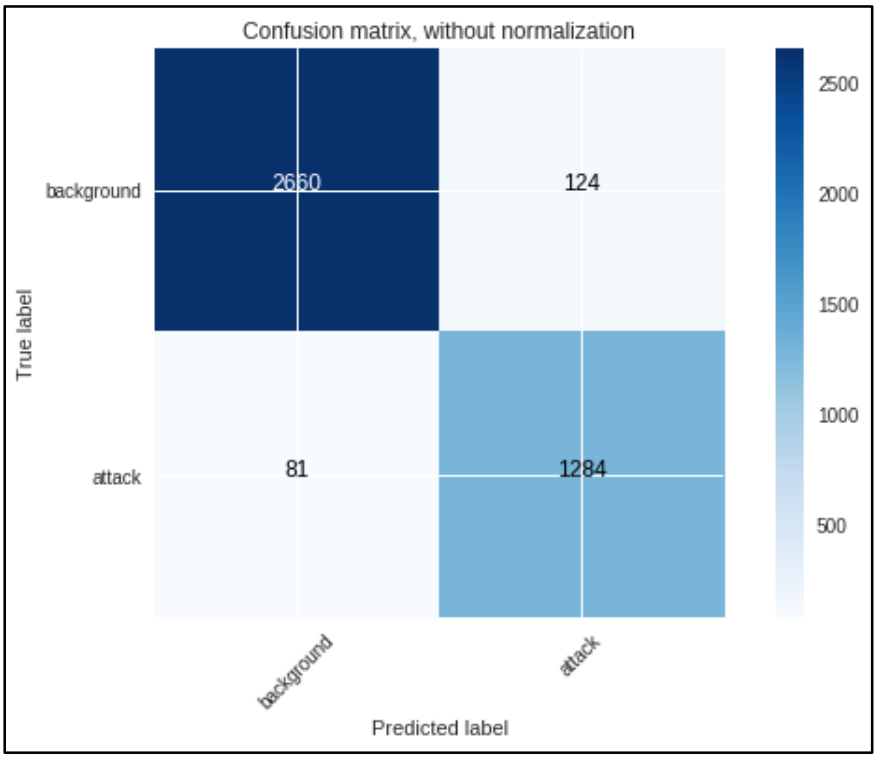

Figura 44. Matriz de Confusión de RNA con el mejor resultado de exactitud en el conjunto de datos de entrenamiento para el experimento 2.

En la **Figura 45** se puede evaluar el desempeño de la RNA con el conjunto de prueba, notamos que el modelo realiza mejor la tarea de recuperar las instancias de fondo que el de recuperar las instancias de ataque con los valores de sensibilidad del 0.93 y 0.56, respectivamente. El mismo tiene una precisión, sensibilidad y medida F de 0.82.

	precision	recall	f1-score	support
0	0.84	0.93	0.88	6463
1	0.78	0.56	0.65	2698
avg / total	0.82	0.82	0.82	9161

Figura 45. Reporte de Clasificación de RNA con el mejor resultado de exactitud para datos de prueba en el experimento 2.

La cantidad de instancias correctamente clasificadas como malignas (TP) es de 1517 con un número de 1181 instancias mal ubicadas como trazas de fondo cuando deberían ser malignas (FN) más alto que la cantidad de 426 instancias clasificadas como malignas cuando deberían ser de fondo (FP). Estos detalles están representados en la F**igura 46**.

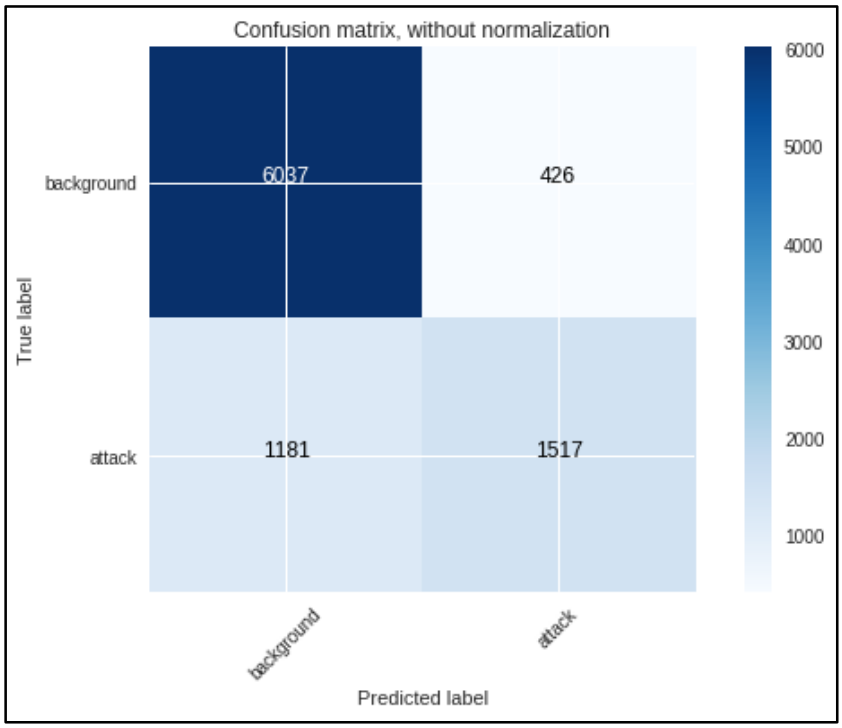

Figura 46. Matriz de Confusión de RNA con el mejor resultado de exactitud para datos de prueba en el experimento 2.

El resumen de los resultados para **Experimento 2** con el conjunto de datos para entrenamiento se representa en la **Tabla 21** y para el conjunto de datos de prueba en la **Tabla 22**. El algoritmo con el menor número de instancias incorrectamente clasificadas es **GB**, ligeramente menor que su contraparte **RF**. A pesar de que el recuento de FP para los tres algoritmos es relativamente bajo durante el entrenamiento, fallan al separar correctamente el tráfico de fondo de los flujos maliciosos y agrupan un alto número de trazas benignas (FN) con los aciertos de las trazas que realmente tienen la etiqueta de *background* (TN).

Nuevamente debemos considerar que dentro de las instancias etiquetadas como *background* existe la posibilidad de encontrar trazas de ataques, podría explicar este alto número de instancias mal ubicadas ya que esto podría confundir a los modelos de aprendizaje automático. Los tres algoritmos durante el entrenamiento han alcanzado un alto porcentaje, pero este porcentaje no se mantiene al utilizar el conjunto de datos de prueba, especialmente el modelo RNA no logra una exactitud mayor a 83% y tiene problemas distinguiendo las categorías de las clases.

Algoritmo	Mal Ubicados	Porcentaje Exactitud	TP	FP	FN	TN	Precisión	Sensibilidad	Medida-F1
GB	4	99.9%	1365	4	0	2780	100%	100%	1
RF	9	99.8%	1363	7	2	2777	100%	100%	1
RNA	205	95%	1284	124	81	2660	95%	95%	0.95

Tabla 21. Resumen de pruebas realizadas con el conjunto de datos de entrenamiento para el Experimento 2.

Algoritmo	Mal Ubicados	Porcentaje Exactitud	TP	FP	FN	TN	Precisión	Sensibilidad	Medida-F1
GB	679	92..6%	2043	24	655	6439	93%	93%	0.92
RF	691	92.5%	2033	26	665	6437	94%	92%	0.93
RNA	1607	82.5%	1517	426	1181	6037	82%	82%	0.82

Tabla 22. Resumen de pruebas realizadas con el conjunto de datos de prueba para el Experimento 2.

3.9.3 Experimento 3: conjunto de datos completo

El conjunto de datos completo «TrainDatasetAll.csv» combina las dos categorías de ataques previamente analizadas por separado. En un entorno de red normal, se suele instalar un IDS para cada tipo de detección de ataques ya que sus características podrían suponer que no hay una similitud que facilite su detección. De igual manera evaluaremos cuál es el desempeño de los algoritmos de aprendizaje automático cuando son presentados con el reto de clasificar los ataques en conjunto.

Un total de 13,995 instancias son utilizadas para el conjunto de entrenamiento, donde el 70% conforma las instancias de fondo «background» y el 30% las instancias de ataque «attack», para el conjunto de prueba el número total de instancias es de 9,161 con 70% de instancias de fondo y el 30% instancias de ataque. El detalle de las instancias agregadas a este conjunto de datos por categoría está representado en la **Tabla 23** y la **Tabla 24** que se encuentran a continuación:

Conjunto de Datos Completo de Entrenamiento		
Categoría	**Cantidad**	
Background	23,141	
Blacklist	738	
Scan44	700	
Dos	700	
Scan11	700	
Nerisbotnet	372	
anomaly-udpscan	2,267	
anomaly-spam	2,250	
anomaly-ssh	2,246	
Total de Instancias	33,114	
Etiqueta	**Etiqueta Binaria**	**Cantidad**
Background	0	23,141
Attack	1	9,973

Tabla 23. Características de los conjuntos de datos de UGR-16

Conjunto de Datos Completo de Entrenamiento		
Categoría	**Cantidad**	
Background	7,265	
Blacklist	209	
Scan44	205	
Dos	217	
Scan11	202	
Nerisbotnet	212	
anomaly-udpscan	200	
anomaly-spam	202	
anomaly-ssh	25	
Total de Instancias	8,737	
Etiqueta	**Etiqueta Binaria**	**Cantidad**
Background	0	7,265
Attack	1	1,472

Tabla 24. Características de los conjuntos de datos de UGR-16

3.9.3.1 Aumento de gradiente (gradient boosting)

Las modificaciones y resultados por corrida del modelo creado con **GB** en la modalidad de clasificador se detallan en la **Tabla 45**, con el uso de los conjuntos de entrenamiento y prueba que contienen todos los ataques.

No. de Prueba	learning_rate	n_estimators	max_features	min_samples_split	Exactitud Datos Entrenamiento	Exactitud Datos Prueba
1	0.1	100	0.8	4	96.4%	92%
2	0.01	100	0.8	4	92.6%	89.1%
3	0.2	200	0.8	4	96%	91.7%
4	0.3	300	0.8	4	96.3%	91.9%
5	0.2	500	0.8	4	98%	94.3%

Tabla 25. Resultados de GB para el experimento 3.

El mejor resultado con todos los ataques se ha producido con los parámetros de la **Prueba 5,** el uso de 500 árboles de decisión aumenta la exactitud del modelo con cada corrida usando una tasa de aprendizaje de 0.2. Las métricas de evaluación para el conjunto de datos de entrenamiento se muestran en la **Figura 47**. Tenemos una precisión de 0.99 durante el entrenamiento con una sensibilidad de 0.99 recuperando el 99% de las trazas malignas con una buena medida-F.

	precision	recall	f1-score	support
0	0.99	1.00	0.99	4608
1	0.99	0.97	0.98	2015
avg / total	0.99	0.99	0.99	6623

Figura 47. Reporte de Clasificación de GB con el mejor resultado de exactitud para datos de entrenamiento en el experimento 3.

La matriz de confusión de la **Figura 48** muestra la cantidad de 1955 instancias correctamente clasificadas como malignas (TP) con un número de 13 instancias mal ubicadas como ataque (FP), durante el entrenamiento ha conseguido recuperar casi todas las 4595 trazas de fondo (TN).

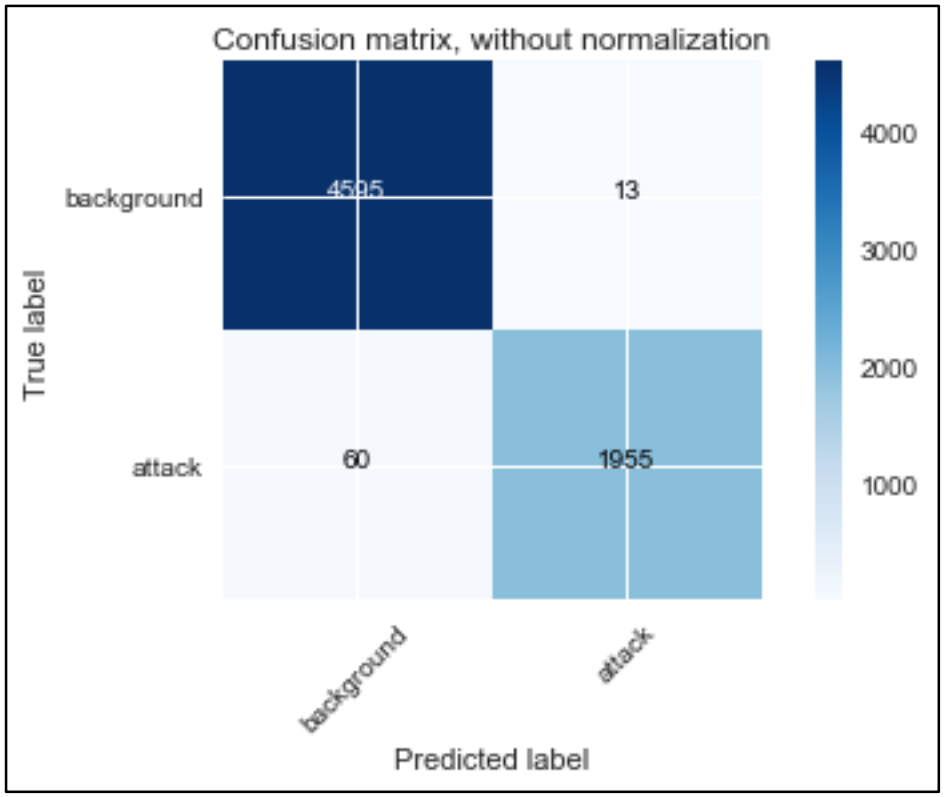

Figura 48. Matriz de Confusión de GB con el mejor resultado de exactitud en el conjunto de datos de entrenamiento para el experimento 3.

En la **Figura 49**, se logró una precisión del 0.96 al momento de predecir correctamente las instancias que caen bajo la etiqueta de ataque, de las cuales se alcanzó una sensibilidad del 0.98 recuperando el 98% de instancias benignas del conjunto total de prueba. Con respecto a las evaluaciones para las trazas de ataques se obtuvo una precisión de 0.87 y recuperó 78% de las mismas.

	precision	recall	f1-score	support
0	0.96	0.98	0.97	7265
1	0.87	0.78	0.82	1472
avg / total	0.94	0.94	0.94	8737

Figura 49. Reporte de Clasificación de GB con el mejor resultado de exactitud para datos de prueba en el experimento 3.

La matriz de confusión de la **Figura 50** muestra la cantidad de 1142 instancias correctamente clasificadas como malignas (TP) con un número de 170 instancias mal ubicadas para el tráfico de fondo (FP), ha conseguido recuperar 7095 de las trazas de fondo (TN) pero desafortunadamente no ha conseguido la misma precisión con las 330 trazas de ataque clasificadas como fondo (FN).

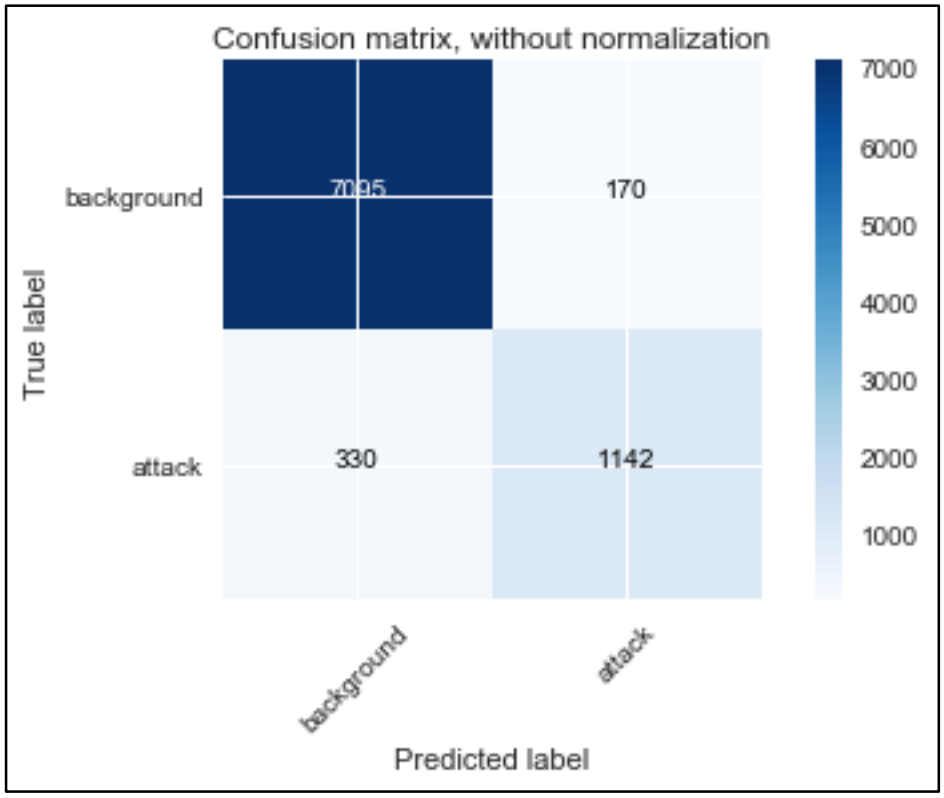

Figura 50. Matriz de Confusión de GB con el mejor resultado de exactitud en el conjunto de datos de prueba para el experimento 3.

3.9.3.2 Bosque aleatorio (random forest)

En la **Tabla 26** se muestran los hiperparámetros utilizados para correr el algoritmo **RF** junto a los resultados por corrida del modelo, con el uso de los conjuntos de entrenamiento y prueba que contienen todos los ataques combinados.

No. de Prueba	n_estimators	min_samples_split	Min_samples_leaf	Exactitud Datos Entrenamiento	Exactitud Datos Prueba
1	100	6	4	98.7%	95.3%
2	200	4	2	99.1%	95.5%
3	300	4	2	99%	95.5%
4	400	4	2	99%	95.4%
5	500	4	2	99%	95.5%

Tabla 26. Resultados de RF para el experimento 3.

Los resultados para el modelo de entrenamiento han sido en general muy constantes con el porcentaje de exactitud rondando el 99% para los algoritmos. Como con el **Experimento 2** con este conjunto de datos el número de estimadores no afecta mucho la salida de las predicciones. Las métricas de evaluación para el conjunto de datos de entrenamiento se muestran en la **Figura 51**. Tenemos una precisión de 1 para la etiqueta de ataque durante el entrenamiento con una sensibilidad de 0.99 recuperando el 97% de las trazas malignas con un balance casi perfecto entre la precisión y sensibilidad.

	precision	recall	f1-score	support
0	0.99	1.00	0.99	4608
1	1.00	0.97	0.98	2015
avg / total	0.99	0.99	0.99	6623

Figura 51. Reporte de Clasificación de RF con el mejor resultado de exactitud para datos de entrenamiento en el experimento 3.

La matriz de confusión de la **Figura 52** muestra la cantidad de 1956 instancias correctamente clasificadas como malignas (TP) con un muy bajo número de 4 instancias mal ubicadas de fondo (FP), durante el entrenamiento ha conseguido recuperar casi por completo las 4604 trazas de fondo (TN) al igual que las 59 instancias de ataque mal ubicadas como fondo (FN).

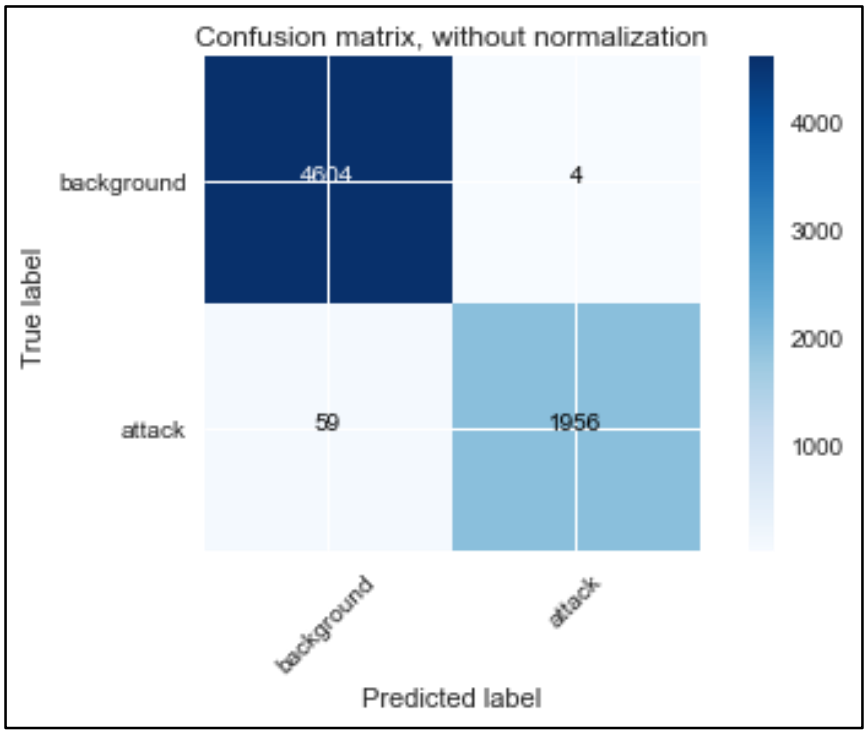

Figura 52. Matriz de Confusión de RF con el mejor resultado de exactitud en el conjunto de datos de entrenamiento para el experimento 3.

En la **Figura 53**, se logró una precisión del 0.99 al momento de predecir correctamente las instancias que caen bajo la etiqueta de fondo, de las cuales se alcanzó una sensibilidad del 0.96. Con la etiqueta de ataque el modelo no consigue la misma precisión con 0.79, sin embargo consigue una buena sensibilidad recuperando el 94% de instancias malignas del conjunto total de ataques de prueba.

	precision	recall	f1-score	support
0	0.99	0.96	0.97	7498
1	0.79	0.94	0.86	1239
avg / total	0.96	0.96	0.96	8737

Figura 53. Reporte de Clasificación de RF con el mejor resultado de exactitud para datos de prueba en el experimento 3.

La matriz de confusión de la **Figura 54** muestra la cantidad de 1159 instancias correctamente clasificadas como malignas (TP) con un número de 80 instancias mal ubicadas para el tráfico de fondo (FN), ha conseguido recuperar 7185 de las trazas de fondo (TN) pero desafortunadamente no ha conseguido la misma precisión con las 313 trazas de fondo clasificadas como ataque (FP).

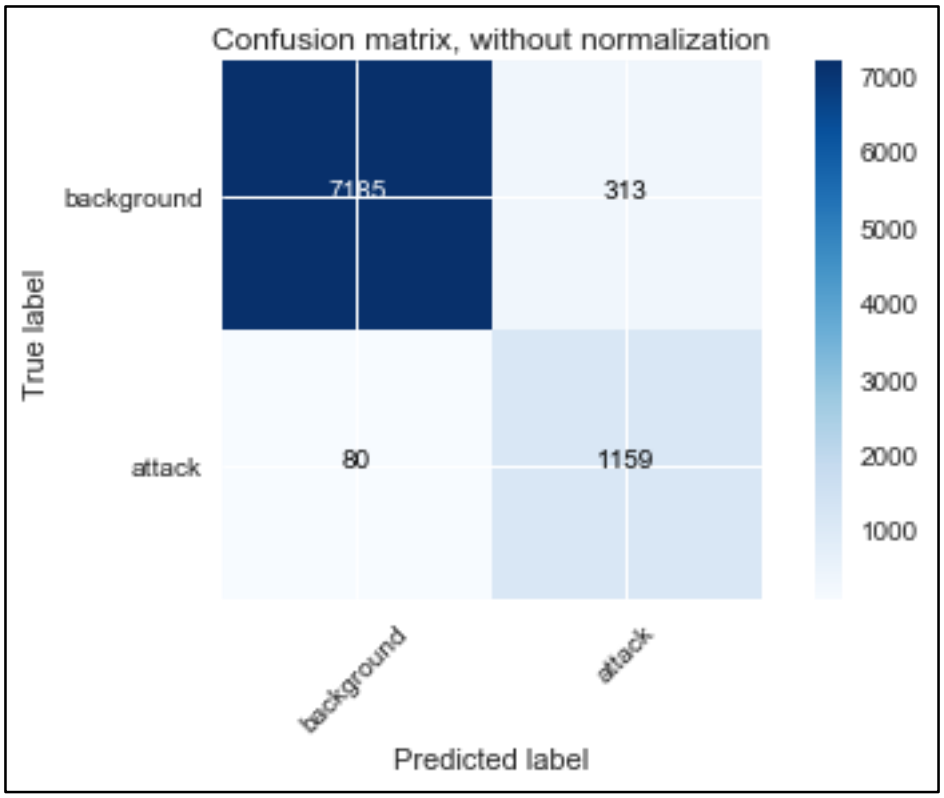

Figura 54. Matriz de Confusión de RF con el mejor resultado de exactitud en el conjunto de datos de prueba para el experimento 3.

3.9.3.3 Red neuronal artificial multicapa

No. de Prueba	Función de Activación Capas Ocultas	Función de Activación Capa de Salida	Optimizador	Loss Function	Exactitud Datos Entrenamiento	Exactitud Datos Prueba
1	Relu	Sigmoid	SGD	Binary_crossentropy	88%	85%
2	Tanh	Relu	Adam	Binary_crossentropy	88%	74%
3	Relu	Tanh	SGD	Binary_crossentropy	70%	83%
4	Sigmoid	Sigmoid	SGD	Binary_crossentropy	89%	61.8%
5	Relu	Sigmoid	Adam	Binary_crossentropy	94%	78.4%

Tabla 27. Resultados de RNA para el experimento 3.

Las diferentes funciones de activación y optimizadores usados en estas pruebas se resumen en la **Tabla 27**. El modelo de red neuronal artificial de 3 capas con el conjunto de datos completo demostró una ligera mejora en el porcentaje de exactitud para clasificar las instancias de ataque y fondo. Podemos observar que en la **Prueba 5** el

modelo durante el entrenamiento tuvo un alto porcentaje de exactitud del 94% en comparación con las demás pruebas sin embargo no arrojó las mejores predicciones para el conjunto de prueba dando 78.4% de exactitud.

El mejor resultado con el conjunto de datos de anomalías de entrenamiento se ha producido con los parámetros de la **Prueba 1,** el uso de función de activación "relu" para las capas ocultas del modelo y la función de activación "sigmoid" para la capa de salida junto al optimizador "SGD" generó un modelo con exactitud media del 88% durante el entrenamiento y 85% al evaluar el conjunto de datos de prueba a partir de la iteración número 30, la convergencia de la exactitud se puede apreciar en la **Figura 56**. La función de pérdida comienza a reducir su valor significativamente luego de la época 40 y continúa bajando hasta alcanzar un valor de 0.27, referirse a la **Figura 55**.

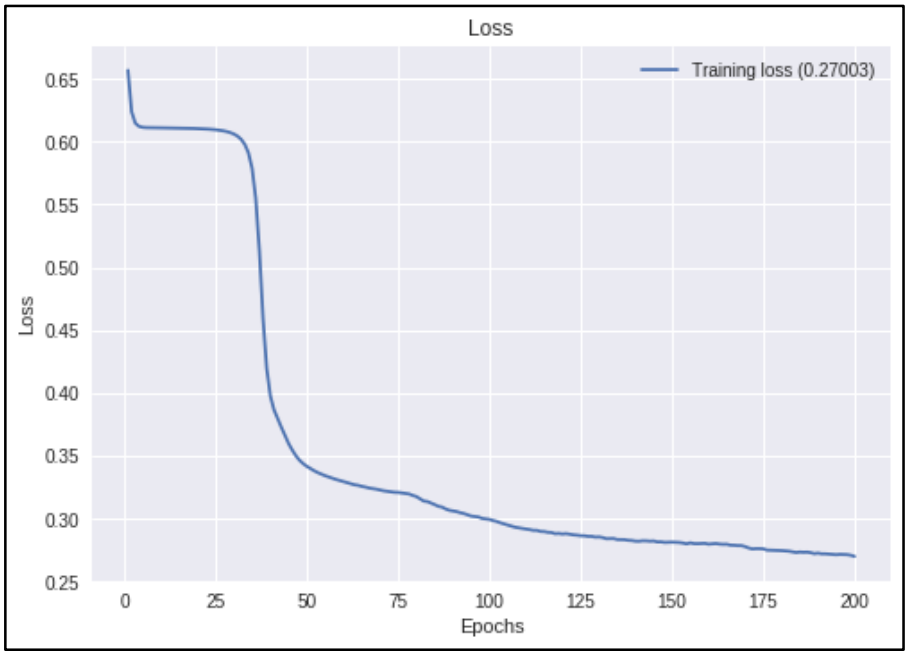

Figura 55. Valores de la función de pérdida durante épocas del modelo RNA para experimento 3.

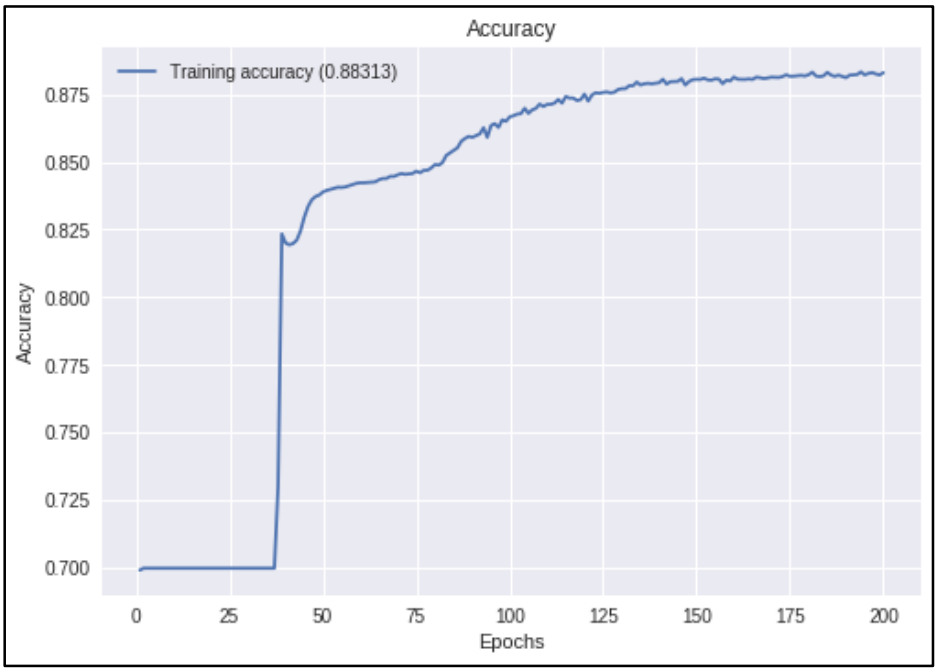

Figura 56. Valores de la exactitud durante épocas del modelo RNA para experimento 3.

La **Prueba 1** brinda los mejores resultados en cuanto a precisión con un 0.88, a pesar de no alcanzar el porcentaje de entrenamiento más alto del conjunto de pruebas, ha logrado un mejor rendimiento al ser presentado con los nuevos datos de prueba. Notamos en la **Figura 57** el valor de 0.95 para la sensibilidad del modelo al momento de recuperar las instancias de fondo, pero no logró actuar bien recuperando las instancias de ataque con una sensibilidad de 0.71 para un promedio del 88% de recuperación de las trazas en ambas categorías.

	precision	recall	f1-score	support
0	0.88	0.95	0.92	4608
1	0.87	0.71	0.78	2015
avg / total	0.88	0.88	0.87	6623

Figura 57. Reporte de Clasificación de RNA con el mejor resultado de exactitud para datos de entrenamiento en el experimento 3.

La matriz de confusión de la **Figura 58** muestra la cantidad de 1430 instancias correctamente clasificadas como malignas (TP) junto a un número de 220 instancias de fondo mal ubicadas (FP). Con este conjunto de datos de entrenamiento el algoritmo RNA ha clasificado 4388 instancias de manera correcta con la etiqueta de fondo «0» (TN) y 585 instancias malignas como benignas (FN) lo cual es una cantidad alta.

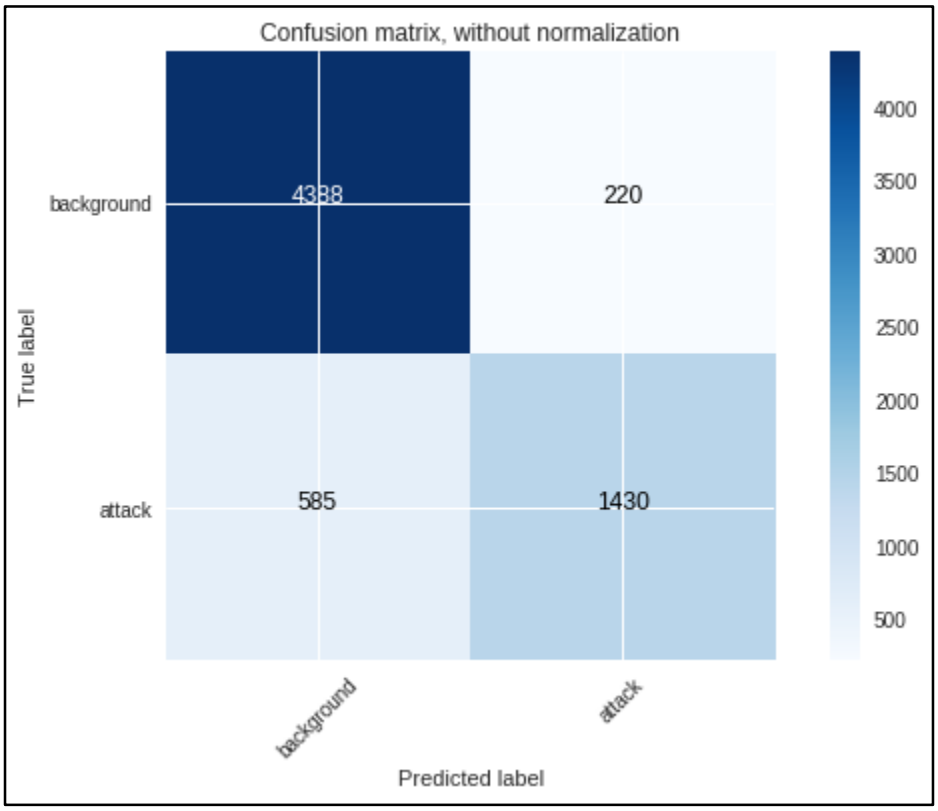

Figura 58. Matriz de Confusión de RNA con el mejor resultado de exactitud en el conjunto de datos de entrenamiento para el experimento 3.

En la **Figura 59** se representa las métricas de evaluación de la RNA con el conjunto de prueba, nuevamente el modelo termina la tarea de recuperar las instancias de fondo con una alta sensibilidad de 0.95 mas no corre con la misma precisión al recuperar las instancias de ataque con una sensibilidad del 0.37, la precisión media del modelo es de 0.84.

	precision	recall	f1-score	support
0	0.88	0.95	0.92	7265
1	0.61	0.37	0.46	1472
avg / total	0.84	0.85	0.84	8737

Figura 59. Reporte de Clasificación de RNA con el mejor resultado de exactitud para datos de prueba en el experimento 3.

La cantidad de instancias correctamente clasificadas como malignas (TP) es de 550 con un número de 922 instancias mal ubicadas como trazas de fondo cuando deberían ser malignas (FN) más alto que la cantidad de 345 instancias clasificadas como malignas cuando deberían ser de fondo (FP). Estos detalles están representados en la F**igura 60**.

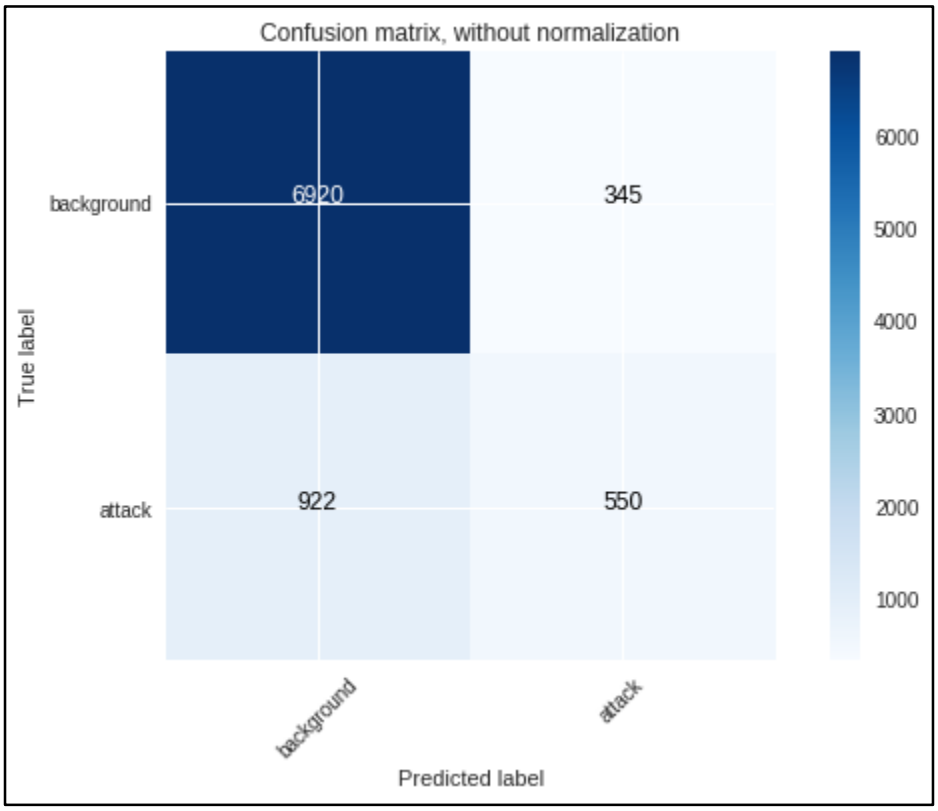

Figura 60. Matriz de Confusión de RNA con el mejor resultado de exactitud para datos de prueba en el experimento 3.

Discutiremos los resultados para el **Experimento 3** con el conjunto de datos para entrenamiento representado en la **Tabla 21** y para el conjunto de datos de prueba representado en la **Tabla 22**. El algoritmo con el menor número de instancias incorrectamente clasificadas es una vez más **RF**, que supera el rendimiento para la **RNA**. En esta ocasión el recuento de FP para los tres algoritmos no es el mismo, ya que la RNA tienen una cantidad alta de instancias mal ubicadas durante el entrenamiento, fallan al separar correctamente el tráfico de fondo de los flujos de ataque y agrupan un alto número de trazas de fondos (FN) con los aciertos de las trazas que realmente tienen la etiqueta de *background* (TN), todo lo anterior mencionado se presentó con el conjunto de datos de prueba. Los algoritmos basados en árboles de decisiones han demostrado superar el rendimiento del algoritmo **RNA** cuando son presentados con todas las instancias de ataques combinadas, podría darse el beneficio de la duda a la red neuronal ya que no tenemos la garantía de cuál es el porcentaje de instancias dentro de la etiqueta de fondo que realmente pertenecen a una amenaza.

Algoritmo	Mal Ubicados	Porcentaje Exactitud	TP	FP	FN	TN	Precisión	Sensibilidad	Medida-F1
GB	73	98%	1955	13	60	4595	99%	99%	0.99
RF	63	99%	1956	4	59	4604	99%	99%	0.99
RNA	805	94%	1430	220	585	4388	88%	88%	0.87

Tabla 28. Resumen de pruebas realizadas con el conjunto de datos de entrenamiento para el Experimento 3.

Algoritmo	Mal Ubicados	Porcentaje Exactitud	TP	FP	FN	TN	Precisión	Recall	Medida-F1
GB	500	94.3%	1142	170	330	7095	94%	94%	0.94
RF	393	95.5%	1159	313	80	7185	96%	96%	0.96
RNA	1,267	85.5%	550	345	922	6920	84%	85%	0.84

Tabla 29. Resumen de pruebas realizadas con el conjunto de datos de prueba para el Experimento 3.

3.10 Comparación de los resultados

Tras realizar los experimentos descritos con distintos grupos de datos, se puede concluir que utilizando un solo grupo de ataques es posible obtener buenos resultados de predicción. Sin embargo, en el Experimento 3 realizado con el conjunto completo de datos de ambos tipos de ataques, es posible observar un declive en el desempeño del algoritmo en comparación con los Experimentos 1 y 2, con una baja en la exactitud de predicción cuando todos los tipos de ataques en conjunto son usados.

La exactitud de los tres algoritmos durante los experimentos con los conjuntos de datos analizados de prueba es representada en la **Figura 61**, los algoritmos de ensamblaje basados en árboles de decisión han demostrado un buen rendimiento en comparación con el uso de la red neuronal multicapa para todos los casos. A pesar de tratarse de algoritmos de ensamblaje que en su composición principal utilizan árboles de decisión para generar sus predicciones estas son utilizados de maneras diferentes en la votación interna por dar la mejor salida de resultados.

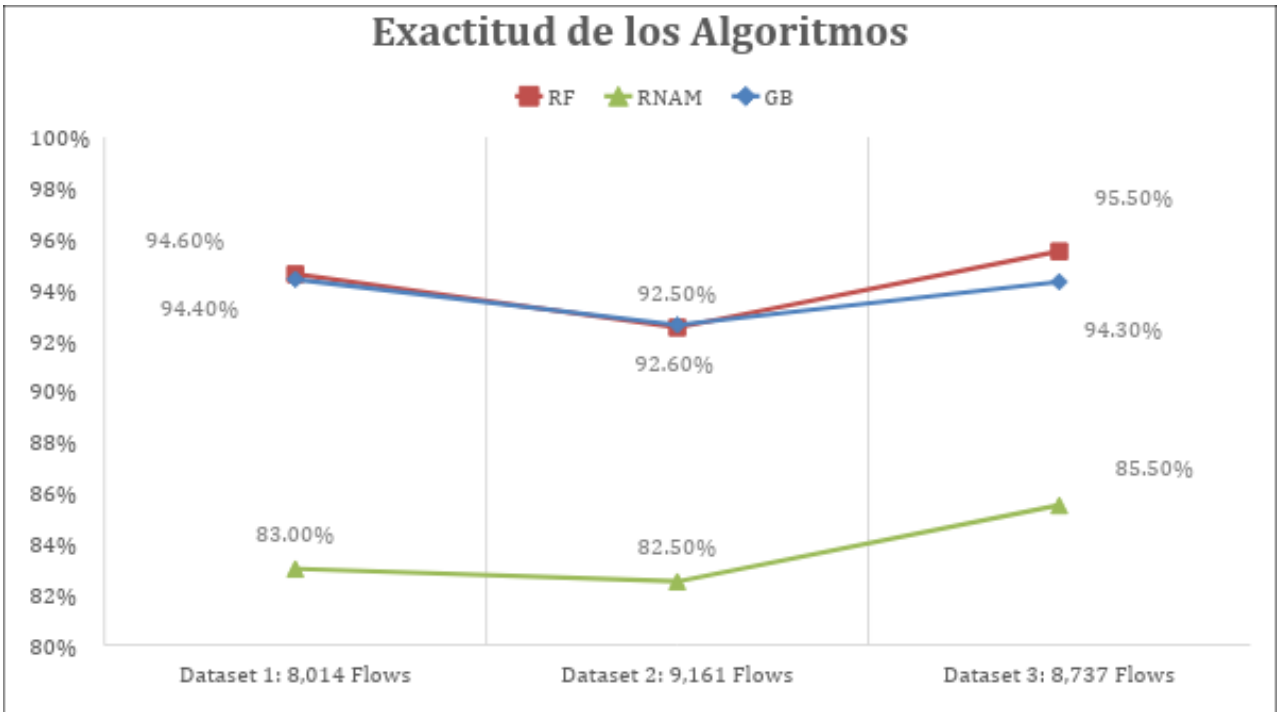

Figura 61. Comparación de la exactitud de cada algoritmo usando el conjunto de datos de prueba.

Los resultados brindados son cercanos en ambas situaciones, podría inferirse que la manera en que agrupan las características de las trazas de red es estable para el conjunto de datos alimentado, **GB** con el primer conjunto de datos -firmas- dio una exactitud del 94.40%, casi igual a la de **RF** con 94.60%. Un porcentaje del 92% de exactitud para el

segundo conjunto de datos -anomalías- y cuando son presentados con todos los ataques combinados **RF** demuestra un rendimiento ligeramente mejor que **GB** con un 95.50% y 94.30%, respectivamente. Ambos algoritmos en general mantienen un porcentaje más alto, pero no necesariamente ideal ya que en este caso se busca llegar a un porcentaje de detección de trazas malignas de un 99%, pero los resultados son prometedores.

Con respecto a la exactitud de la red neuronal multicapa con el primer conjunto de datos -firmas- se logró una exactitud del 83% con el segundo conjunto de datos -anomalías- alcanzó un 82.50% pero el modelo al ser presentado con todos los tipos de ataques en el conjunto de datos 3 ligeramente aumento la exactitud a un 85.50%, lo cual podría indicar que el modelo manejaría bien grandes volúmenes de datos provenientes de diferentes fuentes de datos dentro de una red.

El algoritmo debe enfocarse en recuperar tantos flujos maliciosos como sea posible rechazando notablemente los flujos de la clase normal. En última instancia, construir modelos que tengan un alto valor de pureza para las predicciones maliciosas es importante, es decir, las conexiones de red etiquetadas como maliciosas no deben contener instancias del tráfico normal para evitar rechazar las solicitudes del usuario y las actividades de la red, al igual que etiquetar algunos flujos maliciosos como normales y dejar que se mezclen con el tráfico normal no es aceptable dentro del rango de detección de un IDS.

En la **Tabla 30** podemos comparar los valores promedio de rendimiento real de cada algoritmo. RF demuestra ser un competidor digno contra los flujos de ataque y puede ser utilizado para mejorar la precisión de los sistemas de detección junto con otras herramientas de seguridad. Creemos que cada algoritmo funciona dependiendo de la calidad de los datos y el preprocesamiento previo de los mismos. Por lo tanto, tal vez una combinación de algoritmos de aprendizaje con diferentes parámetros puede dar buenos resultados si se utilizan por cada categoría de ataques. Todo depende de la naturaleza de la alimentación de datos para cada algoritmo de minería de datos.

Algorithms	**Average Accuracy**	**Average Precision**	**Average Recall**	**Average F-measure**
GB	93.76%	0.94	0.93	0.93
RF	94.2%	0.95	0.94	0.94
RNAM	83.6%	0.83	0.83	0.82

Tabla 30. Valores de evaluación promedio por cada algoritmo para todos los experimentos.

La relación promedio de precisión y sensibilidad que se muestra en la **Figura 62** la cual nos da una idea del equilibrio entre estos valores. Los valores ideales siempre deben acercarse a 1 y mantener una buena proporción entre el porcentaje de cuántos

objetos recuperados son maliciosos y la cantidad real de objetos maliciosos que aún deben clasificarse como maliciosos, pero no lo son.

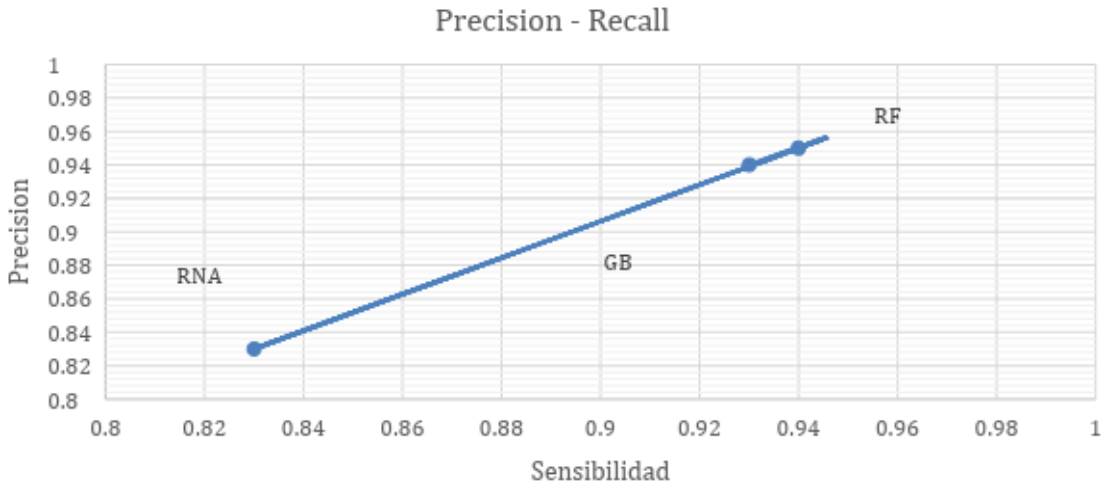

Figura 62. Relación promedio de precisión y sensibilidad por algoritmo.

4. Conclusiones

El objetivo de este trabajo ha sido proponer el uso de la inteligencia artificial en labores de seguridad informática valiéndose de los algoritmos de aprendizaje automático. Se ha realizado una comparación del rendimiento de diferentes algoritmos de aprendizaje automático para clasificar amenazas en tres categorías dentro de una red, los enfoques basados en firmas y anomalías al ser analizados por su cuenta generan menos falsos positivos. Para los enfoques de firmas su efectividad depende completamente de una base de datos de firmas de intrusión preexistentes, los NIDS basados en firmas a menudo quedan paralizados frente a nuevos y sofisticados ataques, aquí es donde tienen la oportunidad de ayudar la inteligencia artificial, con modelos como las redes neuronales que puedes aprender a identificar patrones al ser presentadas con altos volúmenes de datos.

Durante los experimentos se notó el gran potencial de los algoritmos de potenciamiento y ensamblaje para analizar y clasificar correctamente ataques de los que se tiene previo conocimiento sobre su comportamiento, incluso cuando debe analizar firmas y anomalías en conjunto.

Para finalizar podemos concluir que los objetivos propuestos en el apartado **1.3** se han podido alcanzar durante el desarrollo de este trabajo. Aún es necesario investigar y asegurar un alto nivel de detección que sea confiable para evitar a toda costa los falsos negativos y las amenazas de red no se filtren desapercibidas.

4.1. Futuras líneas de trabajo

Con la discusión de los resultados terminada se procederá a la enumeración de implementaciones y cambios que ayudarían a mejor el desempeño de los algoritmos de inteligencia artificial en futuras líneas de trabajo.

4.2. Realizar optimización y análisis de los atributos

En este Trabajo Fin de Máster se ha optado por enfocar los esfuerzos de los algoritmos en la detección de los dos tipos de ataques, con esta implementación se asemeja a un **IDS**, ya que tendría la capacidad de detectar cualquier tipo de ataque. Por lo tanto, en una futura implementación sería recomendable alimentar los algoritmos con atributos cuidadosamente elegidos y estudiados a profundidad aplicando métodos de optimización de datos como por ejemplo LIME, Ribeiro et al. [50] o SHAP, Lundberg et al. [51].

4.3. Combinación de algoritmos de aprendizaje

En este Trabajo Fin de Máster se discutió sobre los algoritmos de ensamblaje y potenciamiento utilizando módulos previamente armados en las librerías de Python, pero existe la posibilidad de investigar el desempeño del uso de varios algoritmos en conjunto, es decir utilizar un tipo de algoritmo para analizar el tráfico en busca de ataques de firmas, un segundo algoritmo encargado de analizar el tráfico en busca de ataques de anomalías y un algoritmo como última línea de detección que valide las predicciones en conjunto. Sería una combinación de un primer modelo de ensamblaje en paralelo y un segundo modelo en modalidad secuencial.

4.4. Implementación en una situación real

El objetivo de un **IDS** es monitorizar la red en tiempo real y detectar amenazas, por lo tanto, tras las mejoras y correcciones comentadas en los anteriores apartados de esta sección, es implementar el modelo a una red real. Para esta tarea deben evaluarse el tiempo que tardan los algoritmos para entregar los resultados de salida, agregando el tiempo del tratado de los datos en tiempo real antes de ser alimentados al modelo. El tiempo que toma realizar este proceso no debe super milisegundos esto representaría un gran reto para su implementación en una situación real.

5. Bibliografía

1. **Stuart J. Russell, Peter Norvig.** *Artificial Intelligence: A Modern Approach.* 3. s.l. : Prentice Hall, 2009.

2. **Flach, Peter.** *Machine Learning: The art and Science of Algorithms that Make Sense of Data.* s.l. : Cambridge University Press, 2012.

3. **Quinlan, J. R.** *Induction of Decision Trees,* Machine Learning, Kluwer Academic Publishers, vol.1: 81-106 pgs., 1986

4.**Dr. Sayad**, **Saed.** *An Introduction to Data Science*, Decision Tree Classification, [En línea] [Citado el: 3 de Junio de 2018.] URL: http://www.saedsayad.com/decision_tree.htm.

5. **Zhou, Zhi-Hua.** *What is the difference between Bagging and Boosting?*, Quant Dare, [En línea] [Citado el: 3 de Junio de 2018.] URL: https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

6. **aporras.** *Data Mining: Practical Machine Learning Tools and Techniques.* 3. s.l. : Morgan Kaufmann, 2016.

7. **Brownlee, Jason.** *A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning*, Machine Learning Mastery, [En línea] [Citado el: 3 de Junio de 2018.] URL: https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/.

8. **Rogozhnikov, Alex.** *Gradient Boosting explained [demonstration],* [En línea] [Citado el: 3 de Junio de 2018.] URL: http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html

9. **Cruz, Pedro Ponce.** *Inteligencia artificial con aplicaciones a la ingeniería.* s.l. : Alfaomega, 2011.

10. **Kohonen, T.** *Self-organized formation of topologically correct feature maps*, Biological Cybernetics 43(1), 59-69 pgs., 1982

11. **Fukushima, Kunihiko.** *Cognitron: A self-organizing multilayered neural network*, Biological Cybernetics, Vol.20: 121-136 pgs., 1975.

12. **Mendoza, Miguel Angel.** *¿Ciberseguridad o seguridad de la información? Aclarando la diferencia*, We live security ESET., [En línea] [Citado el: 3 de Junio de 2018.] URL: https://www.welivesecurity.com/la-es/2015/06/16/ciberseguridad-seguridad-informacion-diferencia/

13.**Estandar Internacional**, *Gobierno de la ciberseguridad*, ISO/IEC, 2007, URL: http://seguridad-de-la-informacion.blogspot.com/2007/09/iso-27001-en-castellano.html

14. **Melo, Paola.** *La Disponibilidad como el primero de los 3 ejes fundamentales de la Ciberseguridad Industrial*. [En línea] [Citado el: 3 de Junio de 2018.] URL: http://www.altadisponibilidadlogitek.com/la-disponibilidad-como-el-primero-de-los-3-ejes-fundamentales-de-la-ciberseguridad-industrial/.

15. **National security agency/central security service fort, George G., Meade Md.** *National Information Systems Security (INFOSEC) Glossary* [Citado el: 3 de Junio de 2018.] URL: http://www.dtic.mil/dtic/tr/fulltext/u2/a433929.pdf.

16. **Crido, Santiago.** *CIBERSEGURIDAD: ¿Cómo implementar una Política de Seguridad Informática? Una propuesta.* [En línea] [Citado el: 3 de Junio de 2018.] URL: https://es.linkedin.com/pulse/ciberseguridad-c%C3%B3mo-implementar-una-pol%C3%ADtica-de-crido-santiago.

17. **Axelsson, Stefan.** *Research in Intrusion-Detection Systems: A Survey.* Götenborg : s.n., 1998.

18. **UpGuard.** *Top Free Network-Based Intrusion Detection Systems (IDS) for the Enterprise.* [En línea] [Citado el: 3 de Junio de 2018.] URL: https://www.upguard.com/articles/top-free-network-based-intrusion-detection-systems-ids-for-the-enterprise

19. **Snort.** [En línea] [Citado el: 3 de Junio de 2018.] URL: https://www.snort.org/.

20. **Suricata.** *Open Source IDS / IPS / NSM engine* [En línea] [Citado el: 3 de Junio de 2018.] URL: https://suricata-ids.org/

21. **Bro.** *The Bro Network Security Monitor* [En línea] [Citado el: 3 de Junio de 2018.] URL: https://www.bro.org.

22. **OSSEC.** [En línea] [Citado el: 3 de Junio de 2018.] URL: https://www.ossec.net/.

23. **Samhain.** *The SAMHAIN file integrity / host-based intrusion detection system* [En línea] [Citado el: 3 de Junio de 2018.] URL: https://la-samhna.de/samhain/

24. **Pappas, Nicholas.** *Network IDS & IPS Deployment Strategies,* GSEC Gold Certification, 2008

25. **Baker, A. R., Beale, J., Caswell, B., & Poor, M.** *Snort 2.1 Intrusion Detection Second Edition.* Rockland, MA: Syngress Publishing, Inc, 2004

26**. Rehman, R. U.** *Intrusion Detection with Snort: Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID.* Saddle River, NJ: Prentice Hall PTR., 2003

27. **H. Wang, J. Gu, S. Wang.** *An effective intrusion detection framework based on SVM with feature augmentation.* Knowledge-Based Systems, Vol.136, Pages 130-139, ISSN 0950-7051, 2003

28. **I. Ahmad, M. Basheri, M.J. Iqbal, A. Raheem.** *Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection.* doi: 10.1109/ACCESS.2018.2841987 in IEEE Access, 2018

29. **N.Farnaaz, M.A. Jabbar.** *Random Forest Modeling for Network Intrusion Detection System,* Procedia Computer Science, Vol.89: 213-217pgs., ISSN 1877-0509, https://doi.org/10.1016/j.procs.2016.06.047, 2016

30. **Y. Y. Aung and M. M. Min.** *An analysis of random forest algorithm-based network intrusion detection system.* Kanazawa, 127-132 pgs., doi: 10.1109/SNPD.2017.8022711, 2017

31. **Aburomman, Abdulla A., Ibne Reaz, Mamun Bin.** *A novel SVM-kNN-PSO ensemble method for intrusion detection system.* Applied Soft Computing, Vol. 38: 360-372 pgs., ISSN 1568-4946, 2016

32. **Longjie Li, Yang Yu, Shenshen Bai, Jianjun Cheng, and Xiaoyun Chen.** *Towards Effective Network Intrusion Detection: A Hybrid Model Integrating Gini Index and GBDT with PSO.* Journal of Sensors, vol. 2018, https://doi.org/10.1155/2018/1578314, 2018.

33. **I. Syarif, E. Zaluska, A. Prugel-Bennett, G. Wills.** *Application of bagging, boosting and stacking to intrusion detection, in: Machine Learning and Data Mining in Pattern Recognition.* Springer, 593–602 pgs., 2012

34. **J. Ryan, M. Lin, and R. Miikkulainen.** *Intrusion Detection with Neural Networks,* in Advances in Neural Information Processing Systems 10, NIPS Conference, 943–949 pgs., 1997

35. **B. Subba, S. Biswas and S. Karmakar.** *A Neural Network Based System for Intrusion Detection and Attack Classification.* 2016 Twenty Second National Conference on Communication (NCC), doi: 10.1109/NCC.2016.7561088, 2016

36. **Ma, T., Wang, F., Cheng, J., Yu, Y., & Chen, X.** *A Hybrid Spectral Clustering and Deep Neural Network Ensemble Algorithm for Intrusion Detection in Sensor Networks.* eds. Sensors (Basel, Switzerland), 2016

37. **A. Shiravi, H. Shiravi, M. Tavallaee, A. A. Ghorbani.** *Toward developing a systematic approach to generate benchmark datasets for intrusion detection,* Computers and Security 31, 357–374 pgs., 2012

38. **S. Garcia, M. Grill, J. Stiborek, A. Zunino.** *An empirical comparison of botnet detection methods.* Computers & Security 45, 100 – 123 pgs., 2014

39. **J. Stolfo, W. Fan, W. Lee, A. Prodromidis, P. K. Chan.** *Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection,* Results from the JAM Project by Salvatore,1-15 pgs. 2000

40. **Maciá-Fernández, Gabriel & Camacho, José & Magán-Carrión, Roberto & García-Teodoro, Pedro & Therón, Roberto.** *UGR '16: A new dataset for the evaluation of cyclostationarity-based network IDSs.* Computers & Security 73, 2017

41. **Nmap.** [En línea] [Citado el: 3 de Junio de 2018.] URL: https://nmap.org/.

42. **P. Haag.** NFDUMP-NetFlow processing tools [En línea] [Citado el: 3 de Junio de 2018.] URL: http://nfdump.sourceforge.net.

43. **Villamil Torres, Jaime Alberto, & Delgado Rivera, Jesús Alberto.** *Entrenamiento de una red neuronal multicapa para la tasa de cambio euro - dólar (EUR/USD*). Ingeniería e Investigación, 27(3), 106-117 pgs**.**, 2007

44. *Basic evaluation measures from the confusion matrix.* WordPress.com [En línea] [Citado el: 3 de Junio de 2018.] URL: https://classeval.wordpress.com/introduction/basic-evaluation-measures/

45. **Pandas Pydata** [En línea] [Citado el: 15 de Junio de 2018.] URL: https://pandas.pydata.org/

46. **scikitLearn** [En línea] [Citado el: 15 de Junio de 2018.] URL: http://scikit-learn.org/

47. **Keras Documentation** [En línea] [Citado el: 15 de Junio de 2018.] URL: https://keras.io/

48. **Numpy** [En línea] [Citado el: 15 de Junio de 2018.] URL: http://www.numpy.org/

49. **Theano Documentation** [En línea] [Citado el: 15 de Junio de 2018.] URL: http://deeplearning.net/software/theano/

49. **TensorFlow** [En línea] [Citado el: 15 de Junio de 2018.] URL: https://www.tensorflow.org/

50. **Ribeiro Marco, Singh Sameer, Guestrin Carlos.** *"Why Should I Trust You?" Explaining the Predictions of Any Classifier*, 2016, URL: https://github.com/marcotcr/lime

51. **Lundberg Scott, Lee Su-In.** *A Unified Approach to Interpreting Model Predictions*. Cornell University Library, 2017 URL: https://github.com/slundberg/shap

6. Anexos

Anexo 1: Ataques UGR-16

Tipo de ataque	Clase	Etiqueta
DoS11	Firma Sintético	Dos
DoS53s	Firma Sintético	Dos
DoS53a	Firma Sintético	Dos
Scan11	Firma Sintético	Scan11
Scan44	Firma Sintético	Scan44
Botnet	Firma Sintético	Nerisbotnet
IP en Blacklist	Firma	Blacklist
Escaneo UDP	Anomalía	Anomaly-udpscan
Escaneo SSH	Anomalía	Anomaly-sshscan
SPAM	Anomalía	Anomaly-spam

Tabla 31. Tipos de ataques base de datos UGR-16

Anexo 2: Codificaciones

Codificación de la variable *protocol type*:

Valor de la etiqueta identificativa	Codificación *one-hot*
icmp	[1,0,0]
tcp	[0,1,0]
udp	[0,0,1]

Tabla 32. Codificación one-hot de la característica protocol type

Codificación de la variable *flags*:

Valor de la etiqueta identificativa	Codificación *one-hot*
....S.	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
...R..	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
.A....	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
.A...F	[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
.A..S.	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
.A..SF	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
.A.R..	[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
.A.RS.	[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
.AP...	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
.AP..F	[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
.AP.S.	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
.AP.SF	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
.APR..	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
.APRS.	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
.APRSF	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

Tabla 33. Codificación one-hot de la característica flags

Anexo 3: Descripción del modelo RNA

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                110
_________________________________________________________________
dense_2 (Dense)              (None, 40)                440
_________________________________________________________________
dense_3 (Dense)              (None, 40)                1640
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 41
=================================================================
Total params: 2,231
Trainable params: 2,231
Non-trainable params: 0
```

Figura 63. Resumen del modelo RNA de 3 capas creado.

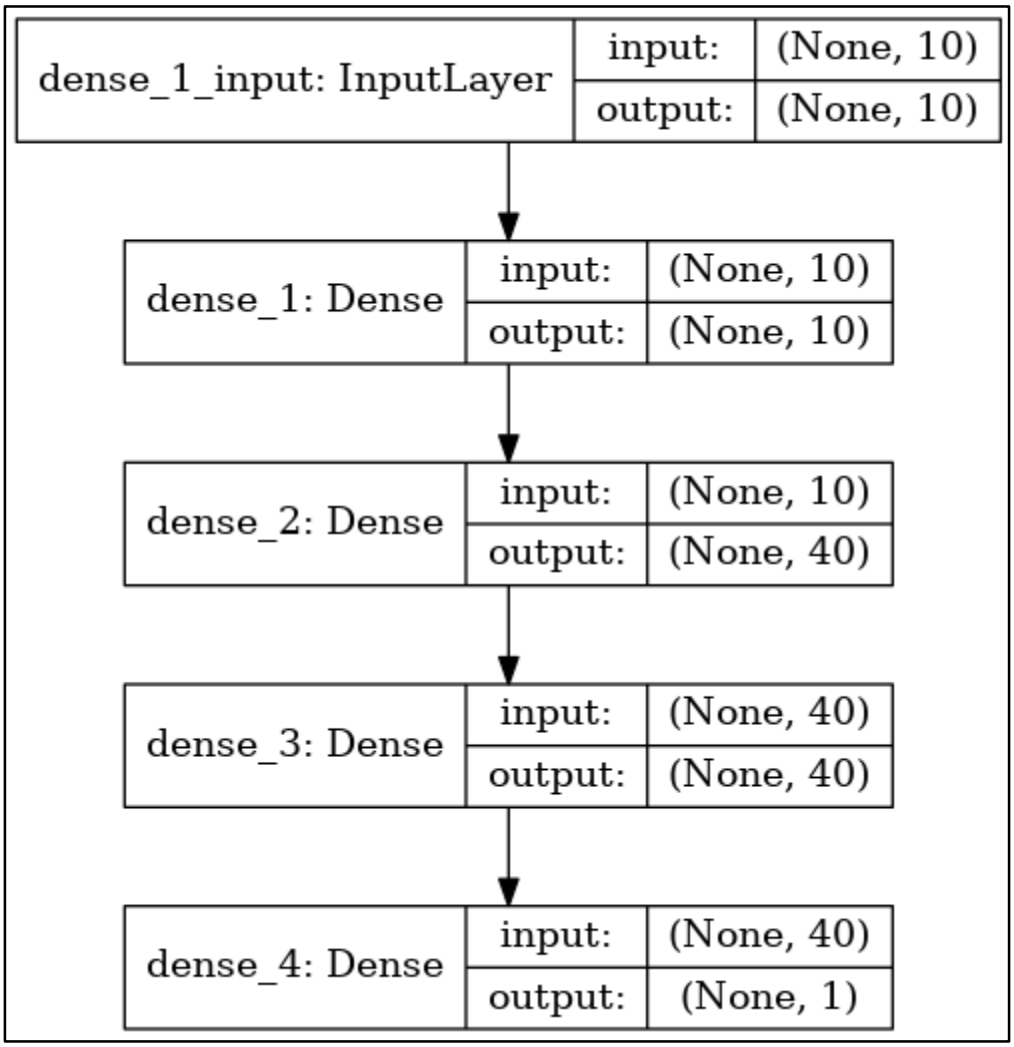

Figura 64. Representación gráfica del modelo RNA de 3 capas creado