UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE

"ALGORITMO DE INTELIGENCIA ARTIFICIAL BASADO EN LA METODOLOGÍA THREAT DETECTION PARA EL MONITOREO DE LOS REGISTROS DE EVENTOS DEL SISTEMA OPERATIVO"

FELIPE ANDRÉS CRUZ VERDUGO

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Xavier Bonnaire
Profesor Correferente: Elizabeth Montero

Agosto - 2023

DEDICATORIA

El presente trabajo está dedicado a Dios y mi familia, que me han apoyado en cada etapa para llegar a estas instancias, y los amigos que hice en el transcurso de vida universitaria.

AGRADECIMIENTOS

Agradezco a Dios, y mis padres Francisco y Johanna, que me entregaron todas las herramientas para culminar esta etapa de mi vida, que me han apoyado y sostenido durante el transcurso de este periodo universitario. A mis hermanas, Catalina y Javiera por su cariño y compañía.

A mis compañeros en general, a aquellos que en el camino se convirtieron en amigos, con quienes compartí risas, frustraciones, desvelos, partidos de fútbol y más.

Finalmente, a todos los profesores que tuve durante la universidad, ya que cada uno aportó su grano de arena a lo que son mis conocimientos profesionales.
Gracias totales.

RESUMEN

Resumen—Frente al aumento de la ciberdelincuencia, y considerando los ataques dirigidos al sistema operativo, la mayoría de las empresas contratan servicios externos para monitorear los registros de eventos del sistema, también llamados *logs*. Estos servicios tienen un alto costo económico, además de requerir de recurso humano para su implementación. Por lo anterior, se hace necesario un algoritmo al alcance de todas las empresas, que además elimine el factor humano. En el presente informe se propone un algoritmo implementado con inteligencia artificial basado en la metodología Threat Detection para el monitoreo de registro de eventos del sistema operativo de manera automatizada. Obteniendo como resultado una aplicación *open source* capaz de predecir la presencia y categoría de eventos maliciosos con la opción de descargar un informe detallado de los resultados obtenidos. Adicionalmente, el *software* es validado por su funcionamiento y por métricas propias de la inteligencia artificial como la exactitud y precisión, obteniendo valores positivos. Finalmente, la propuesta de solución es comparada con herramientas disponibles en el mercado, tanto *open source* como de pago, con el propósito de conocer sus ventajas.

Palabras Clave—Ciberseguridad; Inteligencia Artificial; Threat Detection; Registro de eventos; Sistema Operativo.

ABSTRACT

Abstract—Faced with the increase in cybercrime, and considering attacks directed at the operating system, most companies hire external services to monitor system events records, also called logs. These services have a high economic cost, in addition to requiring human resources for their implementation. Due to the above, an algorithm is necessary within the reach of all companies, which also eliminates the human factor. This report proposes an algorithm with artificial intelligence based on the Threat Detection methodology for automated monitoring of the operating system event log. Obtaining as a result an open-source application capable of predicting the presence and category of malicious events with the option of downloading a detailed report of the results obtained. Additionally, the software is validated by its operation and by artificial intelligence metrics, as accuracy and precision, obtaining positive results. Finally, the solution proposal is compared with tools available in the market, both open source and paid, to knows its advantages.

Keywords—Cibersecurity; Artificial Intelligence; Threat Detection; Event log; Operative System.

GLOSARIO

AMD: Advanced Micro Devices.

AWS: Amazon Web Services.

CMD: Command Prompt.

EE. UU: Estados Unidos.

GB: Gigabyte.

GHz: Gigahercios.

HDD: Hard Drive Disk.

HTTP: Hypertex Transfer Prococol.

HTTPS: Hypertex Transfer Prococol Secure.

IA: Inteligencia Artificial.

IBM: International Business Machines. Compañía privada estadounidense que provee soluciones de hardware y software.

PC: Personal Computer.

PII: Información de Identificación Personal.

TB: Terabyte.

TI: Tecnologías de la Información.

SEM: Administración de Eventos de Seguridad

SIM: Administración de Información de Seguridad.

SIEM: Información de Seguridad y Gestión de Eventos.

SO: Sistema Operativo.

SOC: Centro de Operaciones de Seguridad.

SSD: Solid State Drive.

SQL: Structured Query Languague.

USD: Dólar estadounidense.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Anexo al avance tecnológico, la relevancia de la ciberseguridad[1] va en aumento. Hace algunos años no suponía un riesgo navegar por internet utilizando el protocolo de comunicación HTTP, el cual permite la transferencia de información en Internet, y es que cuando fue desarrollado no se consideró la seguridad, por la casi ausencia de ciberdelincuentes[2] y conceptos cómo ciberataque[3] y ciberseguridad. En la actualidad, realizamos diversas acciones en la web que exponen nuestra información privada, tales como comprar, agendar citas médicas, gestiones bancarias, entre otras. Frente a esto surgió el protocolo HTTPS, siendo la seguridad la principal diferencia, ya que este protocolo impide que otros usuarios puedan interceptar la información confidencial que se transfiere entre el cliente y el servidor web a través de Internet.

Lo anterior, es solo un ejemplo de la relevancia que tiene la disciplina de la seguridad informática en nuestra vida, más conocida cómo ciberseguridad. Esto debido a que la ciberdelincuencia se ha convertido en una de las mayores amenazas para la seguridad y privacidad en el mundo digital. Se trata de un "negocio" altamente rentable que requiere de relativamente pocos recursos, adolece de regulación a nivel internacional y se beneficia de una impunidad que ronda el 95 %. En este contexto, la empresa tiene una gran responsabilidad en la prevención y protección contra el cibercrimen, ya que tiene el deber de garantizar la seguridad de los datos de sus clientes y empleados. Esto implica la adopción de un enfoque proactivo en la protección de la información. (Juan, 2023)

Por tanto, el escenario en que nos situamos es uno donde las empresas son el actor principal, siendo conscientes del alcance que tienen y de todo el recurso humano que manejan, sumado a sus clientes, siendo cada individuo una fuente de información. Junto a la creciente transformación digital[4] en las empresas, la superficie de ataque[5] de estas se incrementa, reconociendo tres categorías: la superficie de ataque de ingeniería social, la física y la digital, siendo esta última nuestro foco, la cual expone potencialmente la infraestructura local y *cloud* de la organización a cualquier ciberdelincuente con conexión a internet.

De acuerdo al informe *The State Of Attack Surface Management 2022*, realizado por Randori, una filial de IBM, el cuál consistió en analizar 398 tomadores de decisiones de TI y

---

[1] Área relacionada con la informática y la telemática que se enfoca en la protección de la infraestructura computacional y todo lo vinculado con la misma.
[2] Actividad delictiva que se lleva a cabo a través de Internet.
[3] Intento no deseado de robar, exponer, alterar, inhabilidad o destruir información mediante el acceso no autorizado de los sistemas.
[4] Cambio asociado con la aplicación de tecnologías digitales en todos los aspectos de la sociedad humana.
[5] La suma de las vulnerabilidades de una organización a posibles ciberataques.

seguridad en los Estados Unidos y Canadá para averiguar cómo administran sus superficies de ataque, cómo adaptan sus programas para las amenazas en evolución, y los mayores obstáculos para una gestión eficaz de la superficie de ataque; se llegó a los resultados de que el 67% de las organizaciones han visto crecer sus superficies de ataque en los últimos dos años. (Randori, 2022) Sumado a esto, Gartner, empresa consultora y de investigación de las tecnologías de la información de Estados Unidos, estableció la expansión de la superficie de ataque como tendencia número 1 en seguridad y gestión de riesgos en 2022. (Rimol, 2022)

Los ciberdelincuentes se enfocan en obtener la información de identificación personal (PII) de los clientes: nombres, direcciones, números de identificación nacional e información de tarjetas de crédito para posteriormente vender estos registros en mercados digitales clandestinos. Los PII comprometidos a menudo generan la pérdida de la confianza del cliente, multas regulatorias e incluso acciones legales (IBM, 2020).

De acuerdo con el estudio realizado por IBM en el año 2022, los vectores de ataque[6] más comunes en la superficie de ataque digital de una organización son:

- Contraseñas débiles.
- Configuración incorrecta.
- Activos con acceso a Internet.
- Bases de datos y directorios compartidos.
- Dispositivos, datos o aplicaciones desactualizados u obsoletos.
- TI invisible.
- **Vulnerabilidades de *software*[7], sistema operativo[8] (SO) y *firmware*[9].**

El eje central del presente documento se basa en los ataques de infraestructura, específicamente en aquellos que buscan vulnerar los servicios que ofrece el sistema operativo de un dispositivo. En el sector TI los registros de eventos, también llamados *logs*, hacen referencia a los archivos de texto en los que se incluyen de forma cronológica los acontecimientos como cambios, actualizaciones y demás que han ocurrido dentro de un sistema informático, como puede ser un servidor, una aplicación o un programa, así como la serie de modificaciones que estos han generado. Haciendo uso de estos es que se ha presentado cómo solución a este tipo de ciberataques, el monitoreo de *logs*, para detectar alteraciones indebidas en el sistema, en este caso en el sistema operativo.

Las herramientas de monitoreo de *logs* son fundamentales para la seguridad informática de cualquier entidad. De hecho, se han convertido en una necesidad que ha propiciado el

[6] Elementos clave para la prevención y defensa de una red empresarial
[7] Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.
[8] Programa o conjunto de programas que realizan funciones básicas y permiten el desarrollo de otros.
[9] Programa informático que establece la lógica de más bajo nivel de un dispositivo.

desarrollo de otras herramientas que sirven para la gestión de eventos y seguridad de información, abreviado cómo SEM (E-DEA NETWORKS, 2019). Yendo un poco más lejos, para evidenciar el alcance de estas herramientas, encontramos las de tipo SIEM, que combinan las SEM y SIM para entregar una solución de seguridad que ayuda a las organizaciones a detectar y analizar amenazas y responder a ellas antes de que afecten a las operaciones del negocio. (Microsoft, 2023)

Retomando nuestra propuesta de solución, debido a la amplia variedad que existe a la hora de utilizar un dispositivo tecnológico y el SO que contiene, este trabajo se centra en los PC que utilizan *Microsoft Windows* como sistema operativo, esto debido a que es el más utilizado a nivel mundial, tanto de manera personal cómo empresarial, cómo consecuencia de venir instalado por defecto en la mayoría de los computadores nuevos y por su preferencia por parte de los usuarios por su practicidad.

A continuación, se muestra una tabla comparativa de los diferentes sistemas operativos utilizados mundialmente durante el año 2022:

Tabla 1: Tabla porcentual del uso de los sistemas operativos a nivel mundial.
Fuente: Elaboración propia a partir de la información de Statcounter GlobalStats.

Sistema Operativo	Porcentaje
Windows	75.34 %
OS X	14.66 %
Unknown	4.78 %
Linux	2.93 %
Chrome OS	2.28 %
Other	0.01 %

Cómo se puede evidenciar, *Windows* es el sistema operativo predilecto por los usuarios y empresas, con un uso del 75.34 % a nivel mundial, muy por encima de los demás.

En la actualidad cada vez es más frecuente visualizar que las empresas externalizan servicios, ya que dentro de los beneficios se encuentra la flexibilidad que le brinda a la misma delegar ciertos procesos o servicios que no son su valor principal, con el objetivo de ser más productivo en el correspondiente. La ciberseguridad no es la excepción, y es que las empresas contratan a otras que se dedican a la consultoría u ofrecer *softwares* orientados a detectar vulnerabilidades dentro de ellas, siendo uno de los servicios

propuestos el monitoreo de *logs*, el cual se integra en la amplia variedad de los servicios ofrecidos por las plataformas SOC, las cuales se encargan de analizar la actividad de un sistema informático con el fin de detectar cualquier tipo de actividad sospechosa. El SOC está compuesto por un equipo técnico y humano que utilizan herramientas de software específicas para formar el núcleo central para la defensa de la seguridad informática de la empresa. (Ambit, 2020)

Frente al contexto planteado, los problemas que se desprenden son que cómo consecuencia de la creciente superficie de ataque en las empresas, éstas tienen la necesidad de contratar servicios externos para proteger y/o prevenir ciberataques, lo que conlleva altos costos de implementación, considerando además que la mayoría de las soluciones basadas en el monitoreo de *logs* no están automatizadas, lo cual añade un costo de recurso humano. Según un estudio realizado por IBM, en el año 2020, el costo promedio de una brecha de seguridad de datos fue de USD 3.86 millones a nivel mundial y USD 8.64 millones en los Estados Unidos. Estos costos incluyen descubrir y responder a la brecha de seguridad, el costo del tiempo de inactividad y la pérdida de ingresos, así como el daño a la reputación y marca de una empresa a largo plazo. (IBM, 2020)

La propuesta de solución que se expone a continuación se basa en la creación de un algoritmo de inteligencia artificial, basado en la metodología de *Threat Detection* para el monitoreo los *logs* al alcance de los usuarios y empresas.

Funcionamiento

A continuación, se presenta una primera noción de la solución propuesta:

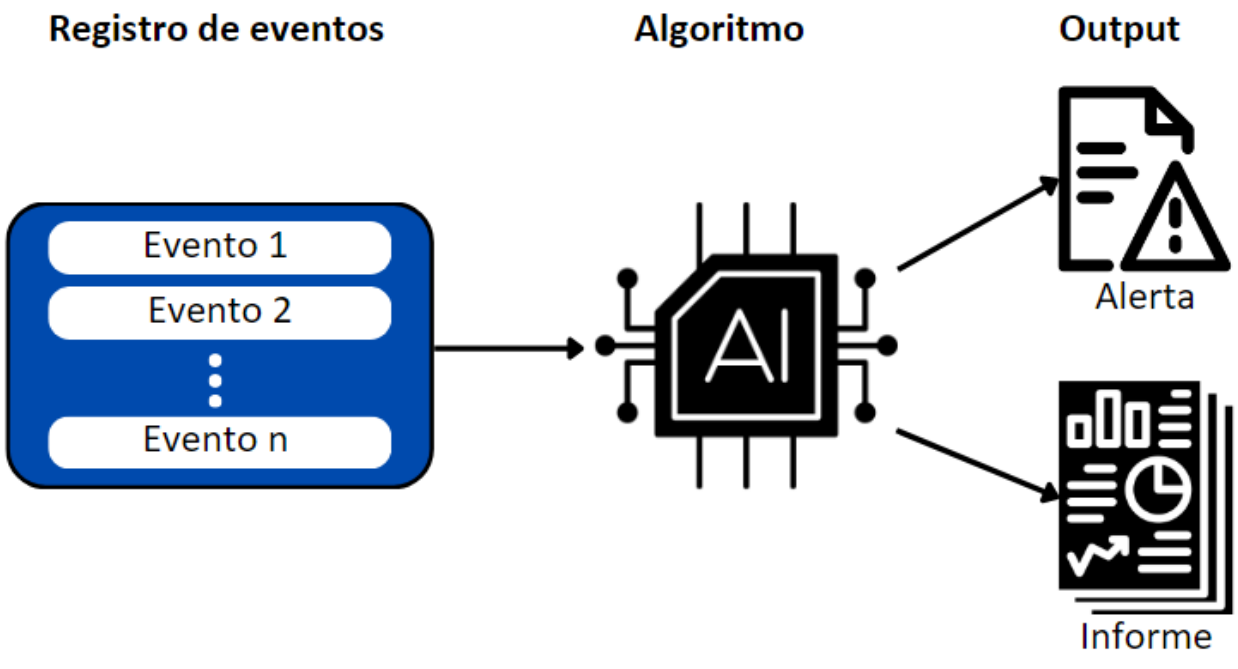


Figura 1: Propuesta de solución.
Fuente: Elaboración propia.

El mayor desafío en la implementación de mi propuesta se basa en la creación del algoritmo con técnicas de inteligencia artificial y métodos de aprendizaje automático. Dentro de estas considero utilizar *Machine Learning*[10], guiándome por investigaciones previas, cómo la desarrollada por Joseph Arboleda y Leslie Medina, quiénes exponen que la capacidad de predicción de los algoritmos de *Machine Learning* puede ser de gran ayuda para las amenazas de seguridad que han ido en aumento a pasos agigantados en los últimos años, pues permite detectar comportamientos sospechosos (Arboleda Díaz & Medina Ruiz, 2017). En segundo lugar, y cómo mayor fuente de información considero apoyarme en el trabajo de magister de José Manuel Rodríguez, titulado *Aplicación de Técnicas de Machine Learning a la Detección de Ataques*, ya que implementa *Machine Learning* en un paso a paso con el lenguaje de programación Python, sabiendo que este es el mejor para esta área de la informática (KeepCoding, 2022), y siendo de gran afinidad con mis conocimientos.

La estrategia empleada para desarrollar la memoria consiste en una la estrategia gradual, lo que significa que en un paso a paso se van desarrollando y alcanzando los objetivos planteados en el siguiente capítulo. Dicha estrategia se compone de las siguientes fases:

Formación e investigación

Esta fase, a pesar de no parecer compleja, es crítica, debido a que las bases de lo investigado sustentan las decisiones tomadas a futuro, y el cómo se implementará la solución propuesta. La formación e investigación abarcan el perfeccionamiento en el manejo de conceptos y de las herramientas a utilizar.

Desarrollo de la solución

Una vez cimentadas las bases del conocimiento, se procede a implementar la solución propuesta, incluyendo labores como la obtención de los requisitos, la creación del código y la validación de la solución.

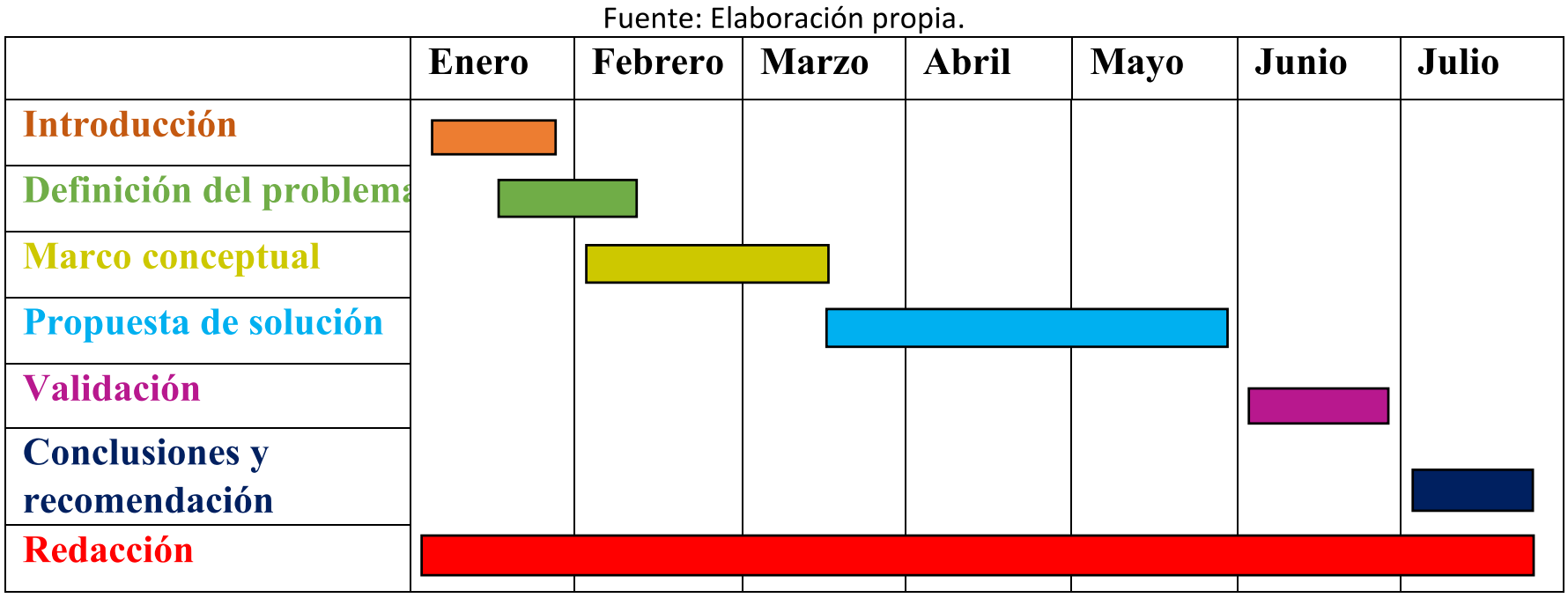
Documentación

En paralelo que se van llevando a cabo las dos fases anteriores, se va redactando el informe, sin embargo, una vez completa la propuesta de solución se da lugar a una revisión exhaustiva de la documentación para precisar en los detalles y corregir cualquier idea que haya variado en el transcurso de la implementación.

[10] Forma de la IA que permite a un sistema aprender de los datos en lugar de aprender mediante la programación explícita.

Planificación del trabajo

Tabla 2: Planificación del trabajo.
Fuente: Elaboración propia.



Como se expresó con anterioridad la estrategia empleada es de tipo gradual, por lo que se puede apreciar que a medida que avanzan los meses se van desarrollando ordenadamente los capítulos de la memoria, explicitando además la redacción de la misma durante todo el periodo.

Breve descripción de los otros capítulos de la memoria

Finalmente, cabe mencionar que el presente documento se estructura en cinco grandes capítulos, siendo el primero de ellos la definición del problema que integra el objetivo general y objetivos específicos. En el segundo capítulo se encuentra el marco conceptual, en dónde se exponen los conceptos técnicos, las metodologías, herramientas, entre otros, que están involucrados en la solución propuesta. En el capítulo tres se desarrolla la propuesta de solución, siendo esta un algoritmo de inteligencia artificial basado en la metodología *Threat Detection* para el monitoreo de *logs* de manera automatizada[11]. En la cuarta arista se valida en base a dos enfoques, el primero siendo la construcción y ejecución del código, es decir que la implementación será testeada a partir de *logs* maliciosos debiendo estos ser identificados por el algoritmo; El segundo enfoque viene a ser la validación por parte de la disciplina de la inteligencia artificial a partir de métricas definidas por la misma. Por último, en el capítulo final se encuentran las conclusiones obtenidas del trabajo realizado y el trabajo a futuro que se podría desarrollar en torno a esta temática.

[11] Aplicación de máquinas o de procedimientos automáticos en la realización de un proceso o en una industria.

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA

El cómo enfrentar la ciberdelincuencia[12] es un tema que ha tomado relevancia en los últimos años, y es que durante estos, el auge en los crímenes digitales o ciberataques ha puesto en jaque a los usuarios de internet, sitios web, empresas y corporaciones, generando pérdidas por miles de millones de dólares al año con tendencia al alza, debido a que cada día los cibercriminales mejoran o evolucionan sus métodos para lograr acceder a la información personal y confidencial de las víctimas y con ello poder obtener un lucro económico. Debido a esto es importante estar al tanto de las amenazas que se pueden conseguir día a día en la web o con solo encender el computador, sin embargo, no basta con ello, se debe tener un conocimiento mínimo de cómo contrarrestar dichas amenazas y tener las herramientas mínimas para no ser víctimas de robo de información, fraudes o estafas. Por lo antes expuesto es que la seguridad informática toma tanta importancia, no solo desde el punto de vista de resguardo de información, sino también económico, la inversión de tiempo para el adiestramiento y programas de protección son la única manera de hacerle frente a los cibercriminales y evitar mayores pérdidas monetarias en el futuro. (Gamboa Suarez, 2020)

Considerando cómo protagonistas a las empresas, se tiene que estas presentan una amplia superficie de ataque, siendo uno de los vectores de ataque las vulnerabilidades del sistema operativo mediante los servicios que este ofrece. Frente a esta situación, las empresas contratan las prestaciones de consultoría en ciberseguridad, la cual consiste en la implementación de diferentes métodos para analizar y mejorar el desempeño de la tecnología de la información de una empresa (Cárdenas, 2022).

Actualmente la revisión de estos ficheros *logs* viene a ser un proceso exhaustivo que implica mucho esfuerzo tanto manualmente o a través de un *software*, causando detención en los servicios brindados, además de generar retrasos y pérdidas de recursos al analizar, buscar e identificar qué tipo evento se suscitó y en qué lugar de la estructura se generó, dado el volumen y la cantidad de registros *logs* que se generan diariamente. Ante esto surge la necesidad de las empresas de disponer de un sistema de análisis de la actividad de los sistemas, aplicaciones y equipos en general, cuya información se registra en los ficheros de *logs* que estos generan (Figueroa Fernández, 2021).

[12] Actividad delictiva que se lleva a cabo a través de Internet.

Por otro lado, la ciberseguridad se puede considerar costosa, ya que cómo indica la revista Entrepreneur, un presupuesto de ciberseguridad saludable debería representar del 9 al 14% del presupuesto anual general de su departamento de Tecnología e Información (Chou, 2021). Sumado a esto, cómo indica Panda Security, empresa líder en ciberseguridad, la falta de presupuesto de las organizaciones, de tecnologías, de procesos y, sobre todo, de un equipo de expertos para hacerlo desde cero, hace que sea imposible para la mayoría construir y evolucionar su defensa tan rápidamente como lo hace el cibercrimen (PandaSecurity, 2018).

La importancia de dar solución a la problemática planteada se respalda en que la complejidad del sistema de seguridad, creada por tecnologías dispares, y la falta de experiencia interna, puede aumentar los costos de una brecha de seguridad. Pero las organizaciones con una estrategia integral de ciberseguridad, regida por mejores prácticas y automatizada con analítica avanzada, inteligencia artificial (IA) y *Machine Learning*, pueden combatir las ciberamenazas[13] de manera más efectiva y reducir el ciclo de vida y el impacto de las brechas cuando ocurren. (IBM, 2020)

Por otro lado, la presencia de una herramienta que permita el monitoreo de *logs* apoya no solo a las empresas, que cómo mencioné anteriormente se relacionan con plataformas SOC, sino también a entidades que realizan análisis forense una vez que se ha perpetrado un delito, dando soporte en la investigación del causante o medio por el cual el ciberdelincuente accedió a la red. Esto se debe a que los registros de eventos permiten recopilar información acerca de los parámetros del sistema operativo, el software instalado, los medios de almacenamiento que se conectaron, los dispositivos instalados, fecha de creación, modificación y acceso a carpetas y/o archivos.

Finalmente, dentro de las soluciones semejantes que existen en el mercado, se encuentran las de tipo *open source* y las de pago. Se realizarán comparativas con ambas categorías, la primera con foco en el funcionamiento, y la segunda con el objetivo de evidenciar los altos costos y la dificultad de acceso a estas para las microempresas. Dentro de las soluciones similares gratuitas se encuentra Zircolite, disponible en la plataforma GitHub, y en las herramientas de pago se considerará encontrando el *software* OpManager de ManageEngine, empresa con más de 20 años de experiencia en el sector TI.

[13] Aquellas actividades “malignas” que tienen lugar en un entorno digital, cuyo objetivo es comprometer la seguridad de un sistema de información alterando su disponibilidad, integridad o confidencialidad.

1.1 OBETIVO GENERAL

Identificar posibles ciberataques al sistema operativo mediante la implementación de una metodología de *Threat Detection* basada en un algoritmo de inteligencia artificial que analiza los registros de eventos de manera automatizada.

1.2 OBJETIVOS ESPECÍFICOS

- Presentar y analizar un problema de ciberseguridad.
- Diseñar un algoritmo basado en técnicas de inteligencia artificial.
- Generar alertas y reportes a partir del análisis de *logs.*
- Evaluar el funcionamiento y desempeño del algoritmo.
- Evaluar el algoritmo mediante métricas de inteligencia artificial.

1.3 ÁRBOL DEL PROBLEMA

A continuación, se presenta el árbol del problema de acuerdo con la siguiente estructura:

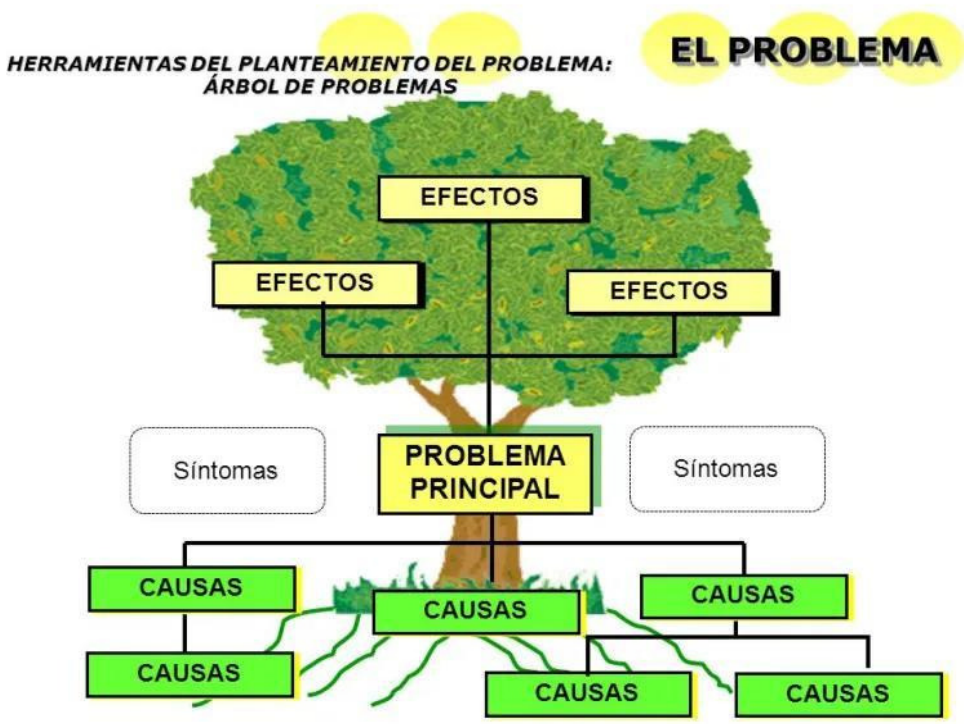


Figura 2: Árbol de problemas.
Fuente: Richard Cárdenas, Proyecto Educativo, 2016.

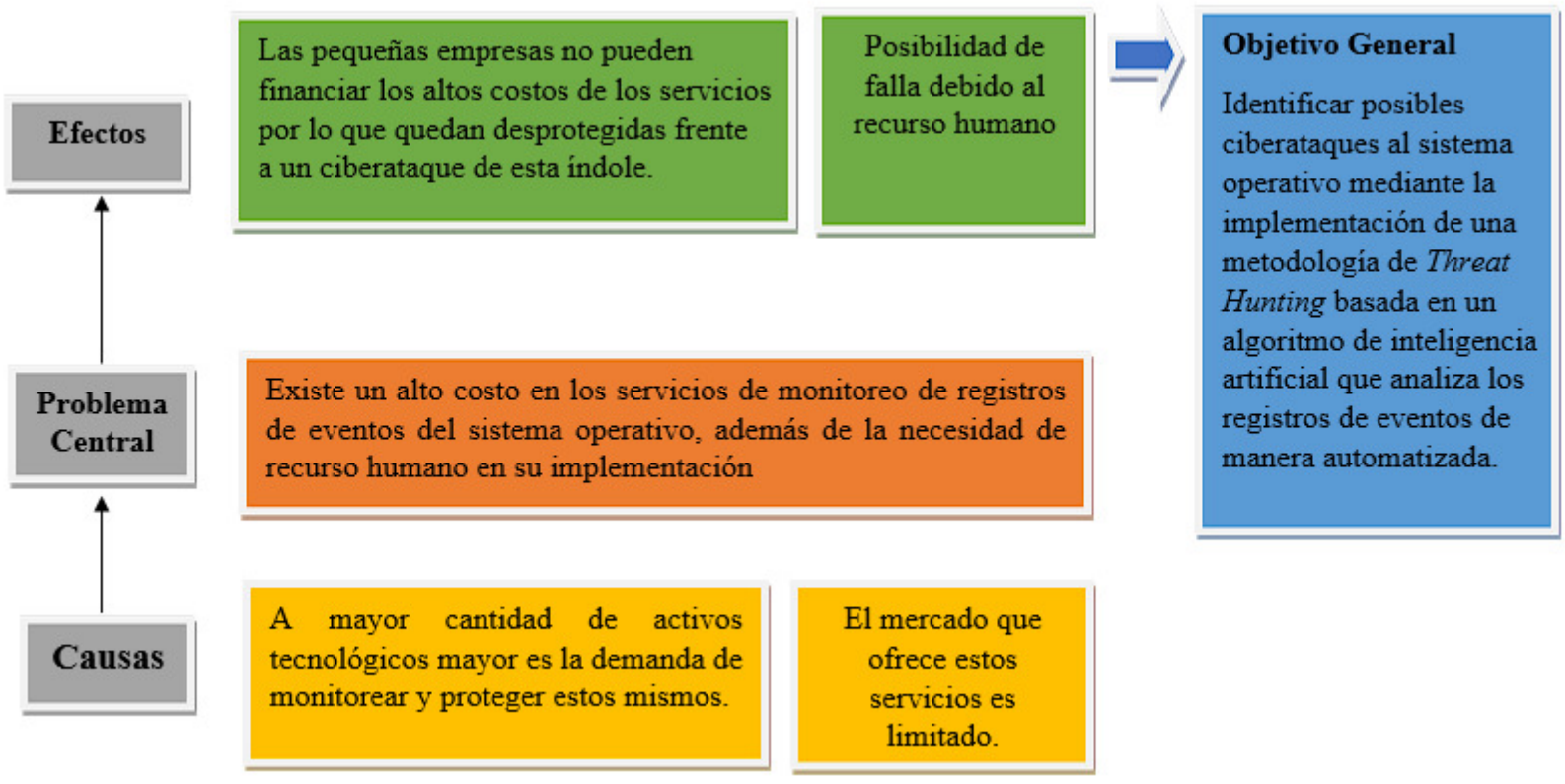


Figura 3: Árbol del problema.
Fuente: Elaboración propia.

Como se puede apreciar se evidencian dos causantes que dan lugar a nuestro problema central, justificando la necesidad de crear un algoritmo de inteligencia artificial que monitoree los registros de eventos del sistema operativo en los computadores de las empresas y que a la vez sea accesible económicamente para estas, con el fin de mitigar los efectos expuestos, como su desprotección frente a ciberataques y la posibilidad de error al revisar los *logs* de manera manual. De este modo se genera el objetivo general que me motiva al desarrollo de la solución propuesta.

CAPÍTULO 2: MARCO CONCEPTUAL

2.1 CONCEPTOS TÉCNICOS

2.1.1 CIBERSEGURIDAD

La ciberseguridad es la práctica de proteger los sistemas importantes y la información confidencial de los ataques digitales. También conocida como seguridad de la tecnología de la información, las medidas de ciberseguridad o seguridad cibernética están diseñadas para combatir las amenazas contra sistemas en red y aplicaciones, ya sea que esas amenazas se originen dentro o fuera de una organización (IBM, 2020). Por lo general, estos ciberataques apuntan a acceder, modificar o destruir la información confidencial; Extorsionar a los usuarios o interrumpir la continuidad del negocio (CISCO, 2022).

Tipos de ciberamenazas

- Delito cibernético: Incluye agentes individuales o grupos que atacan a los sistemas para obtener beneficios financieros o causar interrupciones.
- Ciberataques: A menudo involucran la recopilación de información con fines políticos.
- Ciberterrorismo: Tiene como objetivo debilitar los sistemas electrónicos para causar pánico o temor.

Métodos utilizados para amenazar la ciberseguridad (Kaspersky, 2022)

- *Malware*: Se refiere al software malicioso. Es una de las ciberamenazas más comunes, siendo un software que un cibercriminal o un hacker ha creado para interrumpir o dañar el equipo de un usuario legítimo. Con frecuencia propagado a través de un archivo adjunto de correo electrónico no solicitado o de una descarga de apariencia legítima.
- Inyección de código SQL: Tipo de ciberataque utilizado para tomar el control y robar datos de una base de datos.
- *Phishing*: Situación en dónde los cibercriminales atacan a sus víctimas con correos electrónicos que parecen ser de una empresa legítima que solicita información confidencial, como datos bancarios, entre otros.
- Ataque de tipo *Man-in-the-middle*: Tipo de ciberamenaza en la que un cibercriminal intercepta la comunicación entre dos individuos para robar datos.
- Ataque de denegación de servicio: Sucede cuando los cibercriminales impiden que un sistema informático satisfaga solicitudes legítimas sobrecargando las redes y los servidores con tráfico. Esto hace que el sistema sea inutilizable e impide que una organización realice funciones vitales.

2.1.2 INTELIGENCIA ARTIFICIAL

La inteligencia artificial es una disciplina que desarrolla la informática, siendo un campo amplio de investigación, por lo que se tienen diversas definiciones. De manera sencilla, se puede entender como la capacidad de las máquinas para usar algoritmos[14], aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano. Sin embargo, a diferencia de las personas, los dispositivos basados en IA no necesitan descansar y pueden analizar grandes volúmenes de información a la vez. Asimismo, la proporción de errores es significativamente menor en las máquinas que realizan las mismas tareas que sus contrapartes humanas (Rouhiainen, 2018).

Por otro lado, la Comisión Europea agrega valor a la definición presentada anteriormente, añadiendo que la IA permite que los sistemas tecnológicos perciban su entorno, se relacionen con él, resuelvan problemas y actúen con un fin específico. La máquina recibe datos (ya preparados o recopilados a través de sus propios sensores, por ejemplo, una cámara), los procesa y responde a ellos. Los sistemas de IA son capaces de adaptar su comportamiento en cierta medida, analizar los efectos de acciones previas y de trabajar de manera autónoma. (Parlamento Europeo, 2020)

Machine Learning

Es el aprendizaje automático o aprendizaje de máquinas, siendo un subcampo de las ciencias de la computación y una rama de la inteligencia artificial cuya finalidad es desarrollar técnicas que permitan a las computadoras aprender, convirtiéndose en un pilar fundamental para el trato de datos a gran escala. (Hinestroza Ramírez, 2018)

Machine Learning es una forma de la IA que permite a un sistema aprender de los datos en lugar de aprender mediante la programación explícita. Sin embargo, implementar *Machine Learning* no es un proceso sencillo, considerando que conforme el algoritmo ingiere datos de entrenamiento, es posible producir modelos más precisos basados en dichos datos. Un modelo de *Machine Learning* es la salida de información que se genera cuando entrena el algoritmo con datos. Después del entrenamiento, al proporcionar un modelo con una entrada, se le dará una salida. Por ejemplo, un algoritmo predictivo creará un modelo predictivo. A continuación, cuando proporcione el modelo predictivo con datos, recibirá un pronóstico basado en los datos que entrenaron al modelo. (IBM, 2022)

Dentro de la Inteligencia Artificial y específicamente en la rama de *Machine Learning* se reconocen diversos tipos de aprendizaje, siendo los más destacables los de:

---

[14] Cualquier procedimiento computacional bien definido que parte de un estado inicial y un valor o un conjunto de valores de entrada, a los cuales se les aplica una secuencia de pasos computacionales finitos, produciendo una salida o solución.

Optimización

Tipo de aprendizaje de Machine Learning orientado a encontrar la mejor solución incluso cuando existen restricciones complejas. Por ejemplo, la optimización podría responder a la pregunta "¿Cuál es la ruta óptima para seguir o la asignación de recursos o el programa de mantenimiento del equipo?" La optimización utiliza algoritmos genéticos, que se basan en la teoría de la evolución de Darwin. (TIBCO, 2023)

Aprendizaje No Supervisado

Tipo de *Machine Learning* que se utiliza para identificar nuevos patrones y detectar anomalías. Los datos que se introducen en los algoritmos de aprendizaje no supervisados no están etiquetados. El algoritmo (o modelos) intentan dar sentido a los datos por sí mismos mediante la búsqueda de características y patrones. El aprendizaje no supervisado utiliza el agrupamiento, los componentes principales, las redes neuronales y las máquinas de vectores de soporte. (TIBCO, 2023)

Este tipo de *Machine Learning* ha dado lugar a la creación de un nuevo subconjunto dentro de la Inteligencia Artificial, denominado *Deep Learning*, el cual para muchos es posiblemente el futuro del aprendizaje automático. En este paradigma los algoritmos son capaces de aprender sin intervención humana previa, sacando de ellos mismos las conclusiones acerca de la semántica embebida de los datos. En una frase breve se puede entender como un acercamiento más íntimo al modo de funcionamiento del sistema nervioso humano.

Aprendizaje Supervisado

Método de análisis de datos que utiliza algoritmos que aprenden iterativamente de los datos para permitir que los ordenadores encuentren información escondida sin tener que programar de manera explícita dónde buscar.

El aprendizaje supervisado resuelve problemas conocidos y utiliza un conjunto de datos etiquetados para entrenar un algoritmo para realizar tareas específicas. Utiliza modelos para predecir resultados conocidos como “¿Cuál es el color de la imagen?” “¿Cuántas personas hay en la imagen?” “¿Cuáles son los factores determinantes para el fraude o los defectos del producto?” etc. Por ejemplo, un proceso de aprendizaje supervisado podría consistir en clasificar vehículos de dos y cuatro ruedas a partir de sus imágenes. Los datos de entrenamiento tendrían que estar correctamente etiquetados para identificar si un vehículo es de dos o cuatro ruedas. El aprendizaje supervisado permite que los algoritmos “aprendan” de datos históricos de entrenamiento y los apliquen a entradas desconocidas para obtener la salida correcta. Para funcionar, el aprendizaje supervisado utiliza árboles de decisión, bosques aleatorios y *Gradient Boosting Machine* (TIBCO, 2023).

En la siguiente imagen se puede observar el funcionamiento del aprendizaje supervisado, recibiendo como *inputs*[15] los datos ya sea en forma estructurada o no estructurada para generar una matriz de etiquetado que dará lugar a las predicciones, es decir, que en un entorno real cuando se ingrese un *input* se esperará que el *output*[16] sea el reconocimiento correcto del dato entregado a partir de entrenamiento previo.

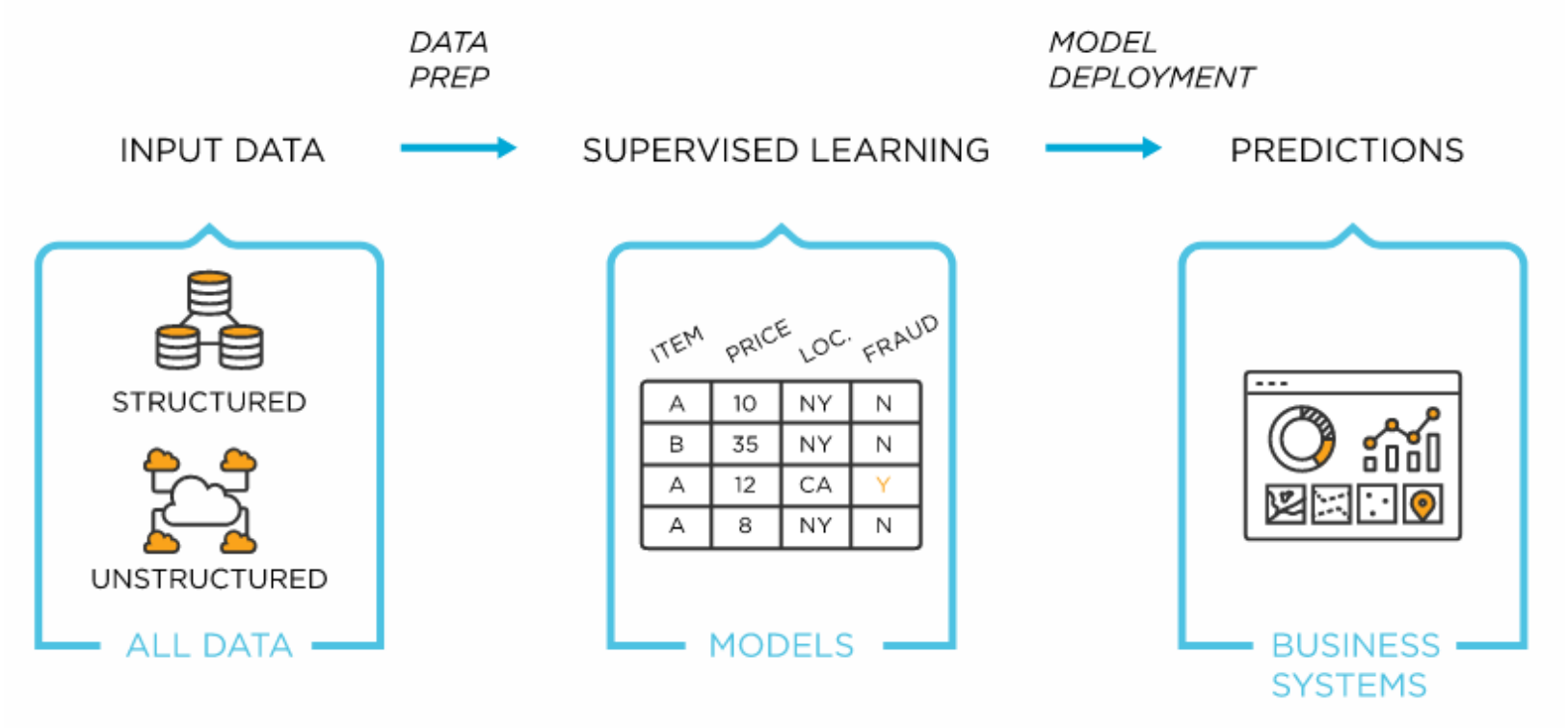


Figura 4: Esquema de Aprendizaje Supervisado en Machine Learning.
Fuente: TIBCO.

A continuación, se puede apreciar de forma gráfica la diferencia entre el aprendizaje supervisado y el aprendizaje no supervisado que ya hemos definido como *Deep Learning.*

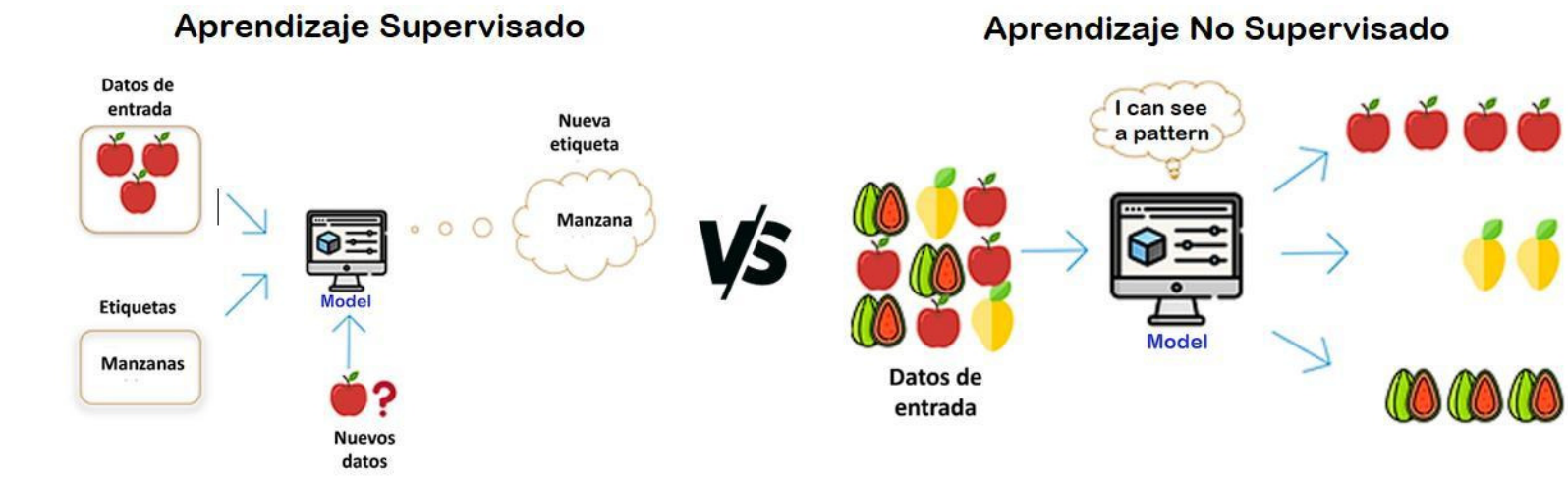


Figura 5: *Machine Learning vs Deep Learning*.
Fuente: Elaboración propia a partir de imágenes de Keep Coding.

[15] En informática se entiende como la entrada de datos para el sistema.
[16] Salida de datos como resultado proveniente del proceso del sistema.

Los tipos de aprendizaje vistos provienen del paradigma *Pattern Recognition.*

PATTERN RECOGNITION

El reconocimiento de patrones es un proceso que consiste en utilizar algoritmos informáticos para clasificar datos de entrada en objetos, clases o categorías, en base a sus características principales o elementos constantes. El reconocimiento de patrones tiene aplicación en visión artificial, segmentación de imágenes, detección de objetos, procesamiento de datos de radar, reconocimiento de voz, clasificación de texto y mucho más. (MathWorks, 2023)

El término reconocimiento de patrones se hizo popular entre las décadas de 1970 y 1980, pero no sería hasta 1990 que muchos se dieron cuenta de que había una forma más efectiva de crear algoritmos de reconocimiento de patrones, en particular reemplazando a los investigadores con probabilidad y estadística. Este paradigma condujo a la creación del *Machine Learning*.

Existen dos tipos principales de aprendizaje supervisado, correspondientes a:

APRENDIZAJE SUPERVISADO DE REGRESIÓN

Método en el que se entrena a un algoritmo para predecir una salida a partir de un rango continuo de valores posibles. Ejemplo, los datos de entrenamiento inmobiliario tomarán nota de la ubicación, el área y otros parámetros relevantes, siendo la salida el precio de un inmueble específico.

El algoritmo necesita identificar una relación funcional entre los parámetros de entrada y salida. El valor de salida no es discreto como en la clasificación, sino que es una función de los parámetros de entrada (TIBCO, 2023).

APRENDIZAJE SUPERVISADO DE CLASIFICACIÓN

La clasificación es el lugar donde se entrena a un algoritmo para clasificar los datos de entrada en variables discretas. Durante el entrenamiento, los algoritmos reciben datos de entrada de entrenamiento con una etiqueta de “clasificación”. Por ejemplo, los datos de entrenamiento pueden consistir en las últimas facturas de tarjetas de crédito de un conjunto de clientes, con la etiqueta de si realizaron una compra futura o no fue así. Cuando el saldo de la tarjeta de un nuevo cliente se presenta al algoritmo, este clasificará al cliente en el grupo de “comprará” o “no comprará”. (TIBCO, 2023)

Existen diversos tipos de aprendizajes supervisados de clasificación, los cuales son:

- Clasificación binaria.
- Clasificación multiclase.
- Clasificación de etiquetas múltiples.
- Clasificación con datos desbalanceados.

Para el desarrollo de la propuesta de solución se hará uso de la clasificación multiclase, por ende, a continuación, se procede a explicar la clasificación binaria, que permite dar una idea inicial a la clasificación que se utilizará. En los anexos de la memoria se puede encontrar material para revisar los demás tipos de clasificaciones.

CLASIFICACIÓN BINARIA

Este algoritmo clasifica los datos de entrada en uno de dos grupos posibles. A menudo, una de las clases indica un estado "normal/deseado" y la otra indica un estado "anormal/no deseado".

Un ejemplo es la clasificación de los *e-mails* en las distintas bandejas que se tiene en el correo electrónico. ¿Cómo son capaces los sistemas de clasificar un e-mail como SPAM[17] y cómo pueden mandar otro directamente a la bandeja de entrada? Es muy sencillo, aprenden millones y millones de datos a los que se les asocia determinadas palabras como SPAM. Así, un e-mail de un cliente o un amigo aparecerá en la bandeja de entrada, y una promoción sospechosa probablemente acabe en SPAM. (EDS Robotics, 2021).

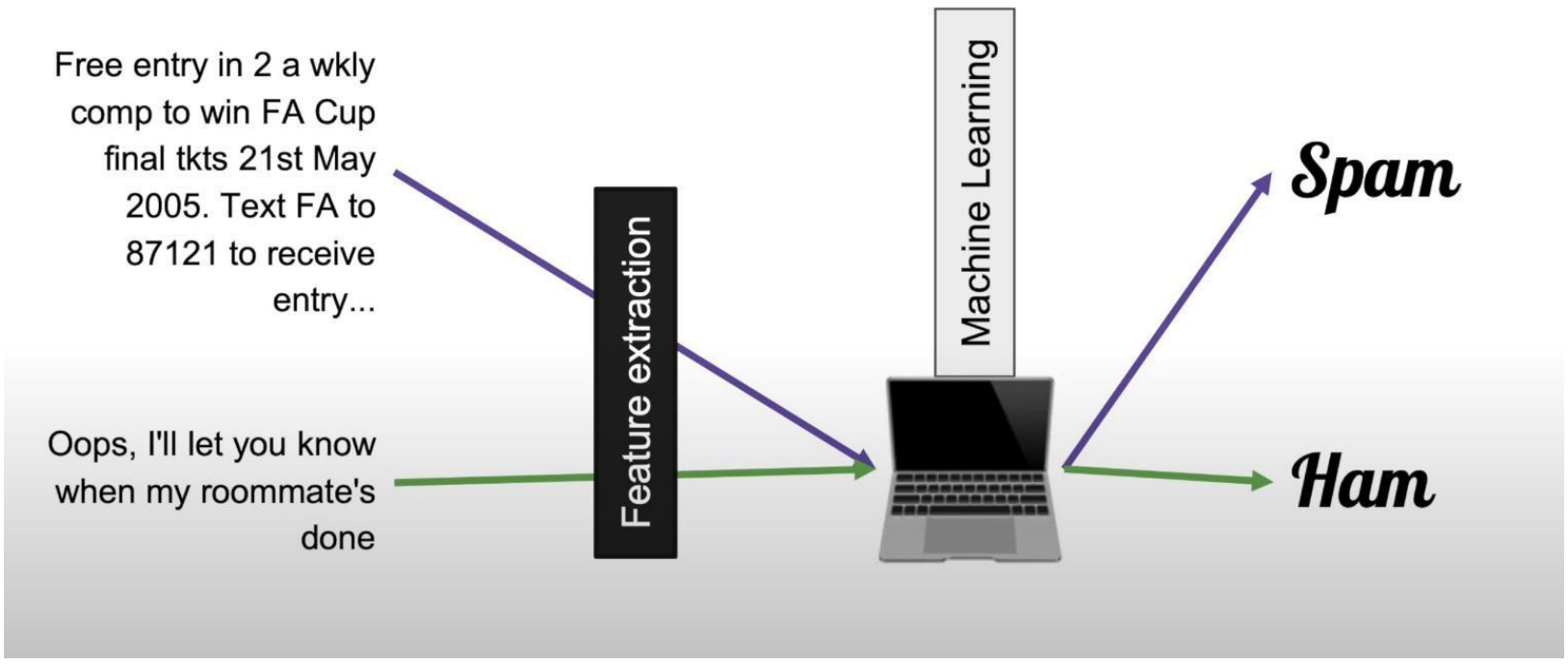


Figura 6: Ejemplo de ML en clasificación de *e-mails*.
Fuente: Feregrino, Youtube, minuto 2:24.

[17] Cualquier forma de comunicación no solicitada que se envía de forma masiva.

CLASIFICACIÓN MULTICLASE

En la clasificación multiclase, el conjunto de datos de entrenamiento se etiqueta con una de las múltiples clases posibles. A diferencia de la clasificación binaria, un algoritmo multiclase se entrena con datos que se pueden clasificar en una de las muchas clases posibles. Una de las aplicaciones para la clasificación multiclase es la clasificación de correo electrónico. Extendiendo el ejemplo de la clasificación binaria acerca de los *e-mails*, podemos pensar en que los correos deseados o HAM, pueden tener su propia clasificación, como social, educación, trabajo, familia, etc.

Como se pudo observar en la figura 5, los algoritmos de *Machine Learning* no reciben texto directamente, debido a que los computadores no lo consumen de esta manera. Previamente existe un proceso conocido como *Feature extraction* también llamado vectorización. Este proceso consiste en convertir las entradas, en este caso texto, a vectores, que vienen a ser números que sí pueden ser consumidos por el algoritmo.

Implementación del *Machine Learning*

Independiente del tipo de aprendizaje a implementar, para su uso se requiere de un *dataset*, es decir un conjunto de datos generalmente estructurados. En el comienzo se tiene el conjunto total de datos, pero para comprobar el correcto funcionamiento de este, se separa en dos conjuntos, uno para el entrenamiento (*train set*) y otro de pruebas (*test set*), generalmente siguiendo las proporciones 80-20 %, respectivamente, tomando muestras aleatorias (no en secuencia, sino, mezclado).

Considerando un escenario en que se tiene un *dataset* en un archivo csv con 10.000 registros, para un algoritmo de clasificación, los conjuntos quedarían cómo:

- X_train 8.000 registros para entrenar.
- y_train con las “etiquetas” de los resultados esperados.
- X_test con 2.000 registros para test.
- y_test con las “etiquetas” de los resultados de X_test.

De modo que el modelo se entrena con (X_train, y_test), y se realizan predicciones de prueba sobre X_test, siendo sus resultados contrastados con los valores reales en y_test. Como métrica inicial se puede utilizar la matriz de confusión, a partir de la cual se pueden calcular las siguientes métricas:

- *Accuracy* (Exactitud).
- *Precision* (Precisión).
- Recall (Sensibilidad).
- Speceficity (Especificidad).
- F1 score (Puntuación F1).

2.1.3 REGISTRO DE EVENTOS DEL SISTEMA OPERATIVO

Para introducir el concepto de registro de eventos, también llamados *logs*, resulta práctico entenderlo con una metáfora. Una torre de vigilancia pierde su sentido si no hay un vigilante dentro, y esto es algo que ha venido sucediendo a lo largo de la historia en diferentes escenarios; por ejemplo, en los siglos XVIII y XIX, los fuertes militares ubicaban centinelas para vigilar el área circundante. Si ocurría una actividad extraña tocaban las campanas, para alertar a los residentes de un peligro inminente. Bajo esta alegoría, el monitoreo de logs viene a representar a los vigilantes de la torre de vigilancia.

En Windows el registro de eventos es un repositorio que se encarga de almacenar los ajustes de configuración y opciones del SO, tales como los ajustes del software, dispositivos hardware, preferencias del usuario, entre otros, por lo que tiene una importancia vital en el equipo. El registro está compuesto por distintos valores, que son instrucciones que se encuentran almacenados en llaves (carpetas que tienen más información). El contenido del Registro de Windows se visualiza y se edita mediante la herramienta Regedit, la cual será presentada más adelante.

Actualmente las empresas u organizaciones que cuentan con sistemas informáticos manejan y procesan una gran cantidad de información, estos sistemas generan ficheros *logs*, con el fin de almacenar los tipos de accesos, errores y alertas que se dan en los distintos niveles de su estructura. Es así como ante cualquier incidente que se esté suscitando como ataques de: inserción de código malicioso, denegación de servicios o simplemente una traza de error en el código de un sistema, estos ficheros *logs* vienen a ser de gran utilidad a la hora de facilitar la información para dar respuesta y seguimiento a cualquier evento de seguridad. (Figueroa Fernández, 2021)

Funciones de los registros de eventos en el sector TI

Los *logs* pueden llevar a cabo labores relacionadas con la seguridad, por ejemplo, el proceso de análisis de uso de un determinado sistema, así como el rendimiento de un programa o sus cambios, entre otros. Además, juegan un rol fundamental para la administración y el control del acceso a determinados recursos, así como las labores de auditoría en un sistema, entre otros. Ayudan a que el usuario tenga una visión amplia de lo que está sucediendo en el entorno de los programas, aplicaciones y demás, lo que permite tener informes relativos a una actividad determinada. Los *logs* pueden analizarse con el objetivo de mantener la seguridad de los sistemas, aplicar técnicas de recuperación de desastres y demás.

Ventajas del monitoreo de los registros de eventos

La importancia del monitoreo *logs* radica en que permite a los administradores mantener un seguimiento constante de los incidentes de seguridad, identificar y solucionar rápidamente los problemas del sistema, y mejorar la eficiencia y el rendimiento del sistema.

Descomponiendo las ventajas del monitoreo de *logs* se tiene que estos ayudan a:

- Detección y análisis de errores: La monitorización de logs ayuda en la identificación temprana de problemas de red y sistemas.

- Entender el funcionamiento del sistema: Ayuda a que el usuario sea consciente de los eventos que ocurren y que podrían ocurrir en su entorno, lo que, a su vez, contribuye en la prevención de fugas de datos y comportamientos que provoquen errores en las plataformas.

- Control de la información: Permite una mejora en la administración de los datos de un sistema o programa, por lo que contribuye a facilitar el acceso y utilización de esa información. De la misma manera, este control de los datos se relaciona con la prevención de amenazas en el sistema.

- Detección temprana de alertas de seguridad: Al monitorear los *logs* se pueden identificar rápidamente los eventos no autorizados o sospechosos que pueden indicar una posible amenaza de seguridad. Esto permite a los administradores tomar medidas para minimizar los riegos y proteger el sistema contra las intrusiones.

- Reparación de problemas del sistema: Los *logs* pueden proporcionar información valiosa sobre los errores y problemas del sistema, lo que permite a los administradores tomar medidas preventivas para minimizar el riesgo de fallos del sistema en el futuro.

- Maximización del rendimiento del sistema: Al monitorear los registros de eventos de Windows, los administradores pueden identificar los problemas que pueden afectar el rendimiento del sistema, como fallos de aplicaciones o retrasos en el procesamiento de datos.

- Cumplimiento normativo: El monitoreo de *logs* es una herramienta clave para cumplir los requisitos normativos. Al mantener una traza de los eventos del sistema, se puede demostrar que se ha cumplido con los requisitos de auditoría y la actividad del sistema puede ser rastreada y auditable.

Herramientas para el monitoreo de los registros de eventos

Los programas de monitoreo de *logs* supervisan la actividad de la red, inspeccionan los eventos del sistema y almacenan diferentes acciones (por ejemplo, cambiar el nombre de un archivo o abrir una aplicación) que ocurren dentro de los sistemas vigilados, contando con la capacidad de consolidar aquellos datos que podrían alertarte sobre una violación de políticas de seguridad.

Ventajas de herramientas para el monitoreo de los registros de eventos

Desde el punto de vista de seguridad, el propósito de las herramientas de gestión y monitoreo de *logs* es permitir la trazabilidad de los acontecimientos en la infraestructura de TI, para así poder mitigar amenazas o prevenir inconvenientes. Revisar los *logs* regularmente, puede ayudar a identificar ataques maliciosos en los sistemas de la organización.

Dada la gran cantidad de datos de registro generados por los sistemas, puede resultar muy engorroso revisarlos de forma manual o asignar a un recurso en el área de TI para la realización de esta tarea de forma exclusiva. Es por ello que el software de monitoreo de *logs* o registros facilita esta tarea mediante la aplicación de reglas que automatizan la revisión de estos registros, para generar alertas solo para aquellos eventos que puedan representar problemas, violación de políticas, incumplimiento de estándares o amenazas. (E-DEA NETWORKS, 2019)

2.2 METODOLOGÍA *THREAT DETECTION*

La detección de amenazas, también conocida como búsqueda de ciberamenazas, es un enfoque proactivo para identificar amenazas previamente desconocidas o amenazas en curso no remediadas dentro de la red de una organización.

Un programa exitoso de detección de amenazas se basa en la fertilidad de los datos de un entorno. En otras palabras, una organización primero debe contar con un sistema de seguridad empresarial que recopile datos. La información recopilada proporciona pistas valiosas para los cazadores de amenazas.

(IBM, 2023)

2.3 HERRAMIENTAS

2.3.1 VISOR DE EVENTOS

Figura 7: Icono Visor de Eventos.
Fuente: Google.

El visor de eventos es un componente del sistema operativo de Windows de Microsoft que permite a los administradores y usuarios ver los registros de eventos en una máquina local o remota. Las aplicaciones y los componentes del SO pueden usar este servicio de registro centralizado para informar eventos que han tenido lugar, como fallas al iniciar un componente o completar una acción.

2.3.2 EXCEL

Figura 8: Ícono Excel.
Fuente: Google.

Excel es una herramienta muy eficaz para obtener información con significado a partir de grandes cantidades de datos. También funciona muy bien con cálculos sencillos y para realizar el seguimiento de casi cualquier tipo de información. La clave para desbloquear todo este potencial es la cuadrícula de las celdas. Las celdas pueden contener números, texto o fórmulas. Los datos se escriben en las celdas y se agrupan en filas y columnas. Esto permite sumar datos, ordenarlos y filtrarlos, ponerlos en tablas y crear gráficos muy visuales. (Microsoft, 2023)

2.3.3 GOOGLE COLAB

Figura 9: Ícono Google Colab.
Fuente: Google.

Colab, también conocido como *Colaboratory* es una herramienta creada por Google que permite programar y ejecutar código Python en el navegador, presentando las siguientes ventajas:

- ❖ No requiere configuración.
- ❖ Acceso a GPUs sin coste adicional.
- ❖ Permite compartir contenido fácilmente.
- ❖ Permite combinar código ejecutable y texto enriquecido en un mismo documento.

Las aplicaciones de Google Colab se vinculan con la ciencia de datos, ya que aprovecha toda la potencia de las bibliotecas más populares de Python para analizar y visualizar datos; y con el aprendizaje automatizado, dado que se puede importar un conjunto de datos, entrenar un clasificador y evaluar en modelo con pocas líneas de código. (Colab, 2023)

2.3.4 PYTHON

Figura 10: Ícono Python.
Fuente: Google.

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el *Machine Learning*. Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. El software Python se puede descargar gratis, se integra bien a todos los tipos de sistemas y aumenta la velocidad del desarrollo. Dentro de los beneficios de este lenguaje se encuentran:

- Los desarrolladores pueden leer y comprender fácilmente los programas de Python debido a su sintaxis[18] básica similar a la del inglés.
- Python permite que los desarrolladores sean más productivos, ya que pueden escribir un programa con menos líneas de código en comparación con muchos otros lenguajes.
- Cuenta con una gran biblioteca estándar que contiene códigos reutilizables para casi cualquier tarea. De esta manera, los programadores no tienen que escribir el código desde cero.
- La comunidad activa de Python incluye millones de desarrolladores alrededor del mundo que prestan su apoyo.
- Se puede trasladar a través de diferentes sistemas operativos.

Librerías de Python

Una librería en programación es un archivo importable que se utiliza para desarrollar software. Suele estar compuesta de código y datos, y su fin es ser utilizada por otros programas de forma totalmente autónoma. Para la instalación de las librerías en Windows una alternativa es la utilización de pip, un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python.

Librería Scikit-Learn

Figura 11: Ícono Scikit-learn.
Fuente: Google.

Scikit-learn es una biblioteca de aprendizaje automático open source para Python. Cuenta con varios algoritmos de clasificación, regresión y agrupamiento, que incluyen máquinas de vectores de soporte, bosques aleatorios, aumento de gradiente, k-means y DBSCAN, y está diseñado para interoperar con las bibliotecas numéricas y científicas de Python NumPy y SciPy. Scikit-learn es un proyecto patrocinado fiscalmente por NumFOCUS.

[18] Disciplina lingüística que estudia el orden y la relación de las palabras o sintagmas en la oración, así como las funciones que cumplen

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN

3.1 PRESENTAR Y ANALIZAR UN PROBLEMA DE CIBERSEGURIDAD

En el marco conceptual se definió lo que son los registros de eventos y la relevancia de su monitorización en la industria. El problema presente son los costos de los servicios de monitoreo asociados a los *logs* considerando como referencia microempresas que no tienen la capacidad de financiar dichos servicios, y por ello en caso de ser conscientes de la importancia del monitoreo de *logs*, tendrían que realizarlo con recurso humano, lo que a nivel de tiempo es más costoso y menos eficiente por la alta tasa de error a comparación de una herramienta o software automatizado.

Frente a esta problemática, la solución propuesta consiste en un *software open source* que analiza los registros de eventos de manera automatizada. El *software* desarrollado está escrito en el lenguaje de programación Python, ya que como se expuso en el capítulo anterior, tiene gran afinidad con la ciencia de datos y *el Machine Learning*, siendo este la aplicación de la IA implementada. El algoritmo de *Machine Learning* se basa en el aprendizaje supervisado de clasificación múltiple, por el hecho de que como su nombre lo indica, un registro de eventos está compuesto por diversos eventos, por lo que en el compendio puede existir más de un ataque malicioso. Por otro lado, se dice que está basado en el aprendizaje supervisado porque el algoritmo es entrenado a partir de un *dataset* de eventos etiquetados. Finalmente, se debe mencionar que la metodología que emplea el algoritmo se denomina *Threat Detection*, ya que como fue explicado con anterioridad, se buscan amenazas conocidas.

Dentro del Registro de Windows, destacan cinco registros importantes para el análisis:

- *SAM*: El Administrador de Cuentas de Seguridad, es una base de datos que almacena cuentas de usuarios y descriptores de seguridad para los usuarios en el equipo local, por ejemplo, su última conexión.
- *SOFTWARE*: Proporciona información relacionada al software instalado y su configuración.
- *SYSTEM*: Contiene información sobre el programa de instalación del sistema Windows, lista de dispositivos montados actualmente que contiene un sistema de archivos, cuándo y cuantas veces un usuario ha utilizado medios extraíbles.
- *NTUSERDAT*: Contiene información sobre los usuarios del equipo, *URL*'s visitadas, *software* instalado, etc.
- *SECURITY*: Pueden proporcionar detalles sobre una variedad de acciones, incluyendo autenticación de usuarios y lo hecho por los mismos después de su autenticación.

La solución propuesta se centra en los registros de eventos de seguridad, ya que el tópico de esta memoria está relacionado a la ciberseguridad, específicamente a los ataques al sistema operativo. Algunas de las acciones que están involucradas en este tipo de eventos son:

- Cambios de contraseña.
- Inicios de sesión no autorizados.
- Fallos de inicio de sesión.
- Nuevos eventos de inicio de sesión.
- Detección de *malware*.
- Ataques de *malware* vistos por IDS u otra evidencia.
- Escaneo en firewalls de puertos abiertos y cerrados.
- Ataques de denegación de servicio.
- Errores en dispositivos de red.
- Cambios de nombre de archivo.
- Cambios en la integridad del archivo.
- Datos exportados.
- Nuevos procesos iniciados o procesos en ejecución detenidos.
- Eventos de acceso compartido.
- Eventos desconectados.
- Instalación de nuevos servicios.
- Auditoría de archivos.
- Nuevas cuentas de usuario.
- Valores de registro modificados.
- Eventos de *logueo* fuera del horario laboral.
- Intento de acceso a sistemas por parte de perfiles no autorizados

(E-DEA NETWORKS, 2019)

Retomando la solución propuesta, el algoritmo de *Machine Learning* fue desarrollado mediante la librería *Scikit-Learn* de Python, esto debido a que es especialmente buena para los no especialistas, ya que les permite utilizar fácilmente una aplicación de software general, encontrándose además dentro de las referentes en relación con el reconocimiento de patrones mediante *Machine Learning* (Phddirection, 2019).

Como cualquier algoritmo de *Machine Learning*, este debe ser entrenado, para lo cual se necesita un *dataset*, generalmente entregado en formato csv. El primer paso en la solución propuesta consistió en crear el *dataset* a partir de *logs* normales y maliciosos etiquetados.

Para la elección y creación del modelo se hizo uso de la plataforma Google Colab, ya que permite visualizar los datos, comparar diversos modelos de *Machile Learning* e implementar criterios de validación del mismo.

Cuando el modelo estuvo finalizado fue llevado al editor de código, en este caso Visual Studio Code, para desarrollar mediante la librería Tkinter, la interfaz gráfica, que sería con lo que finalmente interactúa el usuario, pudiendo cargar un registro de eventos, visualizar la predicción de su categoría y descargar un informe en formato pdf y Excel.

Diagrama de Flujo

A continuación, se presenta un diagrama de flujo del funcionamiento del programa:

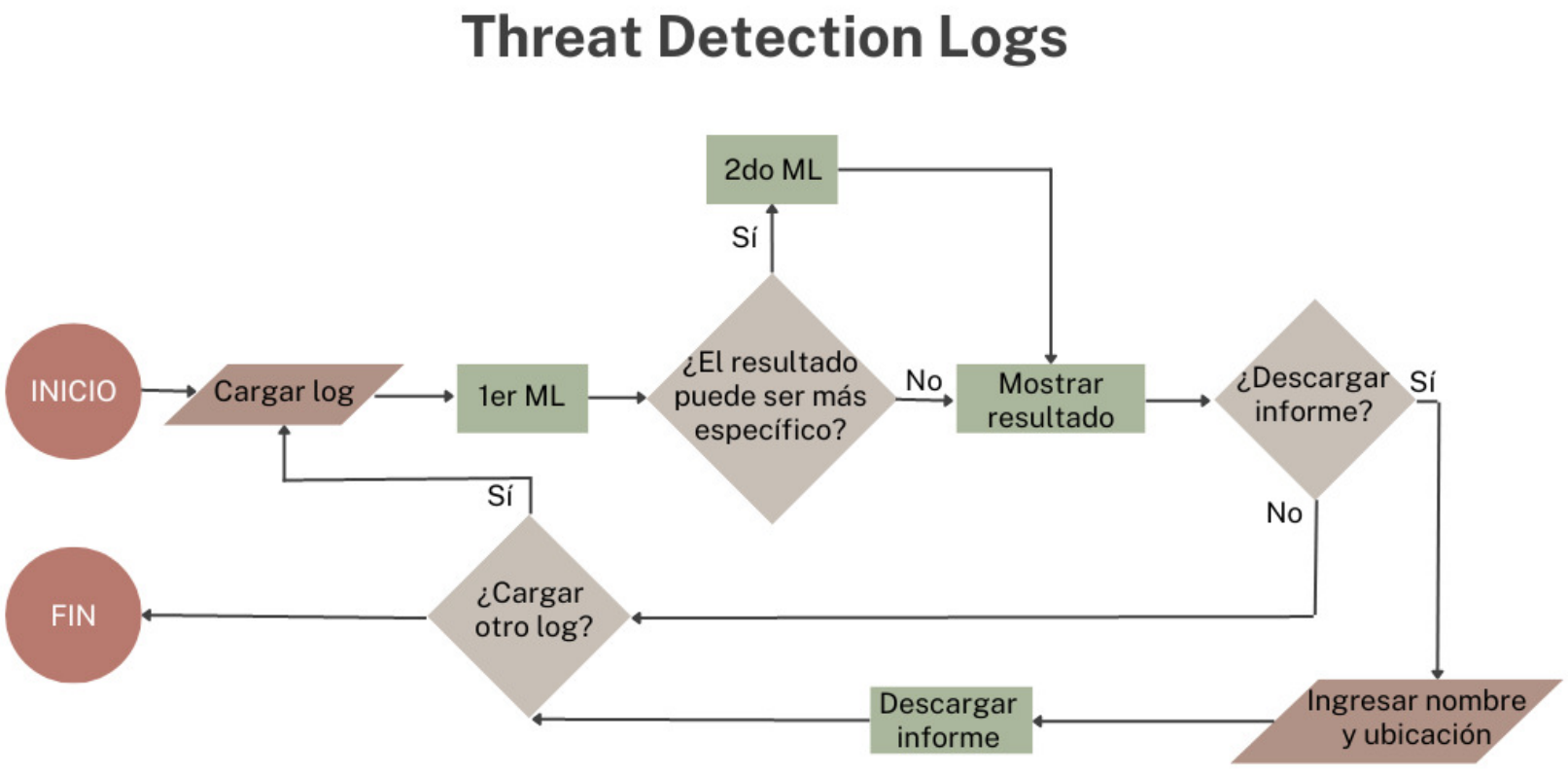


Figura 12: Diagrama de flujo.
Fuente: Elaboración propia.

3.2 DISEÑAR UN ALGORITMO BASADO EN TÉCNICAS DE INTELIGENCIA ARTIFICIAL

Previo a comenzar a desarrollar la implementación, se considera relevante explicitar los recursos empleados para justificar la duración de la ejecución del código y el tamaño del dataset.

Recursos empleados

Hardware: Ordenador personal Acer Nitro 5 AN515-43-RM4-4 con:

- Procesador: AMD Ryzen 7 3750H (4 núcleos / 8 hilos / 2.30 GHz)
- RAM: 12 GB DDR4 (2400 MHz)
- Almacenamiento: HDD 1TB (5400 rpm) | 1 SDD 240 GB | 1 SDD 128 GB
- Tarjetas de video: AMD Radeon RX Vega 10 (Integrada) | Radeon RX 560X Series

Software: Sistema Operativo Windows 10 Pro con los siguientes programas:

- Visual Studio Code: Editor de código.
- Python3 con las librerías.
- Excel Microsoft 365.
- Opera Gx como navegador para utilizar Google Colab.

Descargar registros de eventos de Windows

Para la implementación de la solución se necesitan registros de eventos de Windows benignos, es decir de un usuario promedio, como en este caso yo, un estudiante universitario, y *logs* maliciosos. Para la obtención de los primeros se debe seguir los siguientes pasos:

1. En la barra de búsqueda de Windows escribir *Visor de Eventos.*
2. Abrir la aplicación.
3. Expandir la carpeta de *Registros de Windows* en el panel izquierdo.
4. Seleccionar los registros de Windows de Seguridad.
5. En el panel de la derecha, en la sección de *Acciones* elegir la opción de *Guardar todos los eventos como…* a modo de descargar los *logs* como un archivo evtx.

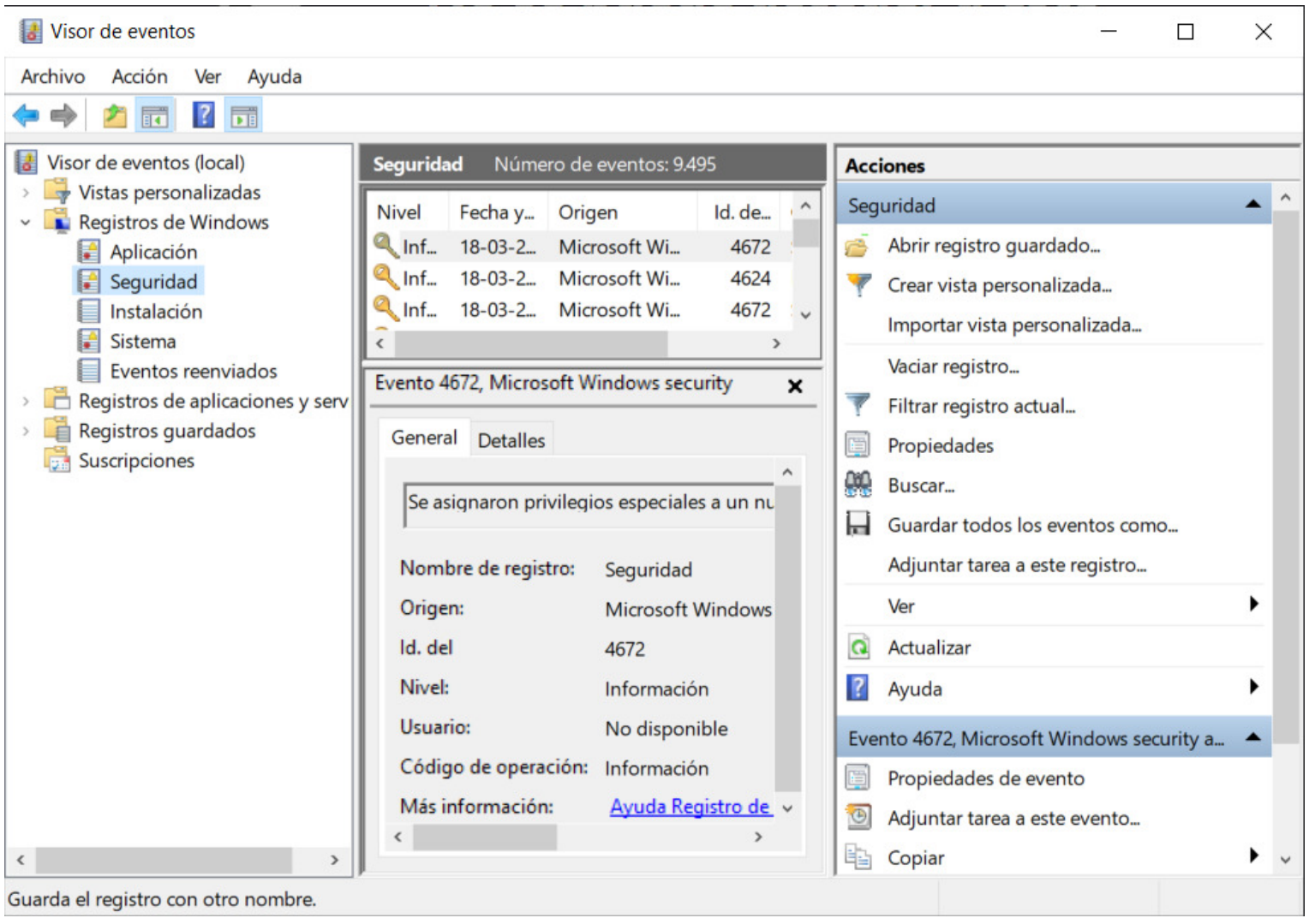


Figura 13: Visor de eventos de Windows.
Fuente: Elaboración propia.

De esta forma se descargaron 4 registros de eventos de diferentes ordenadores, correspondientes a diferentes propietarios con el fin de tener *logs* con comportamiento normal de distintos perfiles.

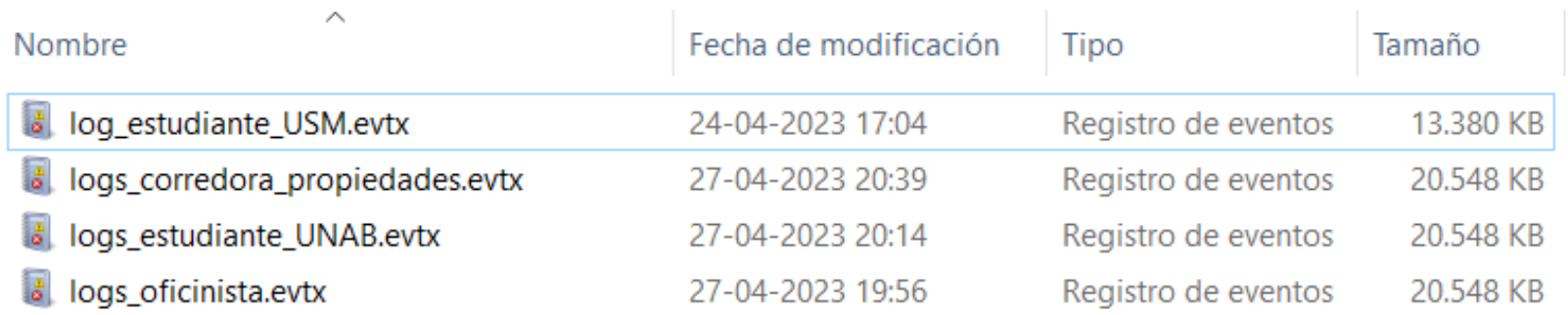


Figura 14: Registros de eventos normales.
Fuente: Elaboración propia.

Cabe mencionar que cada *log* contiene más de 15.000 eventos, aunque como se verá más adelante, no se hará uso de su totalidad.

Para la obtención de *logs* maliciosos se consideró levantar una máquina virtual, realizarle diversos tipos de ataques y repetir los pasos anteriores para la descarga de estos registros de eventos. No obstante, esto implicaría dedicar tiempo en investigar y realizar dichos

ataques, en especial por la variedad que se necesita para entrenar un buen modelo de IA. Es por ello que se decidió hacer uso de *logs* maliciosos existentes y puestos a disposición pública, como los que he encontrado en la plataforma de manejo de versiones GitHub, subidos por los usuarios sbousseaden en su repositorio EVTX-ATTACK-SAMPLES y mdecrevoiser en su repositorio EVTX-To-MITRE-Attack.

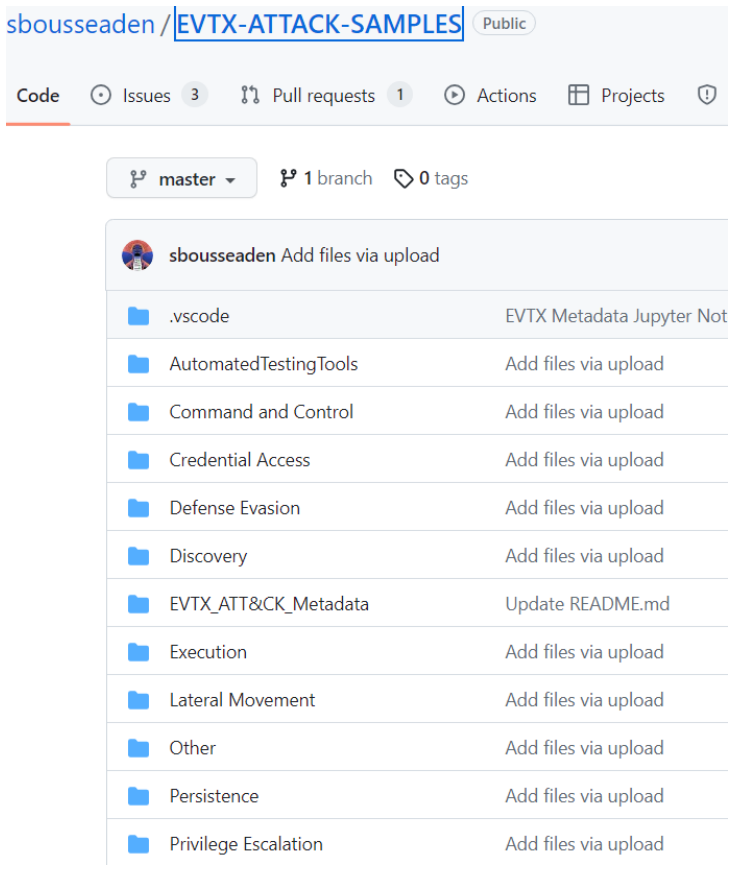


Figura 15: *Logs* maliciosos.
Fuente: *Screenshot* de repositorio GitHub de Sbousseaden.

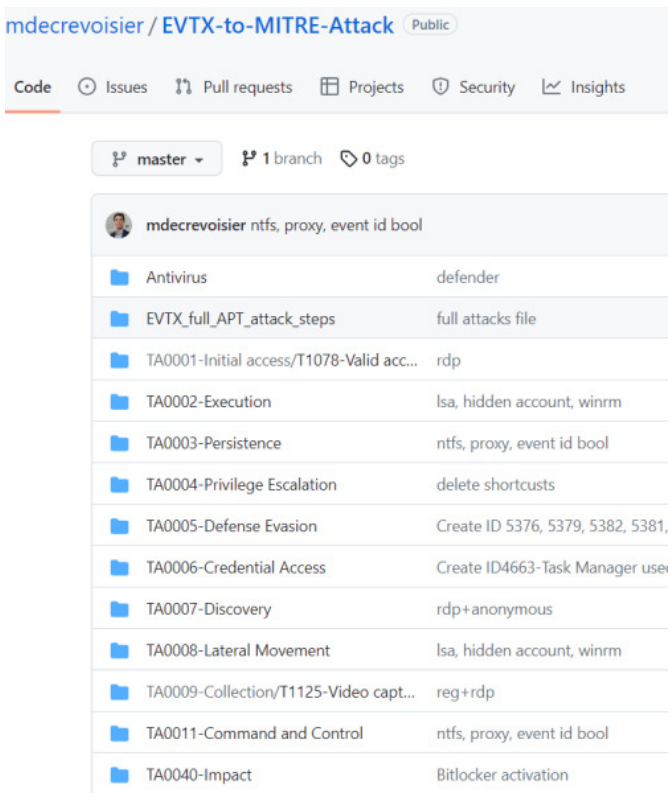


Figura 16: *Logs* maliciosos.
Fuente: *Screenshot* de repositorio GitHub de mdecrevoisier.

Lo valioso de estos *datasets* es que se encuentran diversos tipos de ataques y más de un archivo evtx por cada uno, lo que permite entrenar de mejor manera el modelo de M*achine Learning.* Además, en el segundo repositorio cada categoría presenta una subcategoría del tipo de ataque, pudiendo ser aún más preciso con la detección. A partir de los repositorios se generaron las siguientes etiquetas:

- Automated Testing Tools.
- Command and Control:
  - Proxy.
  - Protocol Tunneling.
- Credential Access:
  - Credential dumping.
  - Traffic sniffing.
  - Brut force.
  - Unsecured Credentials-Private Keys.
  - Credentials from Password Stores.
  - Man in the middle.
  - Steal of Forge Kerberos Tickets.
- Defense Evasion:
  - Obfuscated Files or Information.
  - Clear Windows event logs.
  - Timestomp.
  - Valid accounts-Domain acoounts.
  - Modify registry.
  - Deobfuscate-Decode Files or Information.
  - Rogue domain controller.
  - File and Directory Permissions Modification.
  - Subvert Trust Controls.
  - Windows Credential Manager.
  - Impair Defenses-Disable or Modify tool.
  - Disable Windows Event Logging.
  - Impair Defenses-Disable or Modify System Firewall.
  - Hide artifacts.
- Discovery:
  - System Network Configuration Discovery.
  - Remote System Discovery.
  - Permission Groups Discovery.
  - Account Discovery.
  - Network Share Discovery.
  - Password Policy Discovery.
  - Domain Trust Discovery.

- Execution:
  - Windows Management Instrumentation.
  - Scheduled Task.
  - Powershell.
  - Windows Command Shell.
  - User execution.
  - Service execution.
- Lateral Movement:
  - Remote Desktop Protocol.
  - SMB Windows Admin Shares.
  - Distributed Component Object Model (DCOM).
  - Remote Service SSH.
  - Windows Remote Management.
  - Use Alternative Authentication Material.
- Persistence:
  - Account manipulation.
  - Create account.
  - BITS Jobs.
  - SQL Stored Procedures.
  - Server Software Component.
  - Create or Modify System Process-Windows Service.
  - Event Triggered Execution.
  - Boot or Logon Autostart Execution.
  - Hijack Execution Flow.
- Privilege Escalation:
  - Exploitation for Privilege Escalation.
  - Access Token Manipulation.
  - Domain Policy Modification-Group Policy Modification.
  - Image File Execution Options Injection.
  - DLL side-loading.
- Other: Ataques muy puntuales (pocos registros para tener una categoría propia).
  - Antivirus:
  - Collection - Video Capture:
  - Initial Access – Valid Account:
  - Impact – Data Encrypted for Impact
  - Impact – Inhibit System Recovery
  - Impact – Data manipulation
  - Sin clasificar.
- Normal: *Logs* con comportamiento normal.

Con los repositorios descargados y fusionados se crearon dos carpetas:

- *Logs*: Carpeta que contiene carpetas con los *logs* de ambos repositorios combinados por categoría.

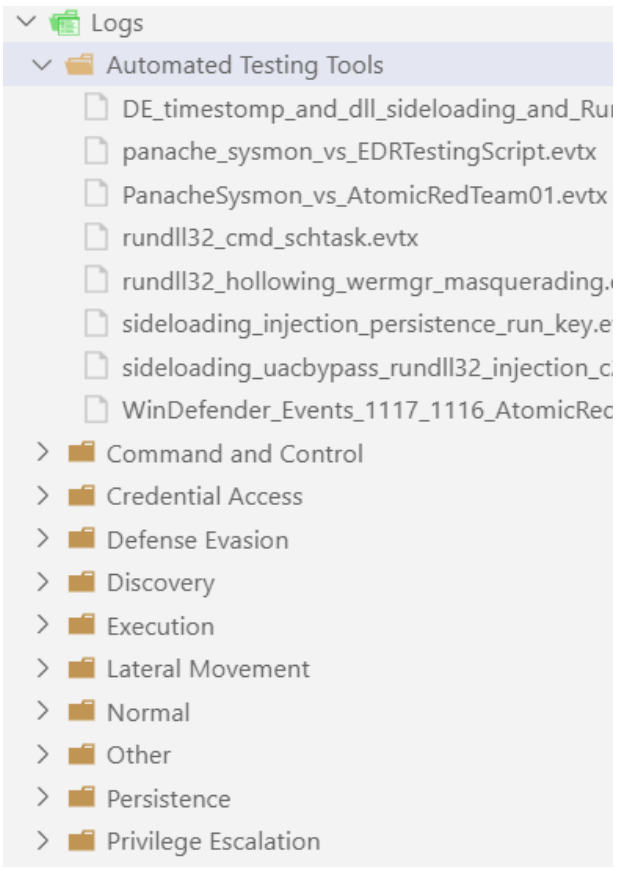


Figura 17: Carpeta Logs y contenido.
Fuente: Elaboración propia.

- *SpecificLogs*: Carpeta que contiene carpetas con subcarpetas con los *logs* del segundo repositorio de acuerdo con las categorías específicas.

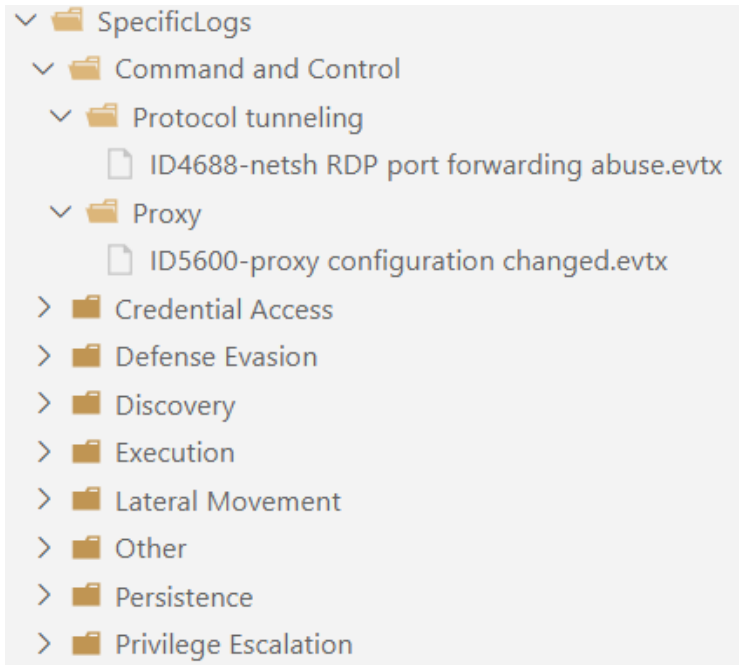


Figura 18: Carpeta SpecificLogs y contenido.
Fuente: Elaboración propia.

Con los registros de eventos listos y etiquetados, el siguiente paso consistió en crear los *datasets*. Para ello se hizo uso de la herramienta Excel, creando los archivos en formato csv, es decir celdas separadas por coma. Al tratarse de muchos *logs*, esta tarea fue automatizada con un programa en Python, por lo que el primer paso fue instalar las herramientas necesarias.

Instalación de herramientas

Python – Librerías

Como se ha mencionado, la propuesta de solución es trabajada en Windows, por lo cual se realizó la instalación del lenguaje de programación Python para este sistema operativo. Para ello se ingresa a la página oficial de Python, y se instala como una aplicación normal.

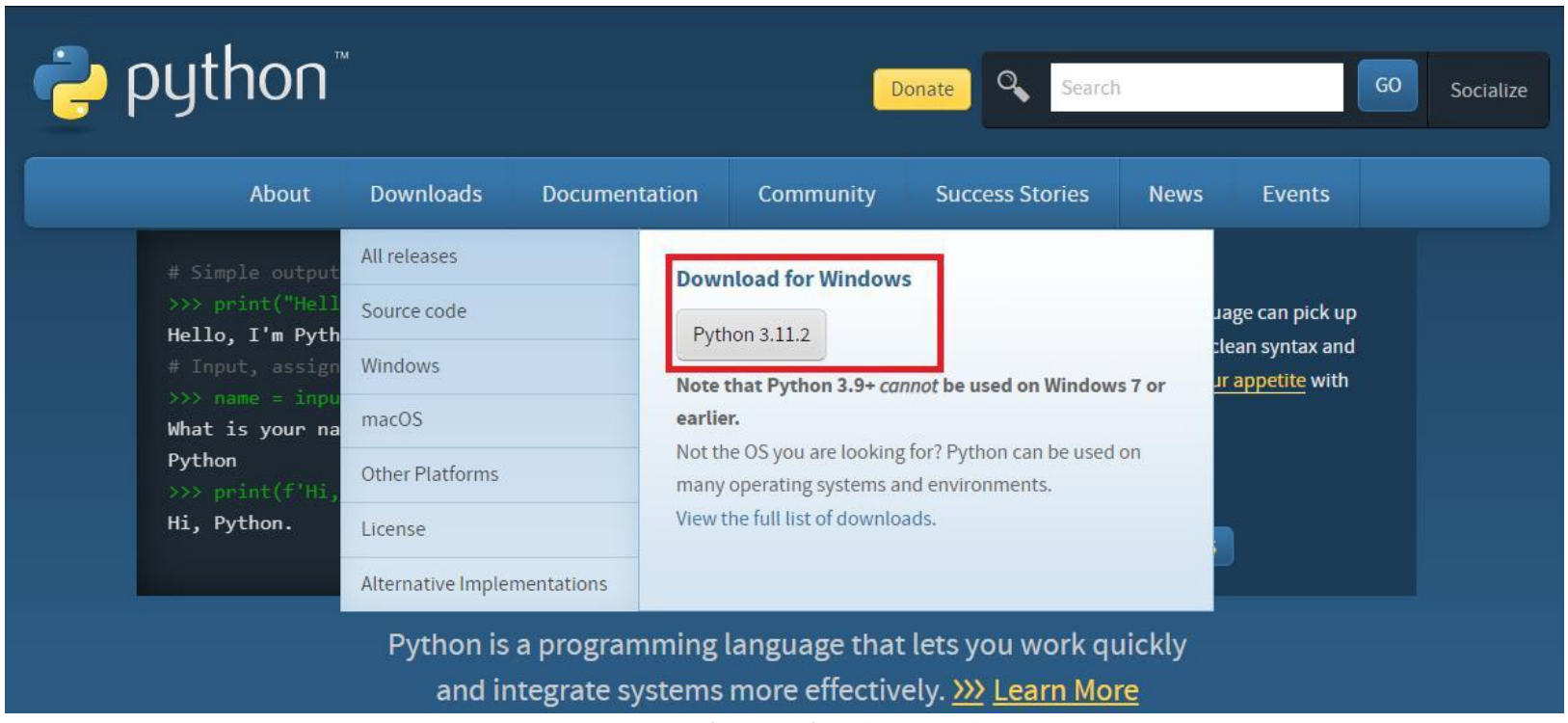


Figura 19: Página oficial de Python.
Fuente: Python.

La versión utilizada es la 3.11.2. Cabe destacar que según Phoronix, un destacado sitio web de tecnología que ofrece comentarios, Python 3.11 es entre un 10-60 % más rápido que Python 3.10, lo que hasta hoy era la versión estable más actualizada. (Phoronix, 2022)

Para instalar las librerías necesarias en Python se debe acceder al símbolo de sistema de Windows, también llamada terminal o CMD, y ejecutar el comando, respectivamente:

```
pip install <NOMBRE LIBRERÍA>
```

Las librerías y el comando de instalación se pueden encontrar en la página oficial de librerías Python.

Las librerías instaladas utilizadas son:

Tabla 3: Librerías Python instaladas.
Fuente: Elaboración propia.

Librería	**Versión**	**Descripción**
evtx	0.8.2	Librería para manipular archivos evtx.
pandas	1.5.3	Librería especializada en la manipulación y análisis de datos.
scikit-learn	1.2.2	Biblioteca de aprendizaje automático.
pillow	9.5.0	Librería para añadir imágenes procesadas.
openpyxl	3.1.2	Librería para leer/escribir archivos Excel (xlxs/xlsm/xltx/xltm).
matplotlib	3.7.1	Librería para generar gráficos 2D a partir de listas.
reportlab	4.0.0	Librería para generar archivos pdf y gráficos.
svglib	1.5.1	Librería para leer archivos SVG y convertirlos a otros formatos.
auto-py-to-exe	2.36.0	Librería para convertir en ejecutable un archivo Python.

Con el comando *pip freeze* se puede verificar la instalación de las librerías.

Las librerías que viene incluidas en Python utilizadas son:

- csv: Permite trabajar con archivos csv.
- Io: Proporciona las funciones principales de Python para manejar varios tipos de E/S.
- os: Para manipular el *path* de un directorio o archivo.
- stadistics: Proporciona funciones para calcular estadísticas matemáticas de datos numéricos (valores reales).
- Tkinter: Biblioteca gráfica tcl/Tk.

Creación de *datasets*

Con el lenguaje de programación instalado junto a las librerías necesarias, se procede a programar 2 programas similares:

- CSV_Events.py: A partir de la carpeta "*Logs*" creada previamente, el programa lee cada carpeta que almacena una categoría de *log*, recorriendo cada registro de evento, evento por evento, para finalmente generar un archivo csv con el nombre "dataset_events.csv" el cual contiene 2 columnas, siendo la primera (A) "*Label*", y la segunda (B) "*Event*". Como su traducción indica, la primera columna hace referencia a una de las 11 categorías de registros de eventos posible, y la segunda columna almacena el contenido del evento. De este modo, el archivo csv creado posee un total de 18876 eventos, pesando un total de 18.4 MB. Este último dato resulta relevante, ya que sobre los 20 MB aproximadamente la CPU se ve sobrepasada y no logra ejecutar el modelo de *Machine Learning*.

A continuación, se muestra el código fuente para la creación del *dataset*:

CSV_Events.py

codigo > CSV_Events.py > ReadLogs

```python
import csv
from evtx import PyEvtxParser
import os

def ReadLogs(segment):
    total_events = 0
    for registry in os.listdir('Logs/'+segment):
        parser = PyEvtxParser('Logs/'+segment+'/'+registry)
        data = []
        for record in parser.records_json():
            event = record['data']
            for elem in remov:
                event = event.replace(elem,'')
            if(segment!='Normal'):
                total_events+=1
                if (len(event) <= 2024):
                    lista_csv.append([segment,event])
            else:
                data.append(event)
        if (segment == 'Normal'):
            event = data[:550]
            total_events+=len(event)
            for item in event:
                lista_csv.append([segment,item])
    return total_events

remov = ['\n','     ','         ','?','\"','}','{',',']
events_analyzed = len(os.listdir('Logs'))*[0]
lista_csv = [['Label','Event']]

cont = 0
for folder in os.listdir('Logs'):
    events_analyzed[cont]+=ReadLogs(folder)
    cont+=1

with open('dataset_events.csv', 'w' , newline='', encoding='utf-8') as file:
    writer = csv.writer(file, delimiter=',')
    writer.writerows(lista_csv)
file.close()
```

Figura 20: Código de CSV_Events.py.
Fuente: Elaboración propia.

Cargando el archivo dataset_events.csv en Google Colab se puede visualizar y extraer la siguiente información del *dataset*:

```
dataset.columns = ["Label", "Log"]
dataset
```

```
Mounted at /content/drive
```

	Label	Log
0	Automated Testing Tools	Event: #attributes: xmlns: http://schem...
1	Automated Testing Tools	Event: #attributes: xmlns: http://schem...
2	Automated Testing Tools	Event: #attributes: xmlns: http://schem...
3	Automated Testing Tools	Event: #attributes: xmlns: http://schem...
4	Automated Testing Tools	Event: #attributes: xmlns: http://schem...
...	...	...
18871	Privilege Escalation	Event: #attributes: xmlns: http://schem...
18872	Privilege Escalation	Event: #attributes: xmlns: http://schem...
18873	Privilege Escalation	Event: #attributes: xmlns: http://schem...
18874	Privilege Escalation	Event: #attributes: xmlns: http://schem...
18875	Privilege Escalation	Event: #attributes: xmlns: http://schem...

18876 rows × 2 columns

Figura 21: dataset_events.csv.
Fuente: Elaboración propia.

```
dataset["Label"].value_counts()
```

```
Defense Evasion          2909
Execution                2724
Normal                   2200
Command and Control      1987
Credential Access        1777
Lateral Movement         1581
Privilege Escalation     1291
Other                    1266
Discovery                1223
Persistence              1120
Automated Testing Tools   798
```

Figura 22: Distribución numérica de eventos en dataset_events.csv.
Fuente: Elaboración propia.

Graficando estos datos con *matplotlib* se puede visualizar de la siguiente manera:

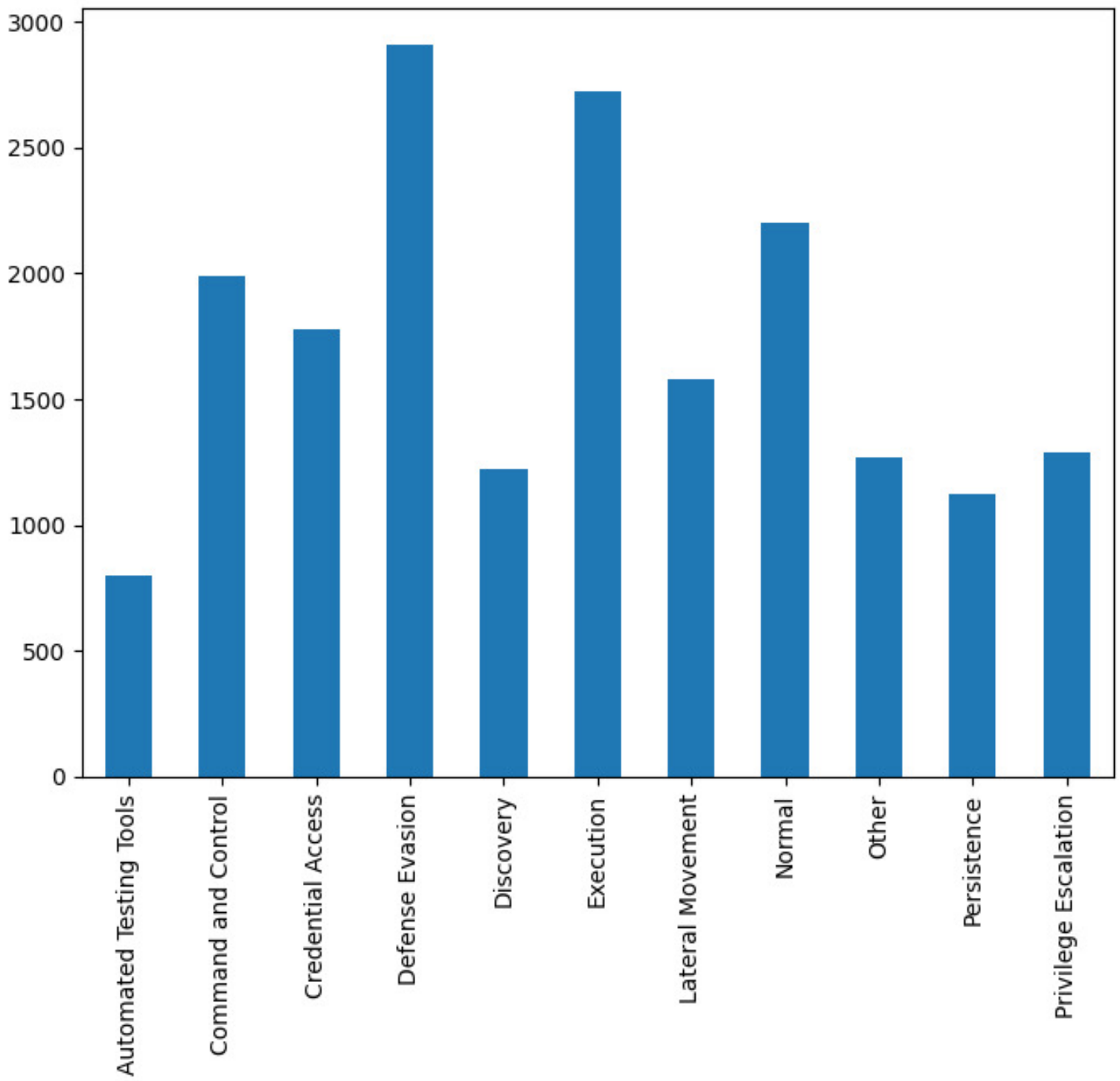


Figura 23: Distribución gráfica de eventos en dataset_events.csv.
Fuente: Elaboración propia.

- CSV_Specific.py: A partir de la carpeta "Specific*Logs*" creada con anterioridad, el programa lee cada subcarpeta dentro de las carpetas que almacena una categoría de *log*, recorriendo cada registro de evento, evento por evento, para finalmente generar un archivo csv con el nombre correspondiente a la categoría, por ejemplo, "dataset_events_Command and Control.csv". De la misma que el archivo "dataset_events.csv", se tienen 2 columnas que indican la categoría y el contenido del evento.

CSV_Specific.py

codigo > CSV_Specific.py > ...

```python
import csv
from evtx import PyEvtxParser
import os

def ReadLogs(folder, segment):
    for registry in os.listdir('SpecificLogs/'+folder+'/'+segment):
        parser = PyEvtxParser('SpecificLogs/'+folder+'/'+segment+'/'+registry)
        for record in parser.records_json():
            event = record['data']
            for elem in remov:
                event = event.replace(elem,'')
            lista_csv.append([segment,event])
    return None

remov = ['\n','     ','         ','?','\"','}','{',',']

for folder in os.listdir('SpecificLogs'):
    lista_csv = [['Label','Event']]
    for label in os.listdir('SpecificLogs/'+folder):
        ReadLogs(folder, label)
    with open('CSV_Especificos/dataset_events_'+folder+'.csv', 'w' , newline='', encoding='utf-8') as file:
        writer = csv.writer(file, delimiter=',')
        writer.writerows(lista_csv)
    file.close()
```

Figura 24: Código de CSV_Specifics.py.
Fuente: Elaboración propia.

Figura 25: Carpeta CSV_Especificos y contenido.
Fuente: Elaboración propia.

Con los *dataset* creados, el siguiente paso consistió en crear el modelo de *Machine Learning*. Previo a escoger un modelo, se realizaron los siguientes pasos:

- Exploración de datos: Se agrega una columna que codifica la categoría como un número entero, ya que las variables categóricas a menudo están mejor representadas por enteros en vez de cadenas. Se obtiene el siguiente resultado:

	Label	Log	category_id
0	Automated Testing Tools	Event: #attributes: xmlns: http://schem...	0
1	Automated Testing Tools	Event: #attributes: xmlns: http://schem...	0
2	Automated Testing Tools	Event: #attributes: xmlns: http://schem...	0
3	Automated Testing Tools	Event: #attributes: xmlns: http://schem...	0
4	Automated Testing Tools	Event: #attributes: xmlns: http://schem...	0
...	...	...	...
18871	Privilege Escalation	Event: #attributes: xmlns: http://schem...	10
18872	Privilege Escalation	Event: #attributes: xmlns: http://schem...	10
18873	Privilege Escalation	Event: #attributes: xmlns: http://schem...	10
18874	Privilege Escalation	Event: #attributes: xmlns: http://schem...	10
18875	Privilege Escalation	Event: #attributes: xmlns: http://schem...	10

18876 rows × 3 columns

Figura 26: dataset_events.csv modificado.
Fuente: Elaboración propia.

- Representación de texto: Los clasificadores y algoritmos de aprendizaje no pueden procesar directamente los documentos de texto en su forma original, ya que la mayoría de ellos espera vectores de características numéricas con un tamaño fijo en lugar de documentos de texto sin procesar con longitud variable. Por lo tanto, durante el paso de preprocesamiento, los textos se convierten en una representación más manejable. Un enfoque común para extraer características del texto es usar el modelo de la bolsa de palabras, un modelo donde para cada documento, en nuestro caso un evento, se tiene en cuenta la presencia (y a menudo la frecuencia) de palabras, pero se ignora el orden en que ocurren.

Específicamente, para cada término de nuestro conjunto de datos, se calcula una medida llamada Frecuencia de términos, Frecuencia de documentos inversa, abreviado a tf-idf. A continuación, se muestra la implementando en código:

```
tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5, norm='l2', encoding='utf-8', ngram_range=(1, 2), stop_words='english')

features = tfidf.fit_transform(dataset.Log).toarray()
labels = dataset.category_id


features.shape
```

```
(18876, 22783)
```

Figura 27: Código de CSV_Specifics.py.
Fuente: Elaboración propia.

Los parámetros de la función TfidfVectorizer se establecen como:

- *sublinear_tf* en *True* para usar una forma logarítmica para la frecuencia.
- *min_df* es el número mínimo de documentos para los que debe estar presente una palabra.
- *norm* se establece en l2 para asegurar que todos los vectores de los eventos tengan una norma euclidiana de 1.
- *ngram_range* se establece en (1,2) para indicar que se considerarán los *unigrams* y *bigrams*.
- *stop_words* se establece en *english* para eliminar todos los pronombres comunes (*a*, *the*, entre otros) con el fin de reducir la cantidad de funciones ruidosas.

El *output* que se muestra en la imagen indica que los 18876 eventos están representados por 22783 funciones, que indican la puntuación tf-idf para diferentes *unigrams* y *bigrams*. Estos vienen a ser indicadores interesantes para considerar, mientras *unigrams* muestra las palabras más relacionadas a una categoría de evento, *bigrams* señala las frases más relacionadas. Seleccionando N = 3 como el número de elementos a mostrar, obtenemos la siguiente información:

Tabla 4: Unigrams y Bigrams modelo *Machine Learning*.
Fuente: Elaboración propia.

Categoría	***Unigrams* más relacionados**	***Bigrams* más relacionados**
Automated Testing Tools	3592, ping	3592 keywords, 5d31 0000
Command and Control	E76b077bd51ename, bits	bits clients, windows bits
Credential Access	Authenticationservice, 0x883836e, 60296	ipport 60296, 12t11 53, 11 ipport
Defense Evasion	508threadid, 13056	eventdata accessmask, security objecttype
Discovery	99db2437fdda, ae3b6f96	abd2 46e7, 46e7 9679
Execution	assembly, control	subjectlogonid 0x18acdd, service control
Lateral Movement	win32, bloodhound	15 ipport, bloodhound win32
Normal	readoperation, returncode, 2023	Eventdata clientprocessid, 2023 04, pc catalina
Other	intelligence, scan	processed 969threadid, security intelligence
Persistence	4622, securitypackagename	securitypackagename windows, eventid 4622
Privilege Escalation	ntssljd, 5f8e	ntssljd correlation, dektop ntssljd

Continuando con la elección del modelo de *Machine Learning*, como se expuso en el marco conceptual, la librería Scikit-learn ofrece diversos modelos, siendo algunos de ellos:

- *Random Forest*: Consiste en combinar una gran cantidad de árboles de decisión independientes entre sí para reducir la varianza. Debido al conjunto de árboles, se le dio el nombre de bosque. (Galindo, 2021)

- *Support Vector Machine* (SVM): Típicamente se usa para problemas de clasificación, pero también se puede usar para regresión. En este algoritmo se construye un hiperplano que separa las clases de datos lo más posible. (Galindo, 2021)

- *Logistic Regression*: Regresión usada principalmente en problemas de clasificación binaria. A pesar de la aparente incongruencia, se trata de una regresión porque el resultado de la ecuación es la probabilidad de que pertenezca a una clase, que dependiendo del umbral que se utilice, se clasifica como positivo o negativo. (Galindo, 2021)

- *Naive Bayes*: Algoritmo basado en la aplicación del teorema de Bayes con la suposición "ingenua" de independencia condicional entre cada par de características dado el valor de la variable de clase. (Scikit-learn, 2023)

Para ver más información de cada modelo y otros, revisar la página oficial de Scikit-learn.

Para la elección del modelo, con el *dataset* cargado en Google Colab, se evaluó la precisión de los algoritmos presentados, de la siguiente manera:

```
models = [
RandomForestClassifier(n_estimators=200, max_depth=3, random_state=0),
LinearSVC(),
MultinomialNB(),
LogisticRegression(random_state=0),
]

CV = 5
cv_dataset = pd.DataFrame(index=range(CV * len(models)))
entries = []
for model in models:
  model_name = model.__class__.__name__
  accuracies = cross_val_score(model, features, labels, scoring='accuracy', cv=CV)
  for fold_idx, accuracy in enumerate(accuracies):
    entries.append((model_name, fold_idx, accuracy))
cv_dataset = pd.DataFrame(entries, columns=['model_name', 'fold_idx', 'accuracy'])
```

Figura 28: Código Google Colab para evaluación de modelos.
Fuente: Elaboración propia.

Con los modelos evaluados, se procede a visualizar su precisión mediante los diagramas de caja o también llamados bigotes:

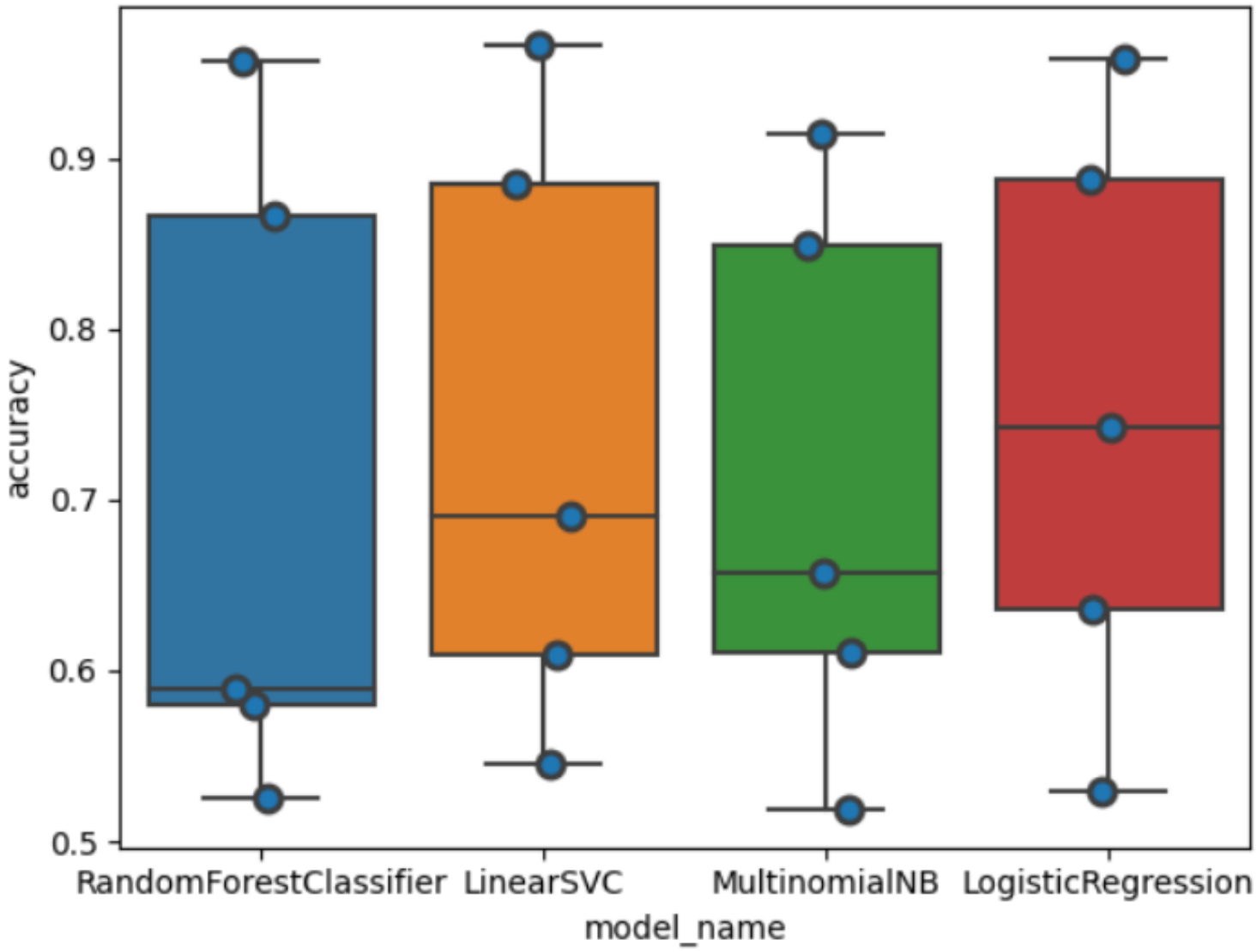


Figura 29: Diagramas de cajas de evaluación de modelos.
Fuente: Elaboración propia.

Obteniendo los siguientes valores de *accuracy*:

Tabla 5: *Accuracy* de modelos de *Machine Learning*.
Fuente: Elaboración propia.

Modelo	***Accuracy***
Random Forest Classifier	0.70
Linear Support Vector Machine	0.74
Multinomial Naive Bayes	0.71
Logistic Regression	0.75

Si bien el modelo con mayor precisión es el de Regresión Logística, por las definiciones previas entregadas y la investigación realizada, considerando además que en el futuro el *dataset* entregado será mayor, se optó por utilizar el modelo *Support Vector Machine.*

Las ventajas del modelo son:

- Eficaz en espacios de altas dimensiones.
- Todavía efectivo en casos donde el número de dimensiones es mayor que el número de muestras.
- Utiliza un subconjunto de puntos de entrenamiento en la función de decisión (llamados vectores de soporte), por lo que también es eficiente en memoria.
- Versátil: se pueden especificar diferentes funciones del kernel[19] para la función de decisión. Se proporcionan kernels comunes, pero también es posible especificar kernels personalizados.

Las desventajas del modelo son:

- Si el número de funciones es mucho mayor que el número de muestras es crucial evitar el ajuste excesivo al elegir las funciones del núcleo y el término de regularización.
- Las SVM no proporcionan directamente estimaciones de probabilidad, estas se calculan utilizando un costoso método cruzado de cinco veces.

(Scikit-learn, 2023)

Con el modelo decidido, se prosiguió a programarlo dentro de Google Colab, con el siguiente código:

```
model = LinearSVC()

X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features, labels, dataset.index, test_size=0.2, random_state=0)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

X_train, X_test, y_train, y_test = train_test_split(dataset['Log'], dataset['Label'], test_size=0.2, random_state = 0)
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)

prediction = LinearSVC().fit(X_train_tfidf, y_train)
```

Figura 30: Código modelo LinearSVC.
Fuente: Elaboración propia.

Como se puede apreciar en la imagen, uno de los parámetros entregados a la función *train_test_split* es el de *test_size*, indicándole que utilice el 20 % del dataset para testear y el 80 % para el entrenamiento. Finalmente, para realizar una predicción basta con utilizar la función *precition* como *prediction.predict(count_vect.transform(log)),* en donde *log* es el registro de eventos a analizar.

[19] En *Machine Learning* es un método para el análisis de patrones, cuyo miembro más conocido son las *Support Vector Machine*.

3.3 GENERAR ALERTAS Y REPORTES A PARTIR DEL ANÁLISIS DE LOGS

Con el modelo de *Machine Learning* creado en la subsección anterior, se procede a desarrollar una interfaz gráfica con la librería Tkinter, para presentar al usuario una aplicación con la cual interactúe. La vista principal del software se enseña a continuación:

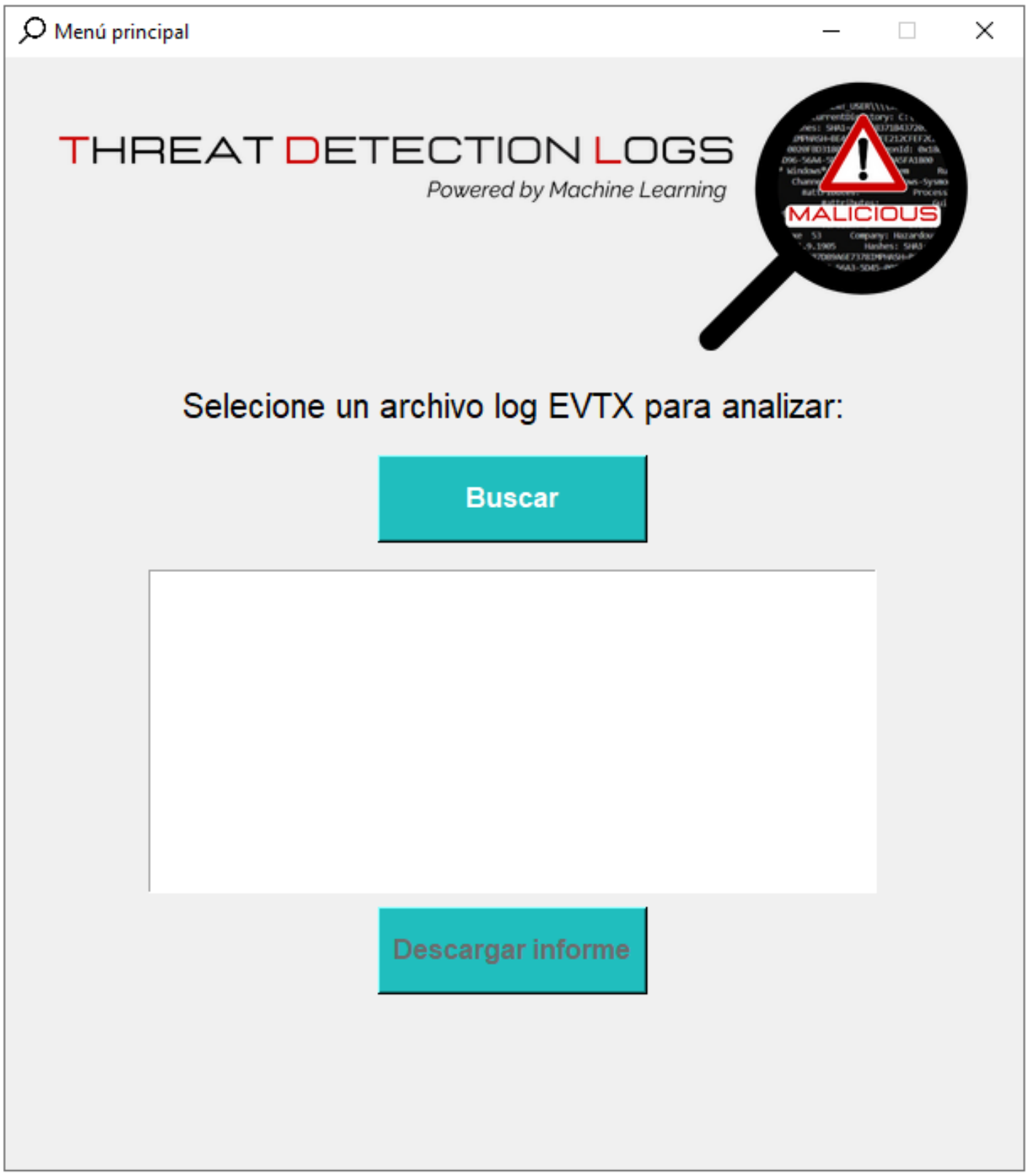


Figura 31: Menú principal aplicación ThreatDetectionLogs.
Fuente: Elaboración propia.

3.3.1 DESCRIPCIÓN DEL CÓDIGO

El código desarrollado se divide en 4 archivos Python, los cuales son:

- TDL.py
- ML_Events.py
- ML_Specific.py
- Report.py

A continuación, se da lugar a la explicación de cada archivo.

TDL.py

Este archivo corresponde a la base del programa, ya que es el que importa los demás archivos y donde se desarrolla la interfaz. Por medio de la librería Tkinter se definen parámetros como el tamaño de la venta, que no sea reajustable, el ícono, la ubicación de los botones, entre otros, y por parte del funcionamiento, se definen las variables globales y las funciones, de estas últimas destacando las siguientes:

- *open_file():* Función que permite al usuario seleccionar entre sus documentos un registro de eventos, es decir un archivo con extensión evtx, una vez que hace click en el botón Buscar.

- *parser_file():* Función que recibe como parámetro el registro de eventos, lo abre mediante la librería evtx y almacena cada evento como un *string* [20] , para posteriormente llamar a la función *result_analysis().*

- *result_analysis():* Función que recibe el *string* que contiene el evento, realiza la predicción del evento.

- *show_results():* Función que muestra el resultado de la predicción, y habilita el botón para descargar el informe.

- *download_report():* Función que al hacer click en el botón de “Descargar informe” pregunta al usuario la ubicación para realizar la descarga.

[20] En cualquier lenguaje de programación es una secuencia de caracteres usado para representar el texto.

ML_Events.py

Archivo que contiene el código desarrollado en el Google Colab para el modelo de *Machine Learning* entrenado con en el *dataset* CSV_Events.csv.

ML_Specific.py

Archivo que contiene una función que recibe cómo parámetros el registro de eventos analizado y el *dataset* respectivo. La función contenida en este archivo es llamada si la predicción realizada con el modelo general contiene al menos un tipo de *log* que puede ser especificado (Ver figura 25). En caso de poder entregar una predicción más específica, es decir el subtipo de ataque, se entrena el modelo de *Machine Learning* con el *dataset* correspondiente, manteniendo los porcentajes, 80 % de entrenamiento y 20 % de test.

Report.py

Archivo que contiene la función que gestiona la creación del informe, tanto del pdf como del Excel, a partir de diversos parámetros cómo el resultado de la predicción, la precisión, el total de datos del *dataset*, entre otros.

3.3.2 FUNCIONAMIENTO DEL CÓDIGO

Previamente se enseñó la vista principal de la aplicación, a continuación, se procede a mostrar el funcionamiento completo de la aplicación a partir del diagrama de flujo presentado inicialmente (Ver Figura 12).

En un inicio la única acción que puede realizar el usuario es cargar un *log*, ya que el botón para descargar el informe se encuentra deshabilitado. Haciendo click en el botón de buscar se despliega una ventana emergente en la que el usuario puede seleccionar desde su dispositivo un archivo con extensión evtx.

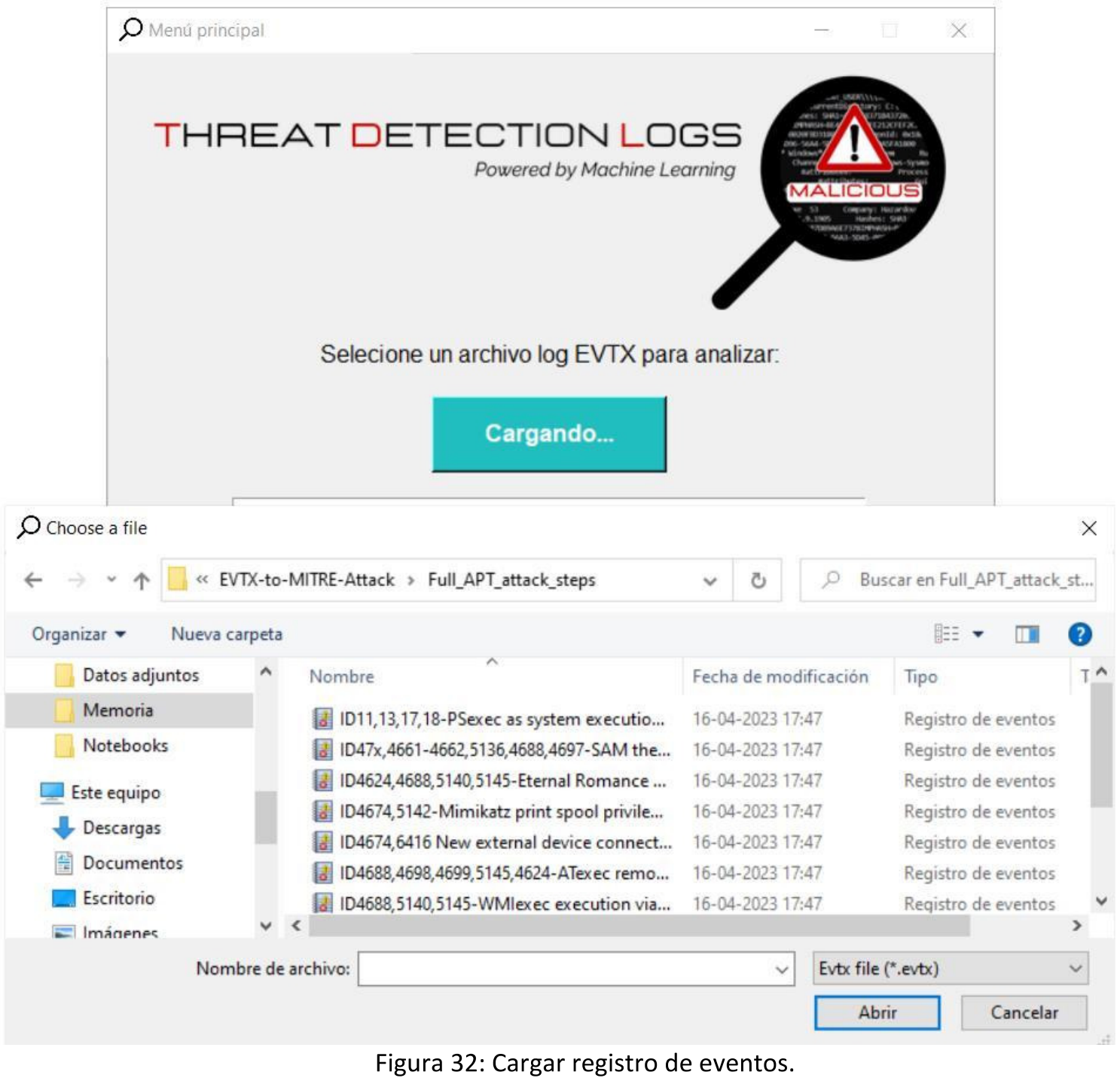


Figura 32: Cargar registro de eventos.
Fuente: Elaboración propia.

Posterior a la selección de un registro de eventos, el botón de buscar cambia su estado a "Cargando", debido a que dependiendo del tamaño del archivo el programa puede llegar a tardar varios segundos en mostrar el resultado. Cómo se explicó anteriormente, un *log* al estar constituido por un conjunto de eventos puede presentar más de un tipo de ataque, de manera que el resultado mostrado en pantalla se ve de la siguiente manera:

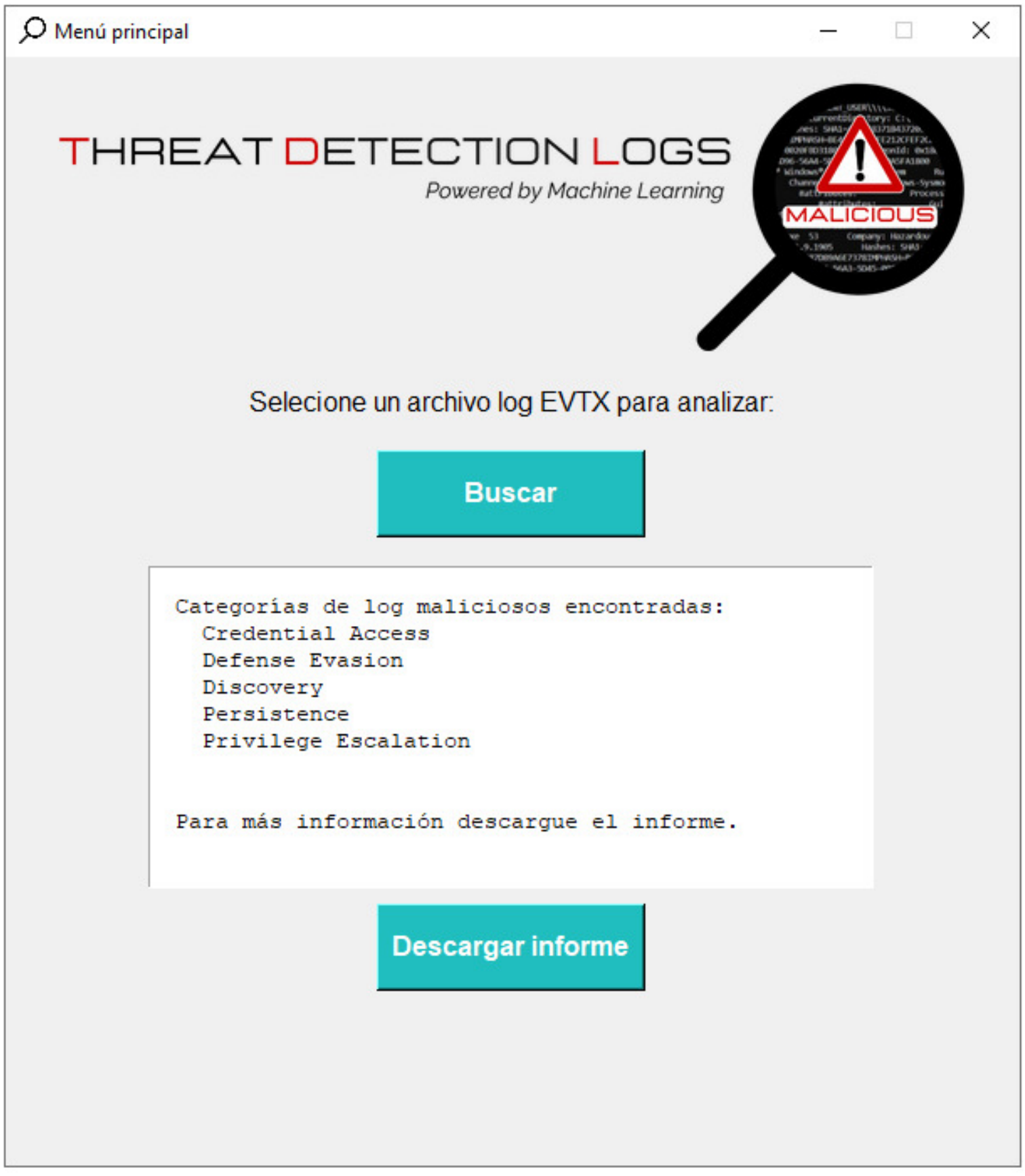


Figura 33: Mostrar predicción.
Fuente: Elaboración propia.

Una vez que el usuario ve los resultados tiene la opción de cargar otro registro de eventos, en un caso en que por ejemplo no se encuentre ningún tipo de ataque, o descargar un informe detallado con los resultados.

En caso de hacer click en el botón para descargar el informe, nuevamente se despliega una ventana emergente, pero en esta ocasión para que el usuario escoja el lugar de destino (dónde se guardará) y el nombre del informe. Una vez ingresados estos parámetros comienza la descarga y esta muestra su estado de finalización mediante una ventana emergente confirmando la descarga correcta:

Figura 34: Confirmación descarga de informe.
Fuente: Elaboración propia.

El informe se compone de dos archivos, un documento pdf y un archivo Excel.

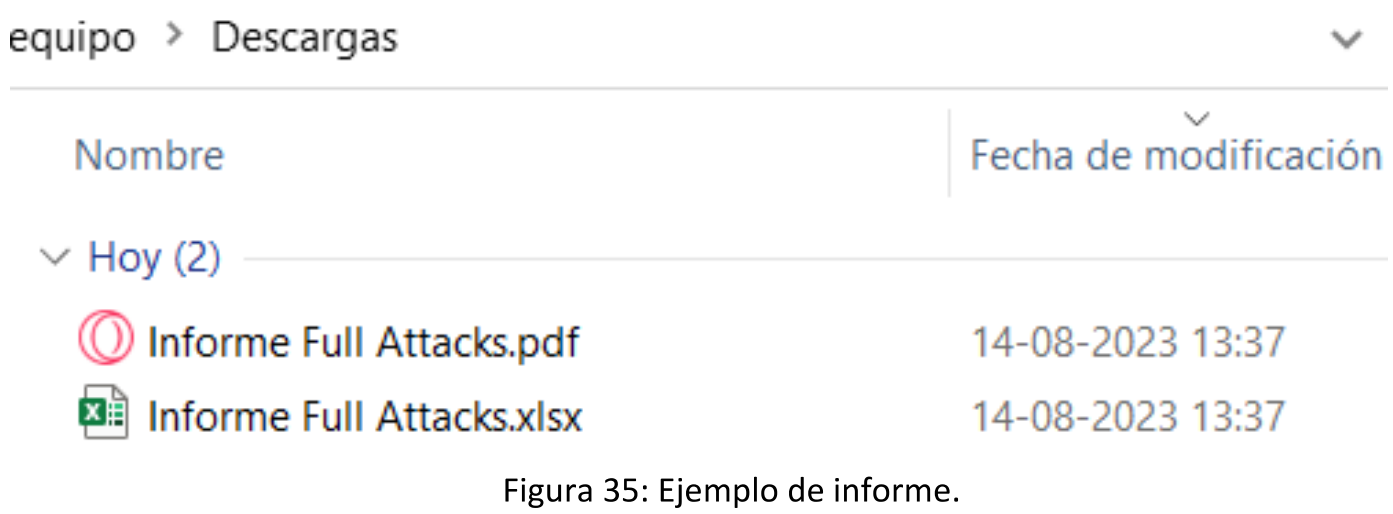


Figura 35: Ejemplo de informe.
Fuente: Elaboración propia.

El documento pdf está compuesto de 4 páginas, en donde se detallan los tipos de ataque que se encontraron dentro del *log*, la cantidad, como se distribuyen porcentualmente e información del modelo predictivo utilizado.

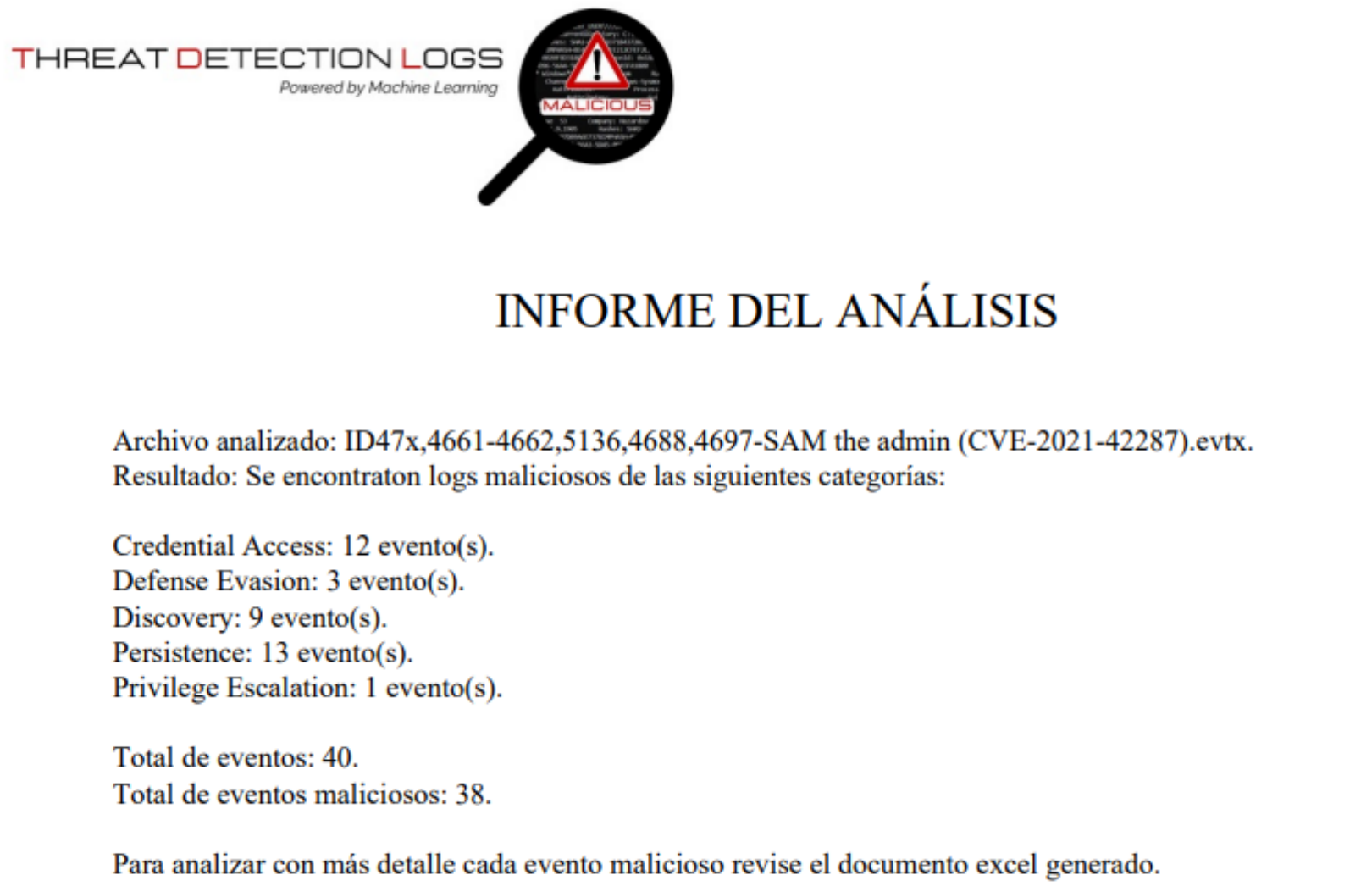
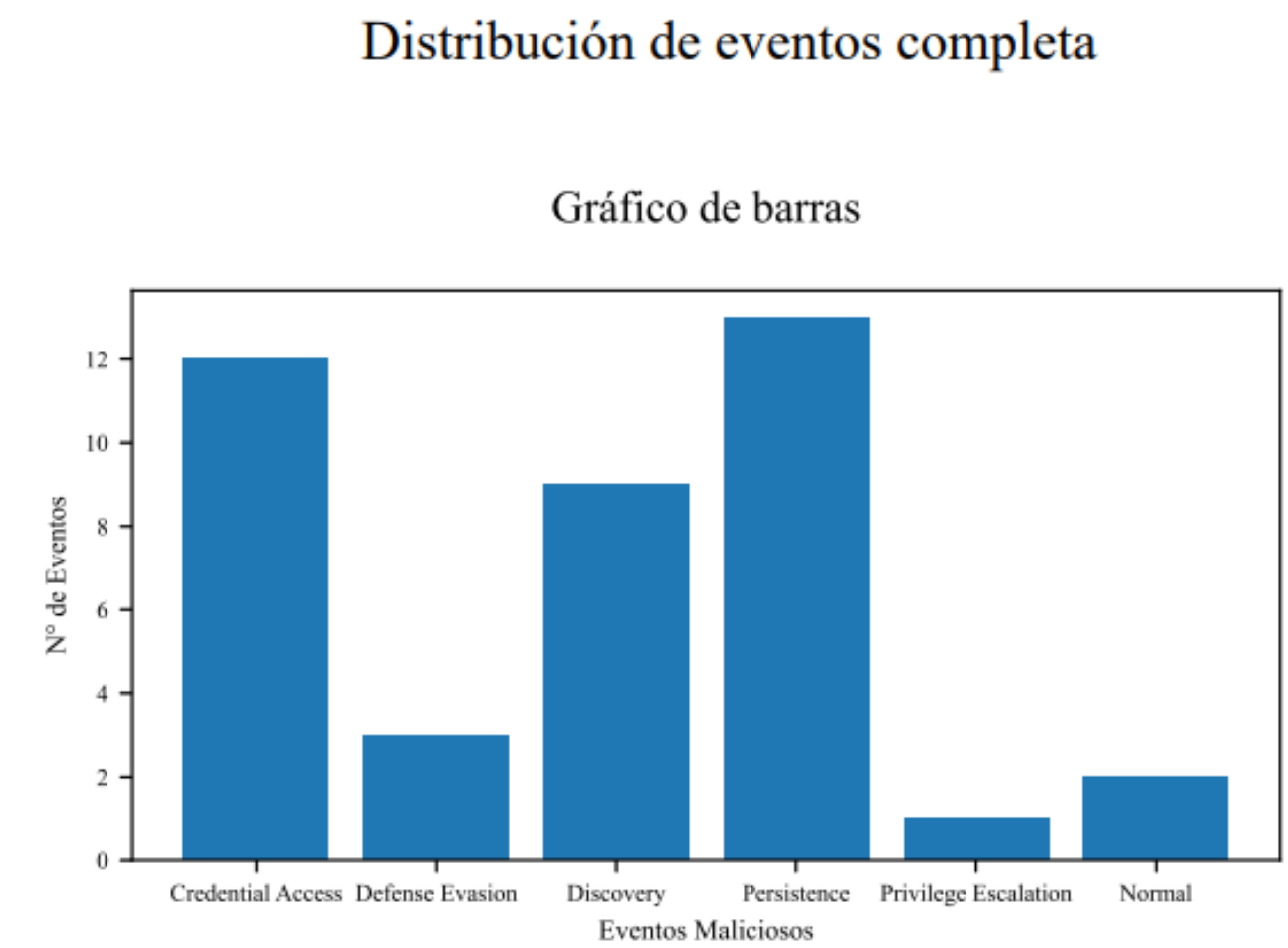


Figura 36: Página 1 Informe pdf.
Fuente: Elaboración propia.



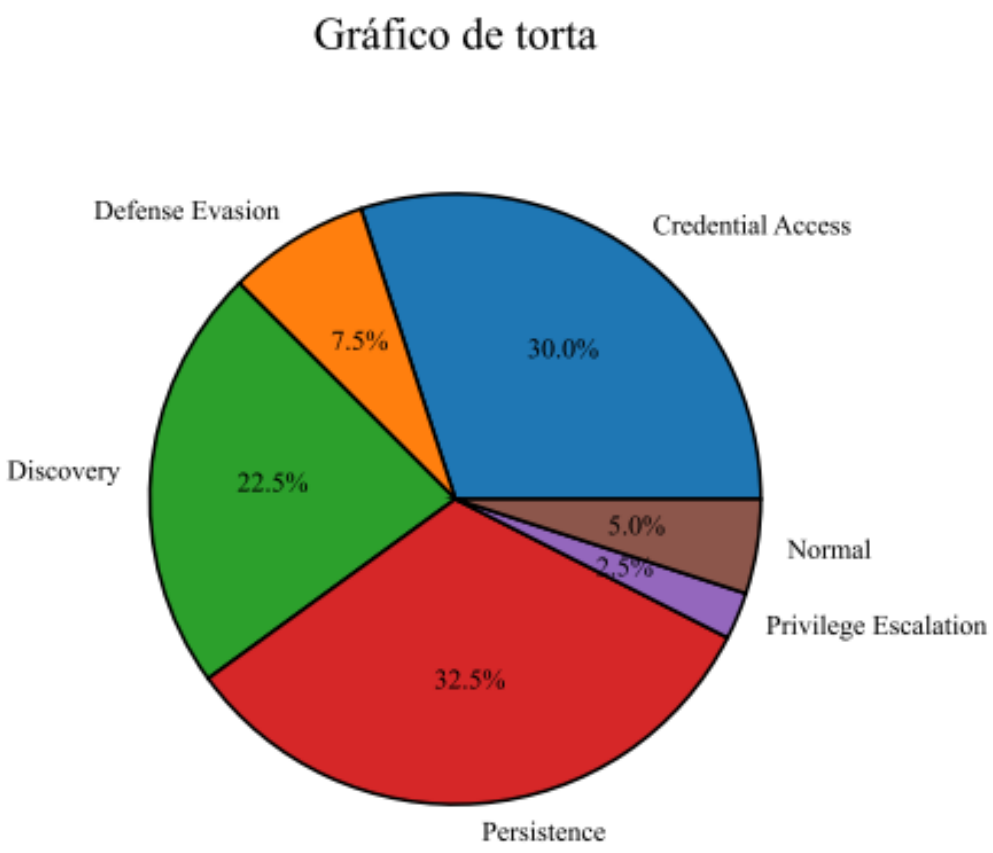
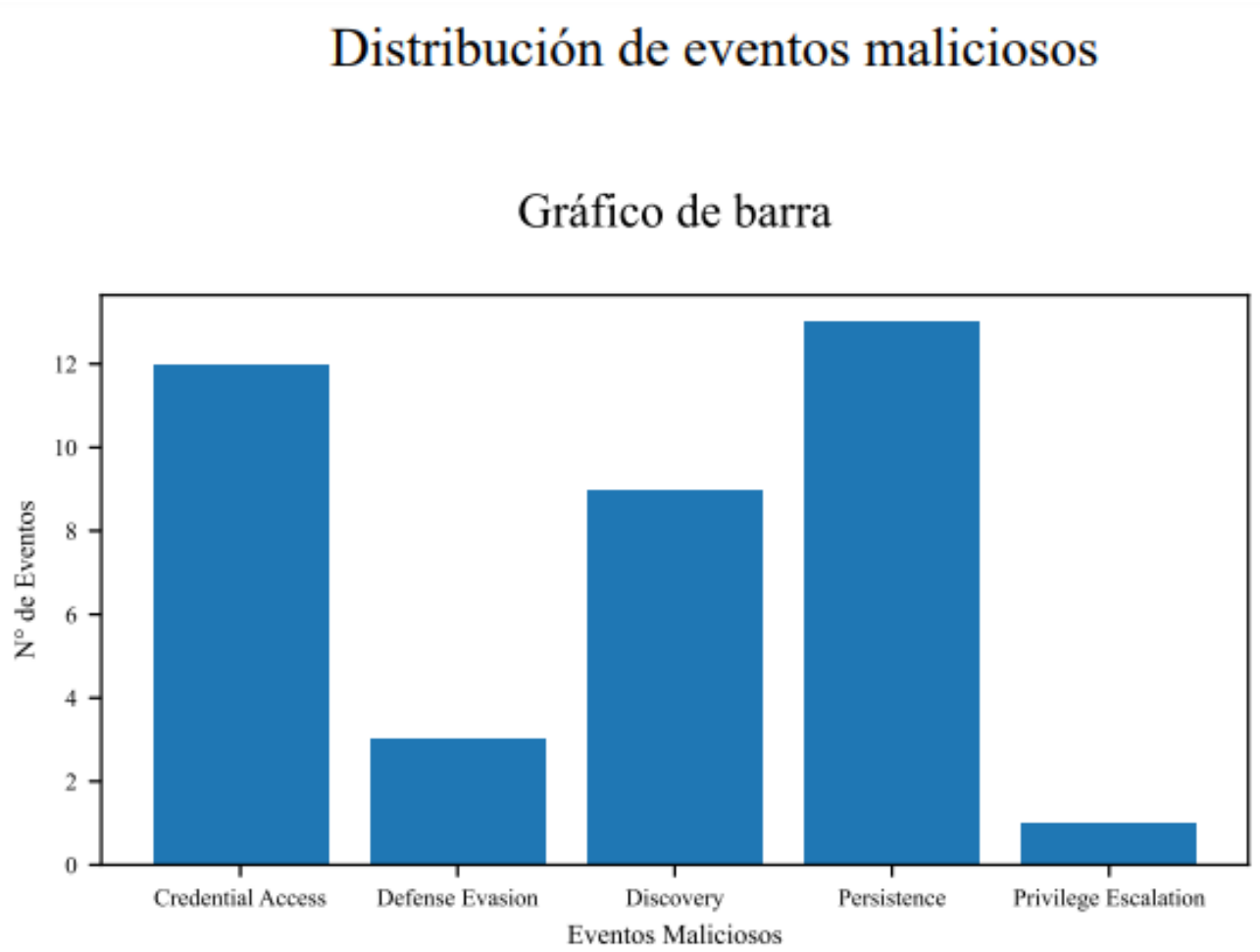


Figura 37: Página 2 Informe pdf.
Fuente: Elaboración propia.

En esta página se visualiza gráficamente la distribución completa de los eventos, incluyendo los que tienen un comportamiento normal.



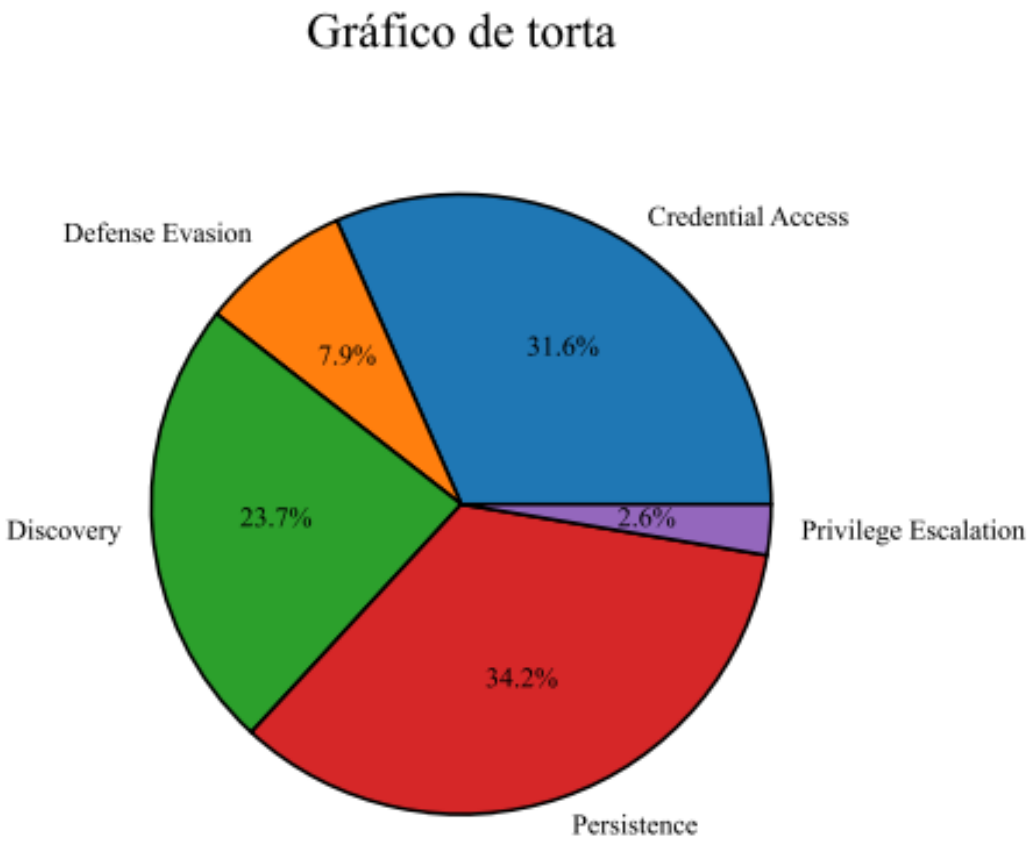


Figura 38: Página 3 Informe pdf.
Fuente: Elaboración propia.

En esta página se visualiza gráficamente la distribución de los eventos maliciosos, con el fin de identificar que tipos de ataques son los más utilizados contra la empresa.

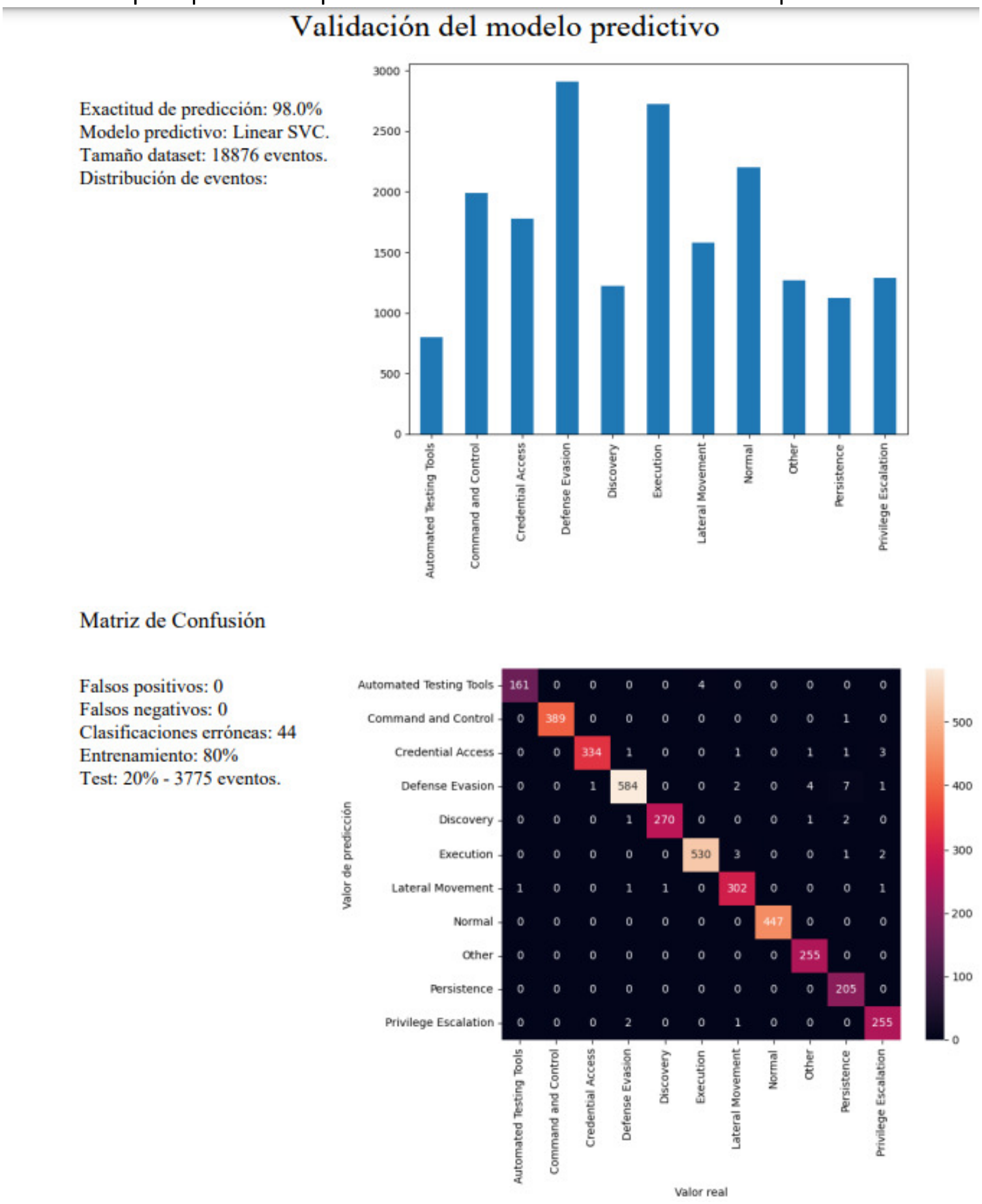


Figura 39: Página 4 Informe pdf.
Fuente: Elaboración propia.

En esta página se enseña información relacionada al modelo de Machine Learning utilizado, como la distribución de eventos del dataset para entrenar el algoritmo, la precisión del mismo, y su validación mediante la matriz de confusión.

Por otro lado, el informe Excel tiene el objetivo de detallar las líneas en dónde se encontraron los eventos maliciosos, y en caso de haber sido posible, explicitar el subtipo de ataque, indicando la propia precisión de este segundo algoritmo.

	A	B	C	D	E
1	Tipo de evento	Evento N°	Tipo específico del evento	Exactitud de predicción específica	Contenido de evento
2	Credential Access	1	Credentials from Password Stores	88.0%	Event: #attributes:
3	Discovery	2	Network Share Discovery	95.0%	Event: #attributes:
4	Credential Access	3	Credentials from Password Stores	88.0%	Event: #attributes:
5	Discovery	4	Network Share Discovery	95.0%	Event: #attributes:
6	Credential Access	5	Credentials from Password Stores	88.0%	Event: #attributes:
7	Credential Access	6	Credentials from Password Stores	88.0%	Event: #attributes:
8	Credential Access	7	Credentials from Password Stores	88.0%	Event: #attributes:
9	Credential Access	8	Credentials from Password Stores	88.0%	Event: #attributes:
10	Credential Access	9	Credentials from Password Stores	88.0%	Event: #attributes:
11	Defense Evasion	10	Obfuscated Files or Information	92.0%	Event: #attributes:
12	Defense Evasion	11	Obfuscated Files or Information	92.0%	Event: #attributes:
13	Defense Evasion	12	Obfuscated Files or Information	92.0%	Event: #attributes:
14	Persistence	13	Boot or Logon Autostart Execution	91.0%	Event: #attributes:

Figura 40: Informe Excel.
Fuente: Elaboración propia.

Adicionalmente, se agrega el contenido del evento para que pueda ser revisado inmediatamente, aunque estando en una celda de Excel no resulta lo más cómodo, por lo que lo óptimo es abrir el registro de eventos con el Visor de Eventos de Windows y dirigirse al evento puntual.

3.3.3 CREACIÓN DE EJECUTABLE

Con el código funcionando en su totalidad, la etapa final consiste en crear un archivo ejecutable, cuya extensión es .exe. Para ello se hace uso de la librería auto-py-to-install descrita con anterioridad. Para utilizarla se debe llamar desde la terminal con el comando auto-py-to-install, de modo que se abre una pestaña en el navegador como la que se visualiza a continuación:

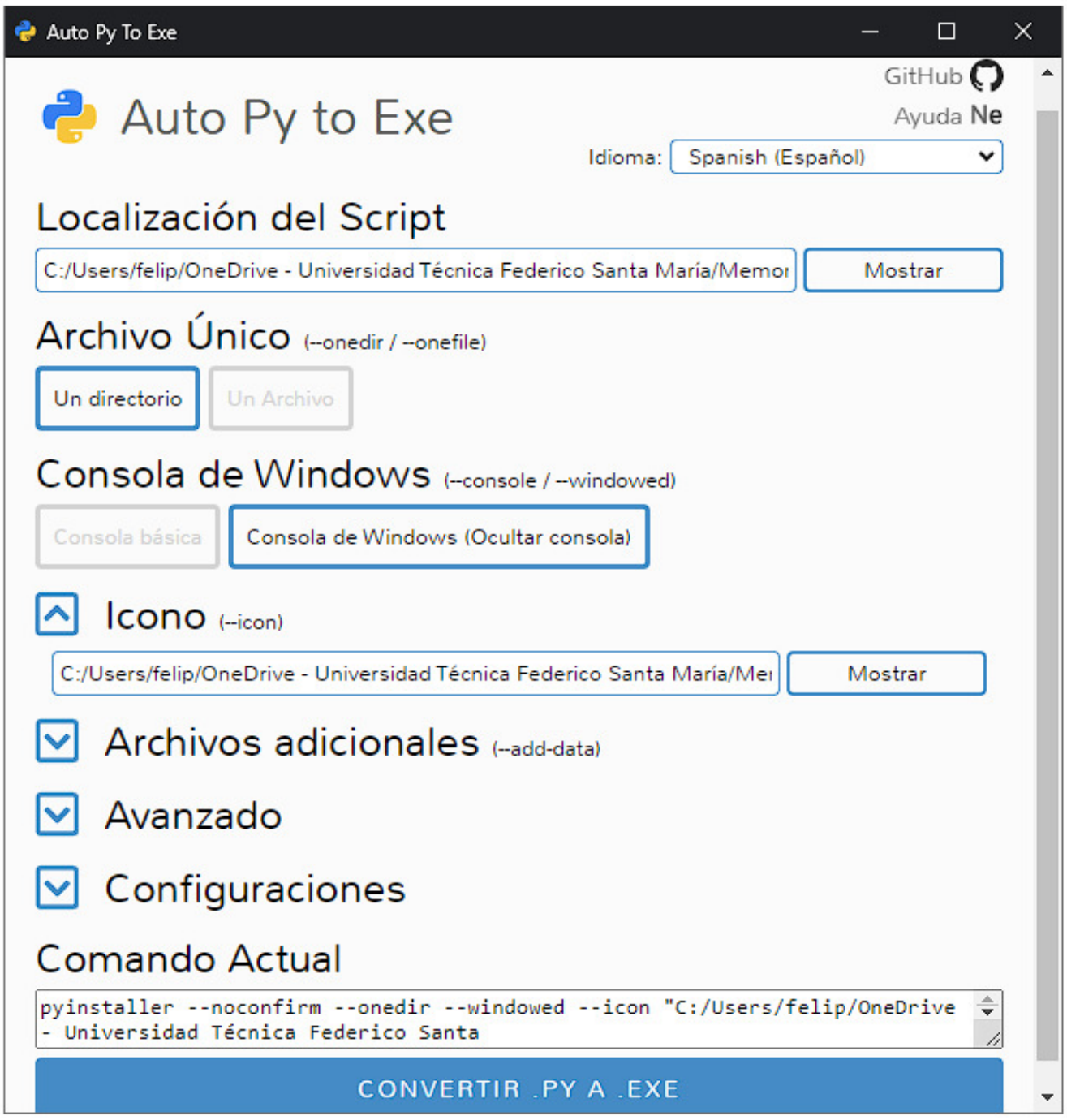


Figura 41: Auto-py-to-install.
Fuente: Elaboración propia.

En esta pestaña se cargan todos los componentes que integran la aplicación, es decir los archivos Python, los *datasets* y las imágenes. El resultado es una carpeta que contiene diversos archivos para el funcionamiento de la aplicación, destacando el archivo con extensión .exe, ya que corresponde al ejecutable, pudiendo crear un acceso directo de él.

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN

En el capítulo anterior se presentó la propuesta de solución, abarcando los primeros 3 objetivos específicos definidos en un comienzo. Comenzando con la definición de un problema de ciberseguridad cómo lo es el monitoreo de registros de eventos del sistema operativo, el diseño de un modelo de *Machine Learning* y la implementación del mismo mediante una interfaz gráfica para generar alertas y reportes a partir del análisis.

En el presente capítulo se tiene el objetivo de validar la solución propuesta, para lo que se plantean dos perspectivas definidas en los últimos objetivos específicos, siendo estos evaluar el algoritmo a partir de *logs* maliciosos conocidos y mediante métricas propias de la inteligencia artificial.

4.1 EVALUAR EL FUNCIONAMIENTO Y DESEMPEÑO DEL ALGORITMO

El primer criterio utilizado para validar la solución se basa en evaluar el algoritmo en base a su funcionamiento, es decir, la capacidad de predecir la categoría de un registro de eventos. Para esto ha utilizado un registro de eventos cuya categoría es conocida con anterioridad, y que no se encuentran dentro del *dataset* utilizado para el entrenamiento del modelo.

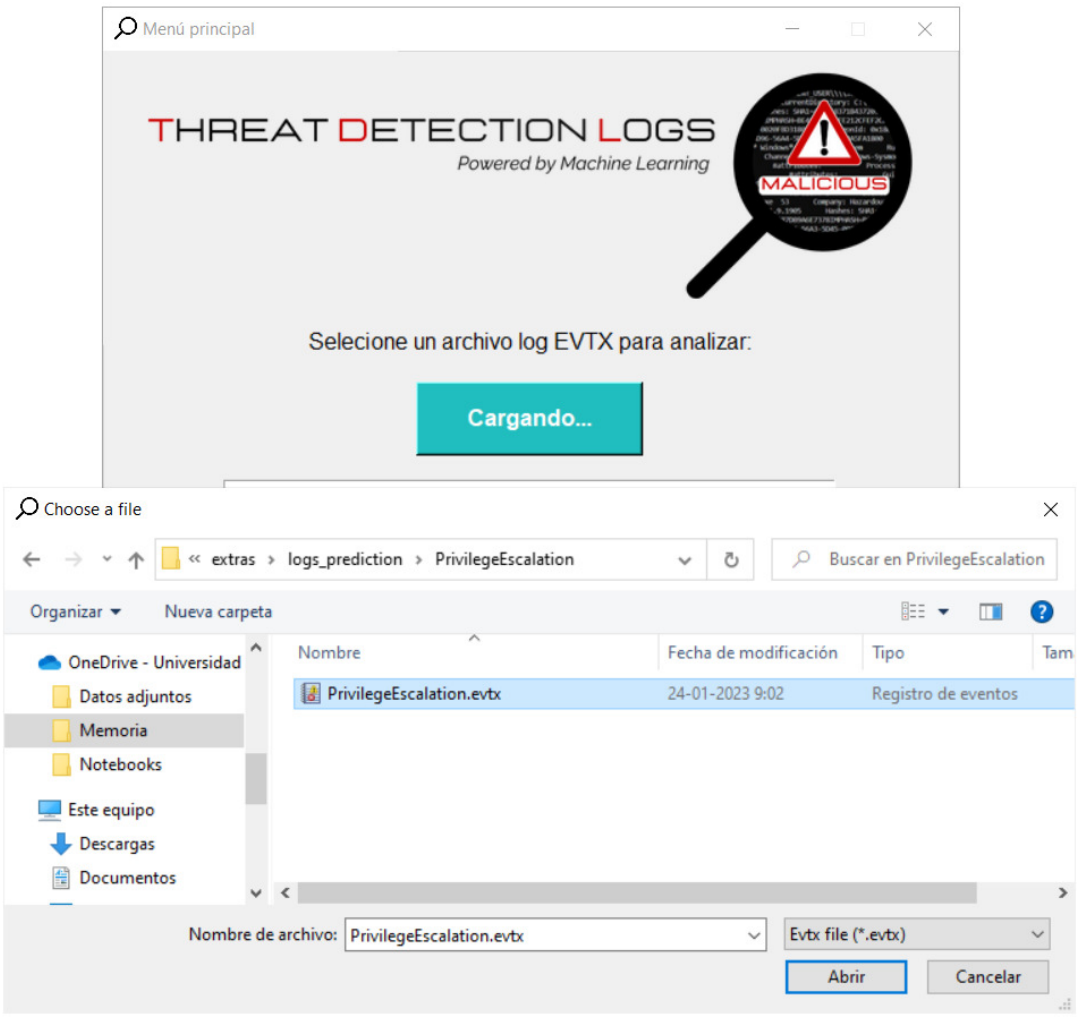


Figura 42: Carga registro de eventos *PrivilegeEscalation*.
Fuente: Elaboración propia.

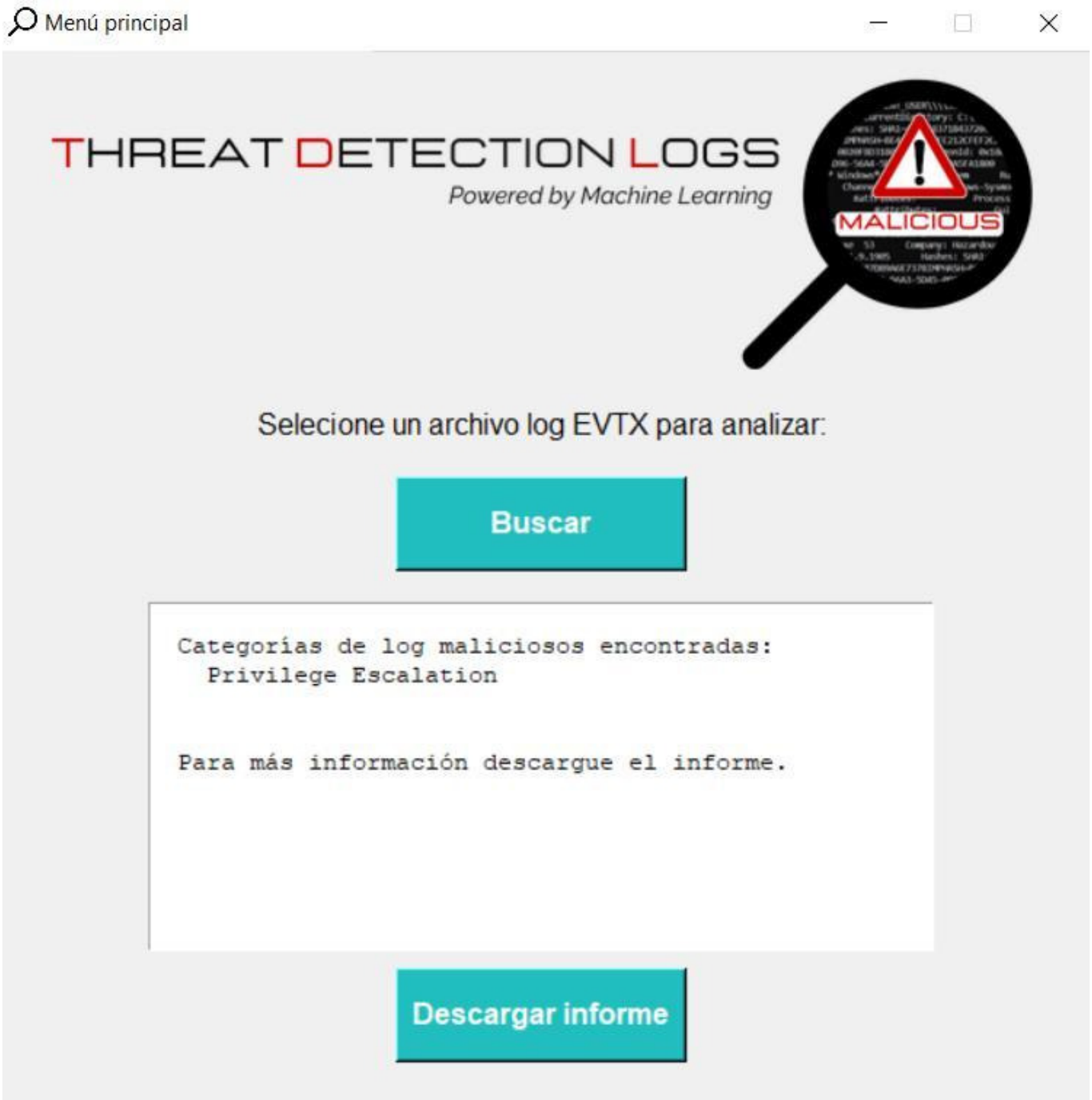


Figura 43: Resultado del análisis del *log PrivilegeEscalation*.
Fuente: Elaboración propia.

Como se puede apreciar, la aplicación funciona correctamente, identificando con la categoría de "*Privilege Escalation*" el registro de eventos seleccionado.

Una característica por recordar es que un registro de eventos está compuesto por diversos eventos, lo que da lugar a tener la presencia de múltiples ataques. Es por ello que una forma para evaluar el desempeño del algoritmo es entregarle un archivo *log* con una gran cantidad de eventos, para comprobar que puede detectar más de una categoría de evento malicioso y que soporta archivos pesados.

El repositorio del usuario mdecrevoisier descargado contiene una carpeta con el nombre "*EVTX full APT attack steps*", que como su nombre indica corresponde a registros de eventos con diversos ataques.

Nombre	Fecha de modificación	Tipo	Tamaño
ID11,13,17,18-PSexec as system executio...	16-04-2023 17:47	Registro de eventos	68 KB
ID47x,4661-4662,5136,4688,4697-SAM th...	16-04-2023 17:47	Registro de eventos	68 KB
ID4624,4688,5140,5145-Eternal Romance...	16-04-2023 17:47	Registro de eventos	68 KB
ID4674,5142-Mimikatz print spool privile...	16-04-2023 17:47	Registro de eventos	68 KB
ID4674,6416 New external device connec...	16-04-2023 17:47	Registro de eventos	68 KB
ID4688,4698,4699,5145,4624-ATexec rem...	16-04-2023 17:47	Registro de eventos	68 KB
ID4688,5140,5145-WMIexec execution vi...	16-04-2023 17:47	Registro de eventos	1.092 KB
ID4720,4698-Fortinet APT group abuse o...	16-04-2023 17:47	Registro de eventos	68 KB
ID4742,4935,4662,4661,5137-DCshadow ...	16-04-2023 17:47	Registro de eventos	68 KB
ID5140-5145,4688,4697-Encrypted paylo...	16-04-2023 17:47	Registro de eventos	1.092 KB
ID5145-4624-DonPAPI full extraction.evtx	16-04-2023 17:47	Registro de eventos	1.092 KB

Figura 44: Carpeta *EVTX full APT steps*.
Fuente: DATASOURCE.AI.

Dentro de los registros de eventos disponibles, el primer parámetro para elegir es el peso del archivo, y en segundo lugar la cantidad de eventos que posee. En este caso, el último *log* llamado “ID5145-4624-DonPAPI full extraction.evtx” tiene un peso de 1092 KB y un total de 750 eventos.

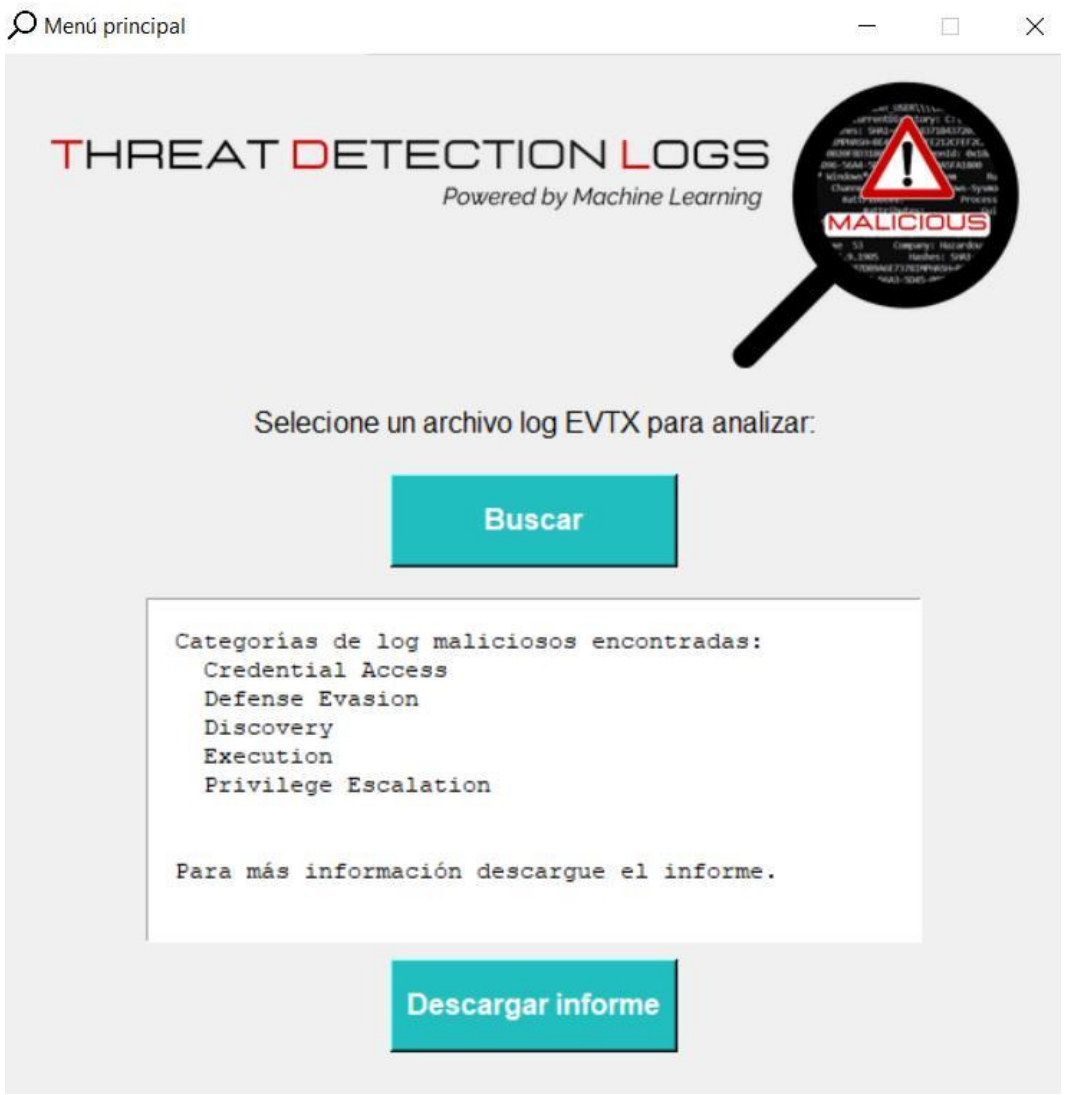


Figura 45: Resultado análisis de *log* con múltiples ataques.
Fuente: Elaboración propia.

Cómo se puede apreciar en la figura anterior, el modelo es capaz de predecir múltiples ataques. Otro dato interesante es que, con el peso del archivo, el análisis tardó en promedio 20 minutos de ejecución. El tiempo realmente no es influyente para la solución propuesta, dado que el objetivo es identificar la mayor cantidad de eventos maliciosos de manera correcta, no obstante, es importante usuario sea consciente del tiempo que puede tardar un análisis. De igual modo se debe mencionar que este tiempo está relacionado a que el evento cargado posee gran cantidad de eventos maliciosos. Al probar el modelo con un archivo de 13.000 registros de eventos con un comportamiento normal, tardó segundos.

4.2 EVALUAR EL ALGORITMO A PARTIR DE MÉTRICAS DE LA INTELIGENCIA ARTIFICIAL

Es sabido que en los algoritmos de *Machine Learning* jamás se deben compartir los datos asignados al entrenamiento y al testeo. Es gracias a los datos destinados al testeo que la solución es validada. Gracias a la implementación de la librería Scikit-learn, esta separación es automatizada, y provee funciones que entregan información relevante a esta subsección.

Previamente fue justificada la elección del modelo *Linear Support Vector Machine* por lo que a continuación se procede a evaluar dicho modelo. En una evaluación inicial en comparación a los demás algoritmos se obtuvo una exactitud (*Accuracy*) de 74 %. No obstante, en ese punto no estaba implementado los algoritmos en completitud, por lo que a continuación se procede a calcular no solo la exactitud real, sino otras métricas relevantes.

En primer lugar, resulta importante mencionar que el modelo presentado no posee falsos positivos ni falsos negativos. Un falso positivo sucede cuando un antivirus no detecta un malware, cuando deja pasar o ejecutarse un código malicioso en el sistema que está protegiendo. Por otro lado, el falso positivo se da cuando el antivirus, por error, identifica como malware un fichero que es legítimo e inofensivo (Hispasec, 2008).

En la solución propuesta, un falso positivo tendría lugar si el modelo reconociera un *log* malicioso como uno normal. Por otra parte, un falso positivo sucedería si el algoritmo no alertara de un *log* malicioso al clasificarlo como normal. Para identificar estos problemas, se hace uso de la matriz de confusión.

En el campo de la inteligencia artificial y el aprendizaje automático una matriz de confusión es una herramienta que permite visualizar el desempeño de un algoritmo de aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. En términos prácticos nos permite ver qué tipos de aciertos y errores está teniendo nuestro modelo a la hora de pasar por el proceso de aprendizaje con los datos**.** (Arce, 2019)

A continuación, se presenta una matriz de confusión basada en un algoritmo de clasificación binaria, es decir una matriz de 2x2, con el fin de comprender de manera sencilla su funcionamiento:

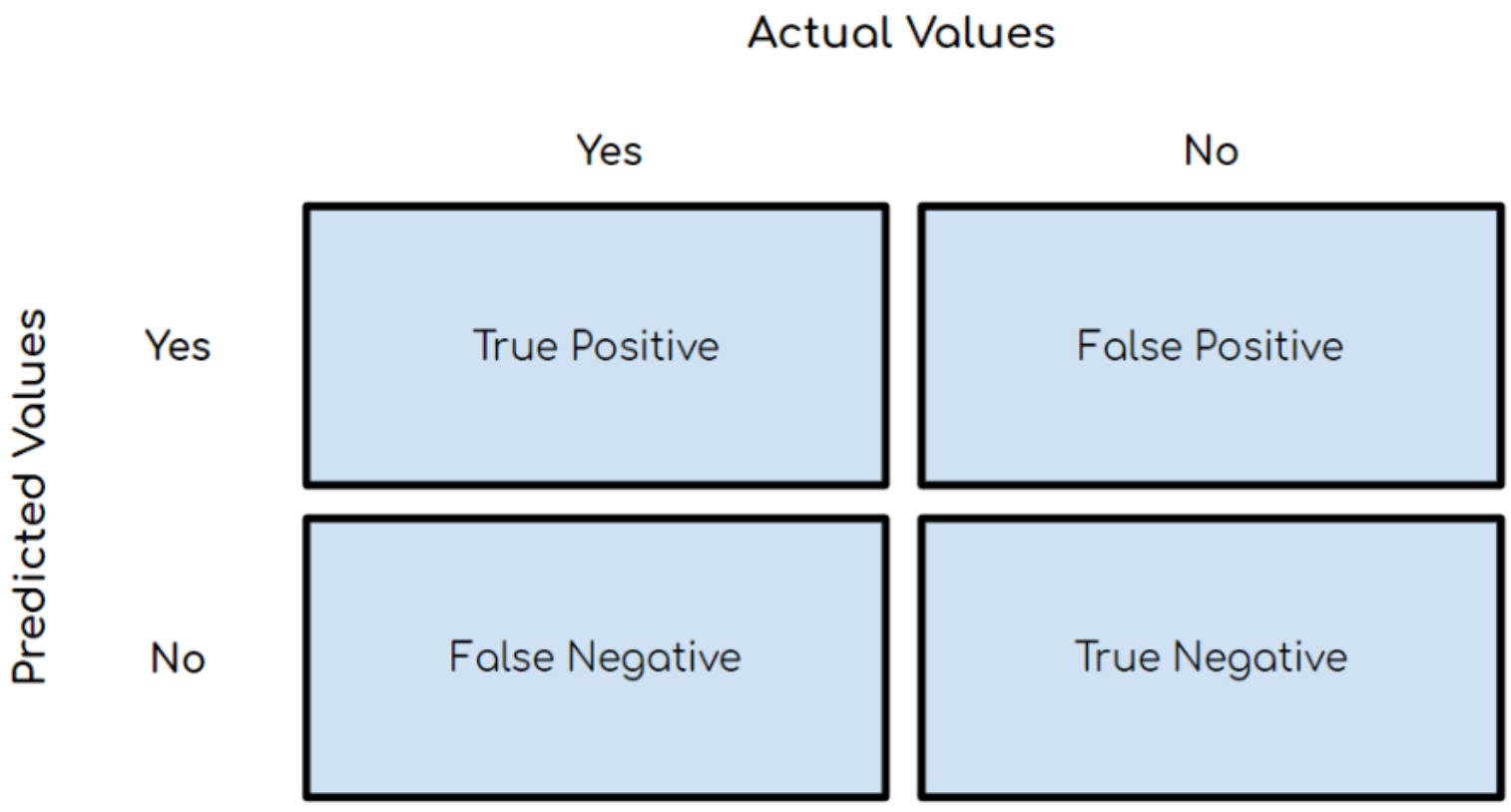


Figura 46: Matriz de confusión binaria.
Fuente: DATASOURCE.AI.

Para las métricas posteriores resulta necesario conocer los siguientes conceptos:

- **Positivo (P):** La observación es positiva.
- **Negativo (N):** La observación no es positiva.
- **Verdadero Positivo (TP):** Resultado en el que el modelo predice correctamente la clase positiva.
- **Verdadero Negativo (TN):** Resultado donde el modelo predice correctamente la clase negativa.
- **Falso Positivo (FP):** También llamado error de tipo 1, resultado donde el modelo predice incorrectamente la clase positiva cuando en realidad es negativa.
- **Falso Negativo (FN):** También llamado error de tipo 2, un resultado en el que el modelo predice incorrectamente la clase negativa cuando en realidad es positiva.

De modo que TP + TN + FP + FN es el total de las predicciones.

La razón por la que se expone que el modelo propuesto no presenta falsos positivos ni falsos negativos se respaldada por la matriz de confusión extraída gracias a Google Colab, mostrada a continuación, siendo una matriz de 11x11.

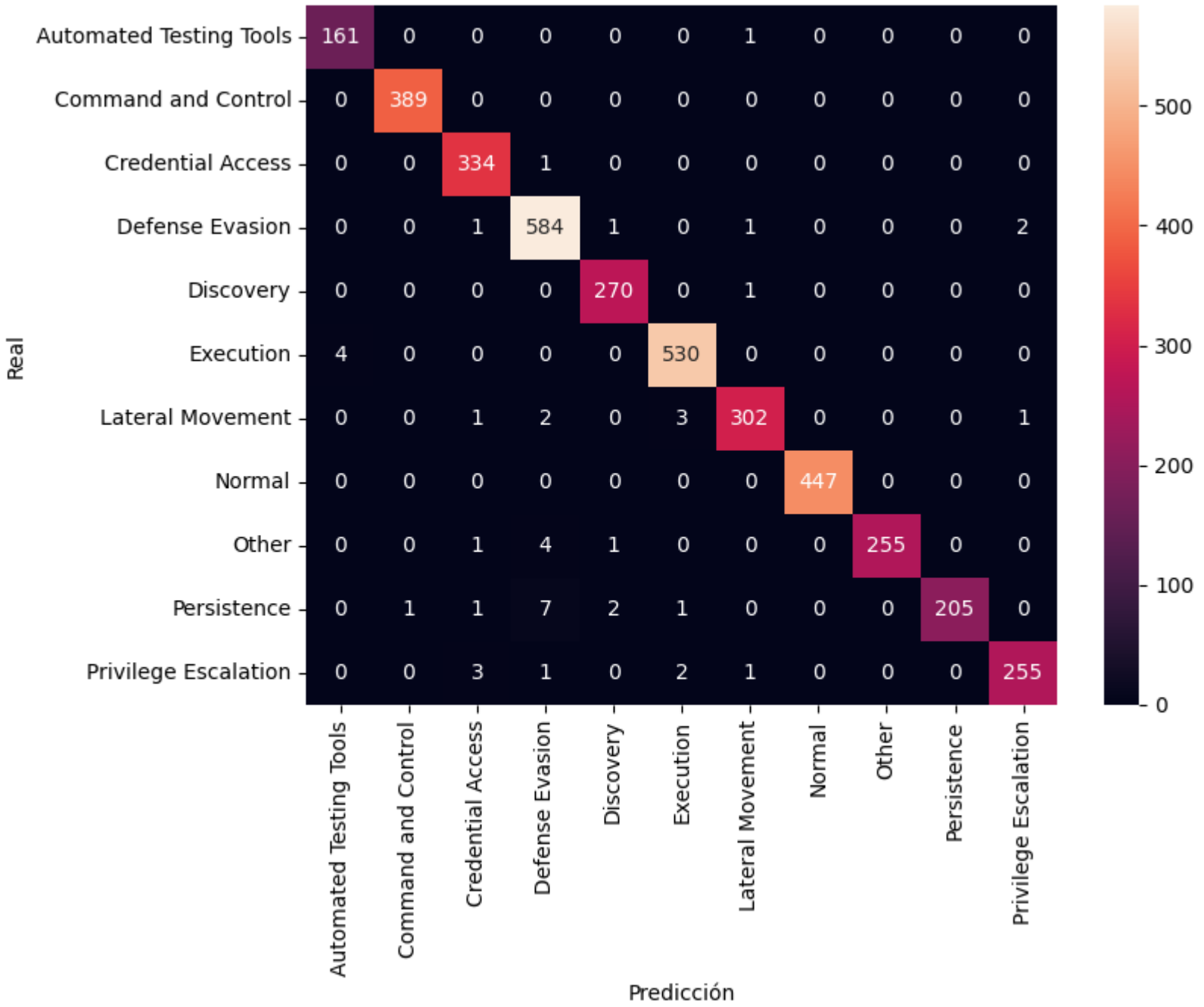


Figura 47: Matriz de confusión.

Fuente: Elaboración propia.

Afortunadamente, la aparición de los falsos positivos no es muy frecuente (sobre todo si tenemos en cuenta la inmensa cantidad de archivos que un antivirus puede analizar) y las compañías desarrolladoras implementan controles de calidad para evitarlos. En cualquier caso, y como comentamos al principio, ningún fabricante está libre de sufrir este problema, que en mi opinión viene a demostrar el desafío que supone el desarrollo de un producto antimalware (Guerrero, 2010). Mostrando así que con la solución propuesta se ha abordado un desafío importante tanto dentro del área de la ciberseguridad como del *Machine Learning*.

A partir de la matriz de confusión se pueden calcular otras métricas:

- Exactitud: Proporción de predicciones que el modelo clasificó correctamente. Se calcula mediante la fórmula:

$$Accuracy = \frac{\text{N° de predicciones correctas}}{N° \; total \; de \; predicciones} = \frac{\text{TP} + \text{TN}}{TP + TN + FP + FN}$$

- Tasa de error: Lo opuesto a la Exactitud, es decir, la proporción de predicciones que el modelo clasificó incorrectamente. Se calcula mediante la fórmula:

$$Missclasification\, rate = \frac{FP + FN}{TP + TN + FP + FN} = 1 - Accuracy$$

- Precisión: También conocido como el valor predictivo positivo, correspondiente a la proporción de instancias relevantes entre las instancias recuperadas. En otras palabras, responde a la pregunta ¿Qué proporción de identificaciones positivas fue realmente correcta? Se calcula mediante la fórmula:

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

- Sensibilidad: Corresponde a la tasa de aciertos o tasa positivo real (TPR), es la proporción de la cantidad total de instancias pertenecientes que se recuperaron realmente. Responde a la pregunta ¿Qué proporción de positivos reales se identificó correctamente? Se calcula mediante la fórmula:

$$Recall = \frac{TP}{TP + FN}$$

- Especificidad*:* La especificidad, también conocida como tasa negativa real (TNR), mide la proporción de negativos reales que se identifican correctamente como tales. Es lo opuesto a la sensibilidad. Se calcula mediante la fórmula:

$$Specificity = \frac{TN}{TN + FP}$$

- Puntuación F1: El puntaje F1 es una medida de la precisión de una prueba, es la media armónica de precisión y recuperación. Puede tener una puntuación máxima de 1 (precisión y recuerdo perfectos) y una mínima de 0. En general, es una medida de la precisión y robustez del modelo. Se calcula mediante la fórmula:

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}$$

En la librería de Scikit-learn se encuentran implementadas todas estas métricas, de modo que no se necesita calcular manualmente cada fórmula. Llevadas a código se obtienen los siguientes resultados:

```python
# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print('Exactitud: '+ str(round(accuracy, 2))+'\n')

# Missclasification rate
missclasification_rate = round(1-accuracy, 2)
print('Tasa de error: '+str(missclasification_rate)+'\n')

# Precision
precision = precision_score(y_test, y_pred, average=None)
precision = list(np.around(np.array(precision),2))
print('Precisión: '+ str(precision)+'\n')

# Recall
recall = recall_score(y_test, y_pred, average=None)
recall = list(np.around(np.array(recall),2))
print('Sensibilidad: '+str(recall)+'\n')

# F1 Score
f1 = f1_score(y_test, y_pred, average=None)
f1 = list(np.around(np.array(f1),2))
print('Puntaje f1: '+str(f1)+'\n')
```

```
Exactitud: 0.99

Tasa de error: 0.01

Precisión: [0.98, 1.0, 0.98, 0.97, 0.99, 0.99, 0.99, 1.0, 1.0, 1.0, 0.99]

Sensibilidad: [0.99, 1.0, 1.0, 0.99, 1.0, 0.99, 0.98, 1.0, 0.98, 0.94, 0.97]

Puntaje f1: [0.98, 1.0, 0.99, 0.98, 0.99, 0.99, 0.98, 1.0, 0.99, 0.97, 0.98]
```

Figura 48: Código para métricas con Scikit-learn.
Fuente: Elaboración propia.

Las listas siguen el orden de las categorías de eventos presentado en la matriz de confusión.

De modo que los resultados son:

- Exactitud ≈ 0.99
- Tasa de error ≈ 0.01

Tabla 6: Métricas de evaluación del modelo Linear SVC.
Fuente: Elaboración propia.

Categoría	**Precisión**	**Sensibilidad**	**Puntaje f1**
Automated Testing Tools	0.98	0.99	0.98
Command and Control	1.0	1.0	1.0
Credential Access	0.98	1.0	0.99
Defense Evasion	0.97	0.99	0.98
Discovery	0.99	1.0	0.99
Execution	0.99	0.99	0.99
Lateral Movement	0.99	0.98	0.98
Normal	1.0	1.0	1.0
Other	1.0	0.98	0.99
Persistence	1.0	0.94	0.97
Privilege Escalation	0.99	0.97	0.98

Además de estas medidas existen otras, quizá de menor importancia, pero que suelen obtenerse también a la hora de pedir a un lenguaje de programación orientado a la Ciencia de Datos (Python, R, Julia) que entregue un resumen del rendimiento de un clasificador a través de métodos de la matriz de confusión. Estas serían:

- Prevalencia (Prevalence)
- Índice de Jaccard (Jaccard Index)
- Tasa de Falsos Positivos (False Positive Rate, FPR)
- Tasa de Falsos Negativos (False Negative Rate, FNR) (Se usa poco o nada)
- Ratio de Predicciones Positivas (Positive Predictive Ratio, PPR) (En realidad es la precisión)
- Ratio de Predicciones Negativas (Negative Predictive Ratio, NPR) (También usado con poca frecuencia)
- Tasa de Verdaderos Positivos (True Positive Rate, TPR) (En realidad es la sensibilidad…)
- Tasa de Verdaderos Negativos (True Negative Rate, TNR) (En realidad es la especificidad)

Para conocer como implementar estas métricas revisar Scikit-learn *metrics*.

Como se pudo apreciar con anterioridad, el algoritmo presenta una tasa de error de 0.1 aproximadamente, y esto se ve reflejado en la matriz de confusión dado que no es una matriz diagonal, sino que presenta valores numéricos distinto de 0 fuera de la diagonal, lo que indica la presencia de predicciones realizadas de manera incorrecta, o más conocido cómo errores de clasificación. Afortunadamente estos errores de clasificación no están relacionados con la categoría “Normal”, lo que fue previamente mencionado al decir que el algoritmo no presenta falsos positivos ni falsos negativos, sino más bien errores de clasificación entre los tipos de eventos maliciosos.

Mediante código se puede obtener los errores de clasificación específico, entregando los siguientes resultados:

Tabla 7: Errores de clasificación.
Fuente: Elaboración propia.

Real	**Predicción**	**Eventos N°**
Execution	Automated Testing Tools	8733 - 8734 – 11359 - 11360
Persistence	Command and Control	17125
Defense Evasion	Credential Access	7353
Lateral Movement	Credential Access	12942
Other	Credential Access	15229
Persistence	Credential Access	16474
Privilege Escalation	Credential Access	17772- 18009 - 18011
Credential Access	Defense Evasion	4520
Lateral Movement	Defense Evasion	11792 - 12916
Other	Defense Evasion	15710 – 15711 – 15712 - 15713
Persistence	Defense Evasion	16951 - 16954 - 16955 - 16956 17162 - 17166 – 17176
Privilege Escalation	Defense Evasion	17771
Defense Evasion	Discovery	7352
Other	Discovery	15719
Persistence	Discovery	17190 - 17192
Lateral Movement	Execution	11489 - 12688 - 12924
Persistence	Execution	17189
Privilege Escalation	Execution	18866 - 17585
Automated Testing Tools	Lateral Movement	760
Defense Evasion	Lateral Movement	7138
Discovery	Lateral Movement	7518
Privilege Escalation	Laretal Movement	17802
Defense Evasion	Privilege Escalation	4623 - 7127
Lateral Movement	Privilege Escalation	12949

4.3 COMPARATIVA DE LA SOLUCIÓN PROPUESTA FRENTE A SOLUCIONES SEMEJANTES

En la presente subsección se tiene como objetivo comparar la solución propuesta frente a alternativas similares en el mercado. Para lo anterior, se consideran dos perspectivas, las herramientas *open source* con las que se puede comparar el funcionamiento debido a la igualdad de condiciones en el desarrollo, y los *softwares* de pago desarrollado por empresas, con los que se puede validar los altos costos mencionados como un problema a abordar durante la memoria.

Dentro de las alternativas *open source* disponibles se encuentra:

- LogonTracer: Es una herramienta para investigar inicios de sesión maliciosos mediante la visualización y el análisis de registros de eventos de Windows Active Directory. Esta herramienta asocia un nombre de host (o una dirección IP) y un nombre de cuenta que se encuentra en eventos relacionados con el inicio de sesión y lo muestra como un gráfico. De esta manera, es posible ver en qué cuenta se produce el intento de inicio de sesión y qué host se utiliza. (LogonTracer, 2022)

  Frente a esta herramienta, a favor de la propuesta de solución implementada se tiene que esta abarca una mayor cantidad de tipos de eventos, y no solo asociadas a inicio de sesión. No obstante, al ser una herramienta con un foco tan específico, LogonTracer posee mayor detalle al visualizar los resultados.

- EvtXHunt: Complemento de la herramienta Autopsy que puede analizar los registros de EVTX de Windows en una biblioteca de reglas SIGMA. EvtXHunt también puede funcionar como una herramienta de interfaz de línea de comandos para cumplir el mismo propósito de analizar archivos EVTX contra un conjunto de reglas SIGMA[21]. (EvtXHunt, 2021)

  EvtXHunt posee una interfaz gráfica pero debido a que está incorporado dentro de una herramienta más grande como un plugin[22], podría generar confusión para un usuario que busca solo la funcionalidad específica que se está comparando. De igual modo se puede ejecutar de manera individual a través de la línea de comandos, pero esto quita la comodidad de la visualización de los resultados. Finalmente, la aplicación no presenta actualizaciones desde el 2021, siendo desarrollada en Python 2.X, una versión ya obsoleta.

[21] Son un formato de firma genérico y abierto que nos permite describir eventos de registros relevantes de una manera directa.
[22] Es una aplicación o complemento a un código principal con el que se añaden funcionalidades extra y mejoras al funcionamiento de dicho código.

- Zircolite: Permite ejecutar reglas SIGMA de manera muy rápida en los ficheros de *log* con formato EVTX. Por tanto, las reglas SIGMA se ejecutan sobre un fichero de *log* con el objetivo de comprobar/detectar aquellas condiciones que se han establecido en la regla, siendo esta un archivo en formato YAML. En esta regla se pretende encontrar el patrón que hay definido en la sección *selection*, la cual está dentro del apartado *detection*. Dicho patrón pretende buscar las cadenas que contengan cadenas de caracteres establecidas. Cuando esto ocurra, la regla dirá que ha encontrado una coincidencia. (Marín, 2022)

  Dentro de las alternativas estudiadas, esta resulta ser la más completa y similar a la propuesta de solución presentada, debido a que se entrega como una interfaz de usuario, evalúa los registros de eventos con las reglas SIGMA, reconoce su categoría mediante el mismo repositorio EVTX-To-MITRE-Attack y entrega los resultados con la opción de descargarse en formato Excel.

  El factor diferencial más importantes es que esta herramienta está más enfocada a los registros de eventos del sistema operativo Linux, y no presenta un *dataset* tan amplio para detectar su categoría como el propuesto en esta memoria. Se destaca positivamente la agradable interfaz gráfica que presenta, aunque se debe considerar que este es un proyecto con más de 2 años de trabajo y 12 desarrolladores, por lo que es una buena referencia para aplicar mejoras a nuestra solución.

Por otro lado, pare revisar las herramientas de pago similares se hizo uso de la plataforma GetApp, la cual se autodefine como un ecosistema de aplicaciones empresariales y una plataforma de descubrimiento de software. Cuya misión es asesorar a los profesionales para ayudarles a encontrar el software y las aplicaciones que mejor se adapten a sus necesidades. Dentro de las aplicaciones para la monitorización de la infraestructura TI, donde se incluye el monitoreo de registros de eventos, se encuentran las herramientas OpManager y Pandora FMS.

Dado que lo que nos interesa de esta comparación son los precios de mercado, la plataforma nos entrega la siguiente información:

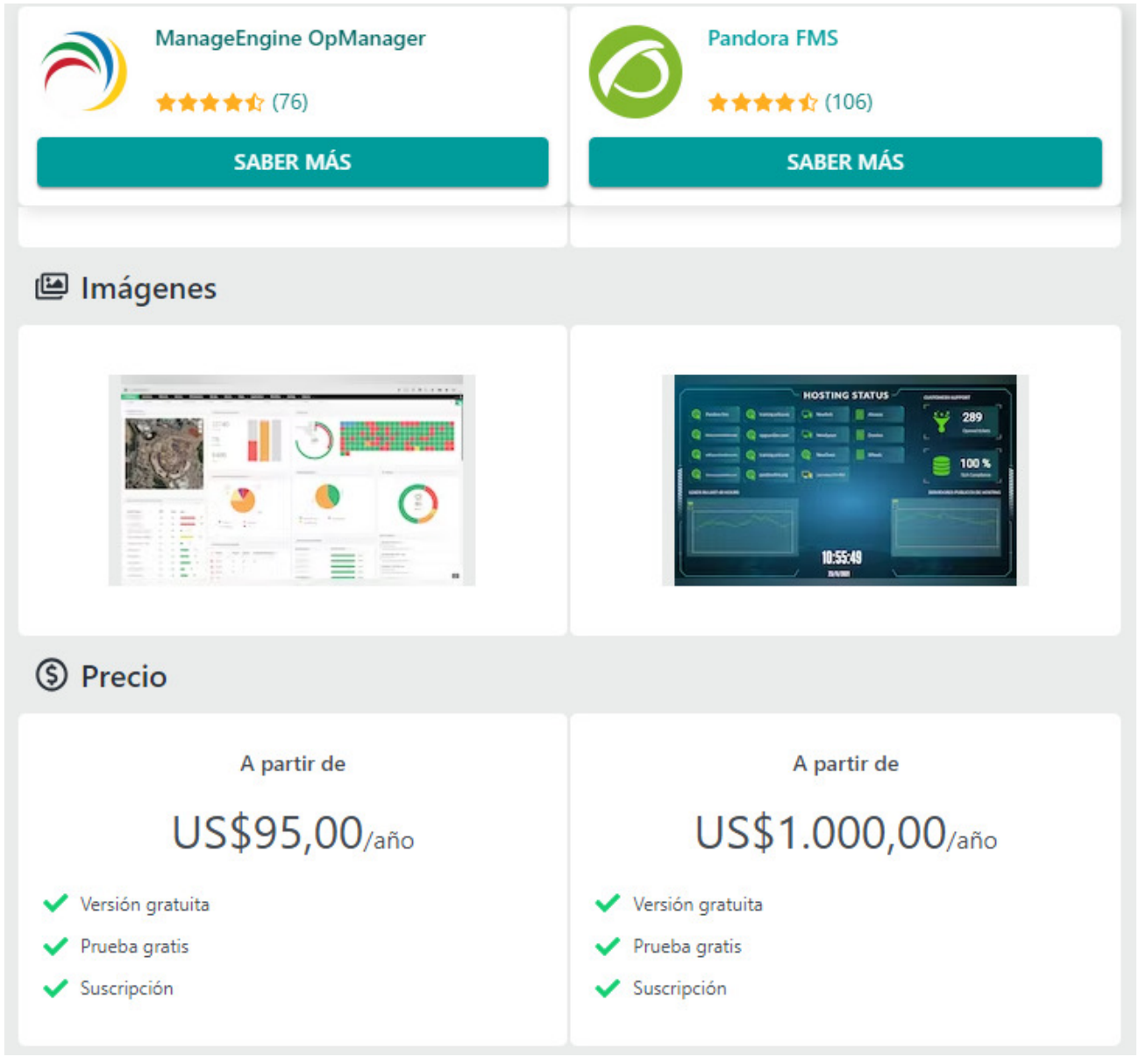


Figura 49: Código para métricas con Scikit-learn.
Fuente: GetApp.

No se expresa explícitamente, pero revisando las páginas oficiales de ambas aplicaciones se tiene que estos precios son por cada equipo o computador, y en el caso de Pandora FMS, exige contratar un mínimo de 50 dispositivos, lo que estaría excluyendo microempresas que cuenten con una baja cantidad de computadores.

Para dar cierre a esta subsección, se puede concluir que la propuesta de solución presentada se encuentra a la par o entrega beneficios extra en comparación a las herramientas *open source* disponibles, y que, frente a las herramientas de pago, resulta una alternativa útil para casos de pocos dispositivos o microempresas que no destinan recursos a la ciberseguridad.

CAPÍTULO 5: CONCLUSIONES

Es sabido que anexo a la transformación digital de las empresas, es decir, su crecimiento tecnológico, viene añadido el aumento de su superficie de ataque, encontrándose dentro de los vectores de ataque a explotar por los ciberdelincuentes, las vulnerabilidades de *software*, sistema operativo y *firmware*. Esto resulta preocupante, debido a que la ciberdelincuencia se ha convertido en una de las mayores amenazas para la seguridad y privacidad en el mundo digital. Se trata de un conjunto de actividades ilegales que utilizan la tecnología para perpetrar delitos, dañar sistemas informáticos, robar información y cometer fraude.

Frente a esto, la ciberseguridad ha tomado relevancia en la industria, llevando a las empresas a invertir en esta área. Lo más frecuente es que la implementen mediante la contratación de servicios externos ofrecidos por otras empresas que se dedican a la consultoría u ofrecer *softwares* orientados a detectar vulnerabilidades. Esto debido a que dentro de los beneficios de delegar servicios se encuentra la flexibilidad que le brinda a la misma encomendar ciertos procesos o servicios que no son su valor principal, con el objetivo de ser más productivo en el correspondiente.

Retomando el vector de ataque mencionado, una de las medidas utilizadas para afrontar dichas vulnerabilidades consiste en el monitoreo de registro de eventos, siendo una parte esencial del mantenimiento del sistema operativo, dado que proporciona una visión detallada de lo que está sucediente en el sistema y ayuda a los administradores a identificar y solucionar problemas de manera oportuna, mejorar el rendimiento del sistema y aumentar la seguridad general. Un registro de eventos, también llamados *logs*, como su nombre indica es un archivo que almacena eventos, que contienen información con respecto a sucesos ocurridos en el sistema.

Si bien existen diversos sistemas operativos, en el presente trabajo se desarrolló la propuesta de solución en torno al sistema operativo Windows, debido a alta popularidad y uso en la industria en general. Dentro de las categorías de registros de eventos de Windows, el eje de desarrollo fue los de seguridad, que contiene información de acciones como inicio de sesión no autorizados, fallo de inicio de sesión, cambio de contraseña, detección de *malware*, entre otros.

Con relación a lo mencionado, las empresas debiesen invertir en la ciberseguridad, no obstante, no todas son conscientes de ello, no destinan recursos económicos para este propósito o bien no tienen capacidad para hacerlo. Por ello, la propuesta de solución desarrollada en esta memoria consistió en un algoritmo de inteligencia artificial basado en la metodología *Threat Detection* para el monitoreo de los registros de eventos del sistema operativo, con el propósito de ayudar a las microempresas a protegerse frente a la ciberdelincuencia de manera gratuita con un *software open source*.

El *software* desarrollado está escrito en el lenguaje de programación Python, ya que tiene gran afinidad con la ciencia de datos y *el Machine Learning*, siendo este la aplicación de la inteligencia artificial implementada. Esta disciplina se define como la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano. El *Machine Learning* es un subcampo de las ciencias de la computación y una rama de la inteligencia artificial cuya finalidad es desarrollar técnicas que permitan a las computadoras aprender. Y la metodología que emplea el algoritmo se denomina *Threat Detection* ya que se buscan amenazas conocidas.

Para la solución propuesta, el algoritmo de *Machine Learning* se basa en el aprendizaje supervisado de clasificación múltiple, por el hecho de que como su nombre lo indica, un registro de eventos está compuesto por diversos eventos, por lo que en el compendio puede existir más de un ataque malicioso. Por otro lado, se dice que está basado en el aprendizaje supervisado porque el algoritmo es entrenado a partir de un *dataset* de eventos etiquetados. Por tanto, el primer paso en la construcción de la solución fue crear el *dataset* a partir de registros de eventos con etiquetas conocidas, para lo cual se hizo uso de dos repositorios de GitHub que contienen *logs* maliciosos identificados, y por parte propia, agregar *logs* con comportamiento normal. De esta forma se generó un *dataset* con un total de 18876 eventos, compuesto por 11 categorías.

Posteriormente, se comenzó a desarrollar el algoritmo, para lo que inicialmente se necesitó definir el modelo de *Machine Learning*. Dentro de las opciones que ofrece Scikit-learn, la librería de Python empleada para el aprendizaje automático, se optó por utilizar el modelo *Linear Support Vector Machine*, por sus buenas referencias y mayormente, porque al realizar una comparación rápida con los datos frente otros modelos, se obtuvo un valor de 0.74 de *accuracy*. Con la ayuda de Google Colab se creó el código del modelo, ya que es una herramienta eficiente para ir probando el modelo, permite visualizar distintos datos de la data, como la distribución de la misma, unigrama y digrama, y aplicar métricas de validación para el modelo, comenzando por la matriz de confusión y con ella valores como la exactitud, tasa de error, precisión, sensibilidad y puntuación F1.

Con el modelo generado, el siguiente paso fue llevarlo al editor de código, para comenzar a desarrollar la interfaz gráfica con la que el usuario interactuaría. Previo a esto, es necesario definir las funcionalidades de la aplicación, para lo que esbozó el diagrama de flujo (ver figura 12), el cual nos orienta al momento de desarrollar. Con todo esto, se llevó a cabo la implementación de la solución, obteniendo una aplicación capaz de recibir un registro de evento, en su formato Evtx, para procesarlo mediante el modelo de *Machine Learning* y predecir con una exactitud del 99 % su categoría, pudiendo además descargar un informe compuesto por un archivo pdf y uno Excel que especifican la cantidad de eventos maliciosos encontrados, el número de evento correspondiente y una predicción de la posible subcategoría del evento.

Por tanto, los resultados obtenidos en base al desarrollo de la propuesta de solución es un *software open source* capaz de predecir la(s) categoría(s) de un registro de eventos con una exactitud del 99 %. Además de esto, el usuario tiene la posibilidad de descargar un informe detallado con la ubicación de cada evento malicioso que posteriormente puede ser revisado con el Visor de Eventos de Windows. Ahora bien, para validar la solución propuesta se presentaron dos enfoques, uno desde la programación, es decir, el funcionamiento correcto del código; y una mirada desde la inteligencia artificial, lo que se traduce en métricas numéricas de las predicciones.

Desde el punto de vista del funcionamiento de la aplicación, esta es capaz de recibir un archivo en formato Evtx y dependiendo del peso del mismo, y la cantidad de eventos que posea entregar una predicción. Para registros de eventos pequeños, considérese de 50 eventos o menos, el programa tarda segundos. Por otro lado, con grandes volúmenes de eventos, por ejemplo, 750 eventos, puede llegar a tardar hasta 20 minutos. Si bien esto puede ser considerado una limitación, no se debe perder de vista el objetivo principal que es identificar correctamente las categorías presentes en un *log* malicioso. En relación con el lenguaje de programación escogido para el desarrollo de la solución, se reafirma la buena elección por las buenas referencias que tiene en el uso del *Machine Learning* y porque al comparar la solución con otras similares en el mercado, se puede observar que se encuentran escritas en Python.

Con respecto a la segunda perspectiva, evaluando el algoritmo con métricas extraídas de la matriz de confusión. De manera sencilla se puede entender la matriz de confusión como una matriz cuadrada del tamaño de las categorías, en nuestro caso de 11x11. La diagonal de la matriz contiene los valores acertados en la predicción y los que se encuentran fuera de ella corresponden a clasificaciones equivocadas. Para la construcción de la matriz de confusión, como en cualquier algoritmo de *Machine Learning*, se destinó un porcentaje de la data total para entrenamiento y otra para el testeo. En nuestro caso se consideró apropiado utilizar una repartición de 80-20 %, respectivamente. A partir del conjunto de datos para el testeo se generó la matriz de confusión, pudiendo observar que el modelo no presenta falsos positivos ni falsos negativos, esto quiere decir que no se reconoció ningún *log* malicioso como normal, ni ningún *log* normal como malicioso. Lo que evita que la matriz de confusión sea una matriz diagonal son los errores de clasificación, siendo casos en los que una categoría de evento maliciosa es reconocida como otra. Afortunadamente los casos son escasos, validando el buen entrenamiento del algoritmo, y siendo visible en métricas como la exactitud, tasa de error, precisión, sensibilidad y puntuación F1.

Con todo lo expuesto con anterioridad es que se dan por validados los objetivos propuestos inicialmente, tanto el general como los específicos, entregando algoritmo de inteligencia artificial que analiza los registros de eventos de manera automatizada, evaluándolo en base a su funcionamiento y métricas propias de la IA.

Dentro de los alcances y limitaciones de la propuesta de solución entregada, se encuentra que está desarrollada para el sistema operativo Windows, y con foco en los registros de eventos de la categoría Seguridad. Además de esto, una limitante durante el proceso de desarrollo fueron los propios componentes del computador, específicamente la memoria RAM, ya que después de los 20 MB de tamaño del *dataset* era incapaz de procesarlo. Del mismo modo, para registros de eventos de gran volumen y presencia de eventos maliciosos, el algoritmo llega a tardar demasiado. Por tanto, como trabajo a futuro se considera llegar a abarcar todas las categorías de los registros de eventos, es decir los de tipo SAM, SOFTWARE, SYSTEM Y NTUSERDAT, pudiendo tener un *dataset* específico para cada una. En relación con el tamaño del conjunto de datos, una alternativa sería levantar una máquina virtual con mayor cantidad de recursos en una plataforma como AWS, por ejemplo, para desde un inicio tener *dataset* de mayor tamaño, y la opción de incorporar al modelo el *log* entregado por el usuario, de forma que el algoritmo esté en continuo progreso, siempre y cuando no se desvalúen las métricas calculadas a partir de la matriz de confusión.

Por otro lado, y con una perspectiva más ambiciosa, se puede decir que desde el inicio de la memoria se fijaron los lineamientos y objetivos para llevarla a cabo, pero en el transcurso del desarrollo y de dar contexto a la propuesta de solución se abarcaron más conceptos que aún son alcanzables en el presente trabajo. La idea sería llevar la metodología *Threat Detection* a *Threat Hunting,* a modo de revisar en tiempo real los *logs* y que estos no deban ser cargados de manera manual, presentando así una solución proactiva, alertando al instante un ataque.

Finalmente, para dar cierre a esta memoria se puede concluir que la relevancia de la propuesta de solución presentada se respalda con el hecho de que la ciberdelincuencia se ha convertido en una de las mayores amenazas para la seguridad y privacidad en el mundo digital. Las empresas tienen una gran responsabilidad en la prevención y protección contra el cibercrimen, ya que tiene el deber de garantizar la seguridad de los datos de sus clientes y empleados. Esto implica la adopción de un enfoque proactivo en la protección de la información. Para ello, el *software* entregado surge como una alternativa para empresas pequeñas o medianas que no poseen de un presupuesto para protegerse contra amenazas digitales, además de poder ser usada de manera sencilla, eliminando el factor humano que debe revisar cada *log* de manera manual, por un sistema que lo revisa de manera automática y detecta la categoría del evento malicioso que pueda estar ocurriendo.

BIBLIOGRAFÍA

Amazon Web Services. (2022). *¿Qué es Python?* Obtenido de AWS: https://aws.amazon.com/es/what-is/python/

Ambit. (10 de diciembre de 2020). *SOC: qué es y cómo implantarlo en tu empresa.* Obtenido de Ambit: Building solutions together: https://www.ambit-bst.com/blog/soc-qué-es-y-cómo-implantarlo-en-tu-empresa

Arboleda Díaz, J., & Medina Ruiz, L. (2017). *Machine learning applied to threat intelligence.* Escuela Colombiana de Ingeniería Julio Garavito. Obtenido de https://repositorio.escuelaing.edu.co/handle/001/702

Arce, J. I. (26 de julio de 2019). *Health Big Data*. Obtenido de La matriz de confusión y sus métricas: https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

Cárdenas, D. (2022). *Necesito consultoría en ciberseguridad y no sé a quién contratar.* Obtenido de DATAWARDEN: https://www.datawarden.com/blog/necesito-consultoria-en-ciberseguridad-y-no-se-a-quien-contratar

Chou, J. (23 de Abril de 2021). ¿Cuánto cuesta realmente la ciberseguridad? *Entrepreneur*. Obtenido de https://www.entrepreneur.com/es/tecnologia/cuanto-cuesta-realmente-la-ciberseguridad/405745

CISCO. (2022). *¿Qué es la ciberseguridad?* Obtenido de CISCO: https://www.cisco.com/c/es_mx/products/security/what-is-cybersecurity.html

Colab, G. (2023). *Google Colab*. Obtenido de https://colab.research.google.com/?hl=es#scrollTo=UdRyKR44dcNI

E-DEA NETWORKS. (31 de Enero de 2019). *Herramientas para el monitoreo de logs: ¿Cómo elegirlas?* Obtenido de EDEA-NETWORKS: https://www.e-dea.co/blog/gestion-de-logs

EDS Robotics. (15 de Febrero de 2021). *EDS Robotics*. Obtenido de Tipos de aprendizaje en la Inteligencia Artificial: https://www.edsrobotics.com/blog/tipos-aprendizaje-inteligencia-artificial/#:~:text=el%20ejercicio%20práctico.-,¿Qué%20tipos%20de%20aprendizaje%20existen%20en%20el%20ámbito%20de%20la,y%20el%20aprendizaje%20no%20supervisado.

EvtXHunt. (7 de noviembre de 2021). *GitHub*. Obtenido de EvtXHunt: https://github.com/Lyc4on/EvtXHunt

Figueroa Fernández, V. (9 de Junio de 2021). Estructura de seguridad para el análisis gestión de logs de aplicaciones multicapa. *Ciencia Sur, 6*(7). Obtenido de http://dicyt.uajms.edu.bo/revistas/index.php/ciencia-sur/article/view/726

Galindo, D. R. (8 de Marzo de 2021). *Aprendizaje supervisado*. Obtenido de Centro de Innovación Industrial en Inteligencia Artificial (CII.IA): https://www.ciiia.mx/noticiasciiia/aprendizaje-supervisado-1

Gamboa Suarez, J. (Agosto de 2020). Importancia de la seguridad informática y ciberseguridad en el mundo actual. Colombia. Obtenido de http://repository.unipiloto.edu.co/handle/20.500.12277/8668

Guerrero, J. (8 de septiembre de 2010). *Panda Security*. Obtenido de Falsos Positivos: ¿y eso qué es?: https://www.pandasecurity.com/es/mediacenter/malware/falsos-positivos-y-eso-que-es/

Hinestroza Ramírez, D. (2018). *El Machine Learning a través de los tiempos, y los aportes a la humanidad.* Obtenido de https://repository.unilibre.edu.co/handle/10901/17289

Hispasec. (18 de junio de 2008). *Falsos negativos y falsos positivos. Problemas para desarrolladores*. Obtenido de https://unaaldia.hispasec.com/2008/06/falsos-negativos-y-falsos-positivos-problemas-para-desarrolladores.html

IBM. (2020). *¿Qué es la ciberseguridad?* Obtenido de IBM: https://www.ibm.com/cl-es/topics/cybersecurity

IBM. (2022). *¿Qué es Machine Learning?* Obtenido de IBM: https://www.ibm.com/cl-es/analytics/machine-learning

IBM. (2022). *¿Qué es una superficie de ataque?* Obtenido de IBM: https://www.ibm.com/es-es/topics/attack-surface

IBM. (2023). *IBM*. Obtenido de ¿Qué es la detección de amenazas?

Juan, J. (24 de abril de 2023). *La ciberdelincuencia sigue en aumento: los ciberataques se multiplican*. Obtenido de EY España: https://www.ey.com/es_es/cybersecurity/la-ciberdelicuencia-sigue-aumento-los-ciberataques-se-multiplican

Kaspersky. (2022). *¿Qué es la ciberseguridad?* Obtenido de Kaspersky: https://latam.kaspersky.com/resource-center/definitions/what-is-cyber-security

KeepCoding. (24 de Agosto de 2022). *¿Cómo programar Inteligencia Artificial?* Obtenido de KEEPCODING: https://keepcoding.io/blog/como-programar-inteligencia-artificial/#:~:text=programar%20Inteligencia%20Artificial.-,1.,como%20para%20el%20Deep%20Learning.

KeepCoding. (10 de Mayo de 2022). *¿Qué son los logs y para qué sirven?* Obtenido de KeepCoding: https://keepcoding.io/blog/que-son-logs-y-para-que-sirven/

LogonTracer. (20 de diciembre de 2022). *GitHub*. Obtenido de LogonTracer: https://github.com/JPCERTCC/LogonTracer/tree/master

Marín, C. (16 de junio de 2022). *Zircolite – Ejecución de reglas Sigma en ficheros EVTX*. Obtenido de https://derechodelared.com/zircolite-ejecucion-de-reglas-sigma-en-ficheros-evtx/

MathWorks. (2023). *MathWorks*. Obtenido de Reconocimiento de patrones: https://la.mathworks.com/discovery/pattern-recognition.html

Microsoft. (2023). *¿Qué es SIEM?* Obtenido de Microsoft: https://www.microsoft.com/es-es/security/business/security-101/what-is-siem

Microsoft. (2023). *Especificaciones y límites de Excel*. Obtenido de https://support.microsoft.com/es-es/office/especificaciones-y-límites-de-excel-1672b34d-7043-467e-8e27-269d656771c3

Microsoft. (2023). *Microsoft*. Obtenido de Tareas básicas en Excel: https://support.microsoft.com/es-es/office/tareas-básicas-en-excel-dc775dd1-fa52-430f-9c3c-d998d1735fca#:~:text=Excel%20es%20una%20herramienta%20muy,la%20cuadrícula%20de%20las%20celdas.

Parlamento Europeo. (8 de Septiembre de 2020). *¿Qué es la inteligencia artificial y cómo se ocupa?* Obtenido de Parlamento Europeo: https://www.europarl.europa.eu/news/es/headlines/society/20200827STO85804/que-es-la-inteligencia-artificial-y-como-se-usa

Phddirection. (2019). Obtenido de Pattern Recognition in Python: https://www.phddirection.com/pattern-recognition-in-python/

Phoronix. (25 de Octubre de 2022). *Phoronix*. Obtenido de https://www.phoronix.com/review/python-311-performance

Randori. (2022). The State of Attack Surface Management 2022. Obtenido de https://www.randori.com/reports/the-state-of-attack-surface-management-2022/

Rimol, M. (7 de Marzo de 2022). *Gartner Identifies Top Security and Risk Management Trends for 2022.* Obtenido de Gartner: https://www.gartner.com/en/newsroom/press-releases/2022-03-07-gartner-identifies-top-security-and-risk-management-trends-for-2022

Rodríguez Rama, J. (2018). *Aplicación de técnicas de machine learning a la detección de ataques.* Obtenido de https://openaccess.uoc.edu/bitstream/10609/81126/11/jmrodriguez85TFM0618memoria.pdf

Rouhiainen, L. (2018). Inteligencia Artificial. Obtenido de https://static0planetadelibroscom.cdnstatics.com/libros_contenido_extra/40/39308_Inteligencia_artificial.pdf

Ruiz, D., & Sánchez, M. (Noviembre de 2020). SINGULAR BANK: Proceso de Threat Hunting en Microsoft Azure Sentinel, más que búsquedas... *Ciberseguridad, seguridad de la información y privacidad*(142), 64-65. Obtenido de https://revistasic.es/sic142/revistasic142.pdf

Scikit-learn. (2023). *Scikit-learn*. Obtenido de https://scikit-learn.org/stable/modules/naive_bayes.html

TIBCO. (2023). *TIBCO.* Obtenido de https://www.tibco.com/es/reference-center/what-is-supervised-learning#:~:text=El%20aprendizaje%20supervisado%20es%20una,de%20manera%20explícita%20dónde%20buscar

ANEXOS

https://www.threathunting.net/files/framework-for-threat-hunting-whitepaper.pdf

https://www.bbva.com/es/machine-learning-que-es-y-como-funciona/

https://www.edsrobotics.com/blog/tipos-aprendizaje-inteligencia-artificial

https://www.xataka.com/robotica-e-ia/deep-learning-que-es-y-por-que-va-a-ser-una-tecnologia-clave-en-el-futuro-de-la-inteligencia-artificial

https://www.phddirection.com/pattern-recognition-in-python/

https://www.alibabacloud.com/blog/deep-learning-vs-machine-learning-vs-pattern-recognition_207110

https://www.simplilearn.com/pattern-recognition-and-ml-article

https://www.youtube.com/watch?v=mMc_PIemSnU&list=PLqXS1b2lRpYS7EYHlQ-r-IfxqGteTgCng&index=2

https://aprendeconejemplos.org/python/aprendizaje-automatico-con-scikit-learn-parte-i-clasificacion

https://sitiobigdata.com/2018/08/27/clasificacion-texto-multiclase-scikit-learn/amp/

https://www.sciencedirect.com/science/article/abs/pii/S0167404821000456